A Location-Based and Augmented Reality Hidden Object Game

Interactive Media and Game Development

A Major Qualifying Project Report
submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Bachelor of Science

By:

Marco Duran
LilyAnne Lewis
Sienna McDowell

Advised by:

Professor Jennifer deWinter

Abstract

This report describes the design and creation process of *Step Symphony*, a location-based hidden object game using directional sound and augmented reality developed to fulfill the Major Qualifying Project requirement for Worcester Polytechnic Institute's Interactive Media and Game Development (IMGD) and Professional Writing majors. This project was developed over the course of three months at Ritsumeikan University's Biwako-Kusatsu Campus in Shiga Prefecture, Japan. This game intends to encourage exploration of the many historical sites of Japan, and to experience joy while collecting different kyokubou with different instruments. This report covers all aspects of the team's development process, including research, playtesting, and the creation of the various features the game possesses, as well as the possibility of future work on the project. We also developed a testing system that could be utilized to evaluate the usability and learnability of all key components of *Step Symphony*, and thereby refine each gameplay component to make them easier and more fun to use while exploring Central Kyoto.

Acknowledgments

The creators of *Step Symphony* would like to thank the many people who have helped us on the creation of this game.

First, we would like to give a heartfelt thanks to Professor deWinter, our adviser, for her help and monitoring our progress. Without her advice and input, this game would be a completely different beast.

We would also like to thank Professor Lopez and Professor Noma who oversaw our work while in the care of Ritsumeikan BKC. Their warm welcomes and assistance helped shape the game's core concepts. An especially warm thanks goes out to the other students in the Media and Design Lab—their enthusiasm made this difficult undertaking bearable, and we hope to be able to keep in contact with them.

Another thanks goes out to Ritsumeikan University Biwako-Kusatsu Campus, for housing us while in Japan, and giving us a space in the lap and providing everything we needed to make this project a reality.

Last and certainly not least, a deep thanks to our families, without whom we would not have been able to do this at all.

Table of Contents

List of Figures

1 Introduction

Step Symphony is a location-based mobile game in which players can search for run-away creatures throughout Central Kyoto. These creatures, known as kyokubou (short for *kyoku no bouame*, or Musical Candy), are colorful animals in Japanese attire that play instruments. In order to find and catch these kyokubou, players must utilize panning directional music as well as their phone's camera to locate them. While searching for the kyokubou, the player is accompanied by the monkey kyokubou Sarushi.

We wanted to create a mobile application that would teach the user more about Japanese culture, and in turn get them interested in learning more themselves. While exploring their surroundings, players will be able to make new kyokubou friends in the real world by following the music the kyokubou play, as well as seeing them through the phone's camera using augmented reality. Players will be able to play and create music with their new kyokubou friends. Kyokubou play instruments that are specific to Japanese culture, such as the Shamisen. In addition, kyokubou can teach the player about certain locations in Kyoto, giving the player more locations to explore. Exploring these new locations will allow the player to find new kyokubou friends, this starting the cycle anew.

Step Symphony is designed to run on an Android or iOS smartphone with a WiFi or data connection. *Step Symphony* also requires a pair of earbuds or headphones that are able to differentiate left and right audio, as the only way of finding kyokubou is to listen for their music as it pans around the player.

The basic story of *Step Symphony* begins with all of the kyokubou had gathered to practice for their upcoming concert. Unfortunately, Sarushi, the monkey, got too excited and

played his taiko drum too loudly, scaring all of the other kyokubou away. In order to find them and calm them down, Sarushi needs to listen to them playing their instruments. However, since Sarushi is deaf, he cannot do that alone. With their superior hearing the player collaborates with Sarushi to find kyokubou by listening for their playing music, locating the kyokubou, and gathering them all together again so they can continue practicing for their concert. Currently, there are 16 kyokubou hidden around Central Kyoto.

When the player gets close enough to a run away kyokubou, he or she will be able to hear that kyokubou playing. By following the music, the player can find the Kyokubou and start to play along with them. When the kyokubou has calmed down, he or she will come along with the player.

Our goal for this project was to create an early build of a playable, location-based mobile application. We wanted *Step Symphony* to get players engaged in, and excited about, learning about popular tourist locations and historical sites in Central Kyoto. We also wanted our game to fit the feeling of these historical sites, by using music and/or instruments that can be heard in these areas.

This report discusses our development process, as well as testing, revisions, and future plans. Chapter 2 discusses the design and gameplay for *Step Symphony*. This includes our planning and target audience, our research on popular tourist attractions, and more details about Sarushi and his fellow kyokubou. The design choices in this section are important, as they are the foundation for *Step Symphony* as well as the player's overall experience.

Chapter 3 talks about the creation of our 2D visual assets for *Step Symphony*. In this chapter we go in depth about the artistic style we chose, as well as why we chose it. We delve

into more detail about our color scheme for our UI and kyokubou, and the meanings behind those colors in Japanese culture.

In chapter 4 we go over the audio for our game. We discuss about how each audio track was composed, as well as the style we had in mind while composing it. We also go into details about the instruments we chose to use, which include the taiko drum, the shinobue, and the shamisen.

Chapter 5 discusses the technology and mechanics we developed for *Step Symphony*. This chapter explains how we developed each mechanic in Unity Engine and Android SDK, including directional sound, snapshot mode, the in-game map, and the rhythm game capture mechanic. We also discuss our kyokubou and user databases created using Python and Django.

Chapter 6 goes over our testing process and results for *Step Symphony*. During testing, faced two problems: Testing in an uncontrolled environment and the language barrier prevalent when working in a foreign country. In order to overcome these difficulties, we decided to do two different tests; one test took place in a controlled environment, and one test will take place in an uncontrolled environment. During testing, we attempted to overcome the language barrier by using simple English and illustrations for our surveys and instructions. We also had a professor on standby if the tester has a question or needs clarification. We then discuss the actual process for both tests, as well as observations and feedback from our results, and how we will use this feedback for future revisions of the game.

Chapter 7 covers the postproduction of the game, where we discuss the team's post-mortem. We go over what went well, what did not, and what we would have changed in our design process looking back on it. We also discuss future plans, what we would like to work on

if we continued to develop *Step Symphony*. This was an important reflection for our team as a whole, as well as our entire experience abroad.

2 Design and Gameplay

This section of the report goes into more detail about our design and planning phase. Included is all of our conducted research and design opinions made as a team. When first designing *Step Symphony*, our design and personal goals were to have a completely playable game. This playable game included:

- Working directional audio to guide the player, as well as an accompanying map screen
- Augmented reality using the phone's camera to find the kyokubou in the real world
- A mechanic to capture the kyokubou to add to the player's collection
- A collection screen where the player could view their collected kyokubou, as well as play their respective notes

Our constraints for *Step Symphony* included:

- *Step Symphony* had to be a mobile application
- We had to incorporate a database written using Python and Django
- *Step Symphony* needed to be location-based

2.1 Platform and Hardware Requirements

We were given a challenge to make a location-based mobile application that uses databases with Python and Django. In order to play *Step Symphony*, the player must have either an Android or IOS smartphone. These smartphones must either have a Wi-Fi or data connection, and their built-in GPS must be fully functional. Because our game relies heavily on directional sound, players must also have a pair of earbuds or headphones that can differentiate between left and right sound.

2.2 Target Audience

Our initial challenge of a location based mobile application gave us a lot of freedom in choosing our target audience. Currently, *Step Symphony* only takes place in central Kyoto, which is a popular area for tourists because of its historically significant aesthetics. We decided that because Central Kyoto was a popular area for tourists, *Step Symphony* would have a target audience of young adults who are visiting Kyoto for the first time because kyokubou could be placed near popular tourist attractions. To research the best way to design this, we looked at similar games such as *Pokémon Go* (Niantic Inc. 2016) and *Ingress* (Niantic Inc. 2012). We also visited popular areas in Central Kyoto to find the best locations to place kyokubou.

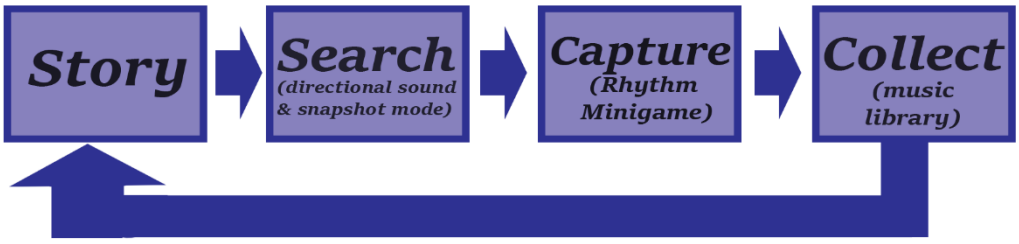


Figure 1: Flow Chart of *Step Symphony*

2.3 Kyokubou

Creatures you find and capture in the game are known as kyoku no bouame (曲の棒飴, lit. Musical Candy or “kyokubou” for short. They're named as such because of their colorful clothing, and the sweet-sounding music they play.

The kyokubou were intended to show some of the colors and aesthetics of Japanese culture to capitalize on the environment and location we made the game in. As such, each

kyokubou is wearing some kind of traditional Japanese garment. Making the kyokubou animals is a decision rooted in where we decided to take reference from, as well as Japanese Shintoism.

Mascots are quite important in Japan. Almost everything seems to have a mascot from brands to sports teams, and even cities, prefectures, and landmarks have a cute little character to represent them. The most popular mascots, like Kumamon, have events dedicated to them. Most notably, the mascot character is usually an animal or an object, and is very rarely a person. In keeping with that theme, the kyokubou are also all animals with slightly more humanoid proportions. All of the kyokubou, as well as their names and a short description about them, can be found in Appendices B and C.

The connection to Shintoism comes because we didn't want to just pick random animals. We wanted to pick animals that would have some connection to Japan, and some symbolism in its culture. Therefore, we chose animals that we've seen during our time in the country, and animals that made appearances in temples and on landmarks, like foxes, rabbits, and cats.

Currently, there are only two different groups of kyokubou—shamisen and shinobue because those are the two instruments we have in the game, and the player can play their music from the collection screen once they have befriended them. From the collection screen, it is also possible to read a description of the kyokubou, including their name and their relationship with where they were found and other kyokubou they interact with.

Our main character is a kyokubou called Sarushi, named after the monkey island of Sarushima (猿島, lit. monkey island). Sarushi is the only monkey kyokubou, as well as the only kyokubou that plays the taiko drum.

Sarushi acts as the player's guide, and his taiko is present as the player explores on the map screen, almost as if he is exploring with the player.

The game's plot is set in motion by Sarushi's drum-playing. All the kyokubou are together to practice their music, but in his excitement, Sarushi plays his drum too loud and scares all his friends away. The taiko is known for its loud, powerful sound, and is historically used as a tool on the battlefield and as a way to give orders. It is also associated with Raijin, the god of thunder, whose drum is the source of thunder's loud sound.

Figure 2: Sarushi

Sarushi was chosen to be a monkey because monkeys are very human-like. They can be bipedal or quadrupedal, as well as having opposable thumbs and large eyes. All of these things make monkeys easy to emphasize with, and also makes them easier to anthropomorphize and animate.

We were told by our fellow lab students that, in Japanese culture, monkeys were once seen as sacred mediators between gods and humans. Nowadays, the perception of monkeys in Japan has shifted radically from being one of reverence to being one of scorn. Monkeys have a reputation of mindlessly imitating human behavior or a scapegoat for negative human traits. This was easily seen during our visit at the monkey park at Arashiyama.

However, since our game is aimed more towards tourists and people visiting Japan, we chose to use a more western interpretation of the monkey, as a creature that is cute and friendly that occasionally pulls pranks on others.

2.4 Kyokubou Locations

We hid kyokubou near or in popular tourist locations in Central Kyoto. These locations ranged from parks, such as Umekoji Park, to temples, such as Honnoji Temple. Each kyokubou can be found at three different coordinates, usually close together or around the same location. Figure 3 below shows the location of each site, the hours of operation, as well as the cost of entry, and Figure 4 shows a map of Central Kyoto with the locations of all the kyokubou.

Name	Location	Hours of Operation	Cost
Honnoji Temple	Central	9 am – 5 pm	Free
Nijo Castle	West	8:45 am – 5 pm	600 yen
Myoshinji Temple	North West	9 am – 5 pm	Free
Kyoto Imperial Palace	North	9 am – 5 pm	Free
Nishi Honganji Temple	South West	5:30 am – 5 pm	Free
Higashi Honganji Temple	South West	5:30 am – 5 pm	Free
Yasaka Shrine	East	Always Open	Free
Kiyomizu-dera	East	6 am – 6 pm	400 yen

Figure 3: Locations with cost and hours of operation

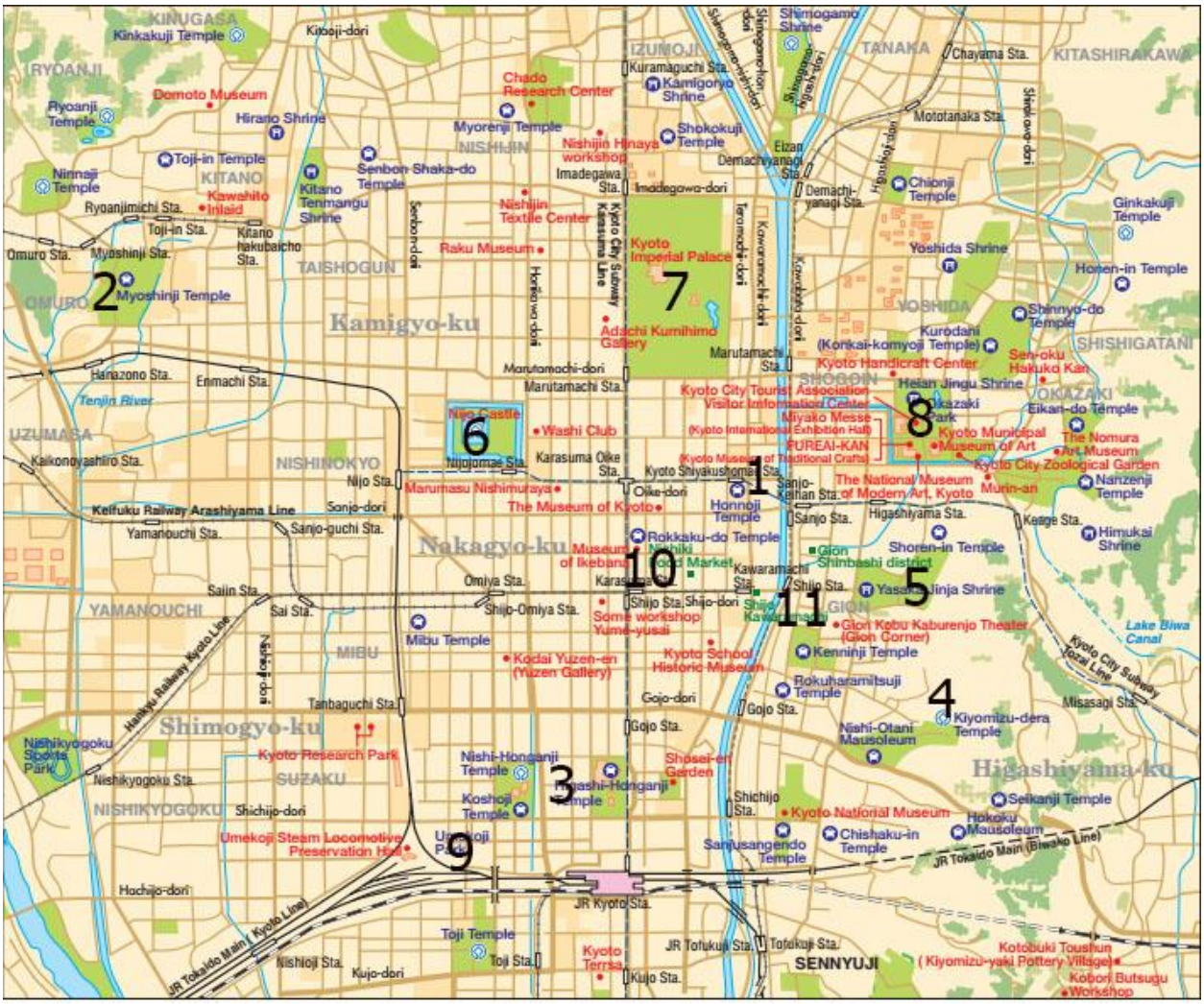


Figure 4: Kyokubou Locations

Honnoji Temple

Honnoji Temple, also known as Honnō-ji, is a Buddhist Temple located just south of Kyoto City Hall, located at the number 1 on the map. We chose this temple because unlike most other temples, it is hidden within the city and is also significantly smaller than the other temples in Kyoto. Another reason why we chose this area is to show the player a significant location in Japanese history during the 16th century. On June 21st, 1582, Oda Nobunoga was killed here during the Honnōji Incident. On this day, Nobunoga's most trusted vassal daimyo, Akechi Mitsuhide, led 13,000 troops against Nobunaga and proceeded to burn down Honnoji Temple and the surrounding buildings while he was inside (McMullin 93, 1984).

Figure 5: Honnoji Temple

On the outside of the temple, there is a small area where there is a fountain and is also a place where many tourists gather. In the game, one of the kyokubou, Kun-Kun, is located here and will be feeding the fish that are located in the fountain. We felt that since there were many people in this location that were also feeding the fish, it would be a great place to put a kyokubou.

Figure 6: Kun-Kun

Myoshinji Temple

Myoshinji Temple is a Buddhist temple located in the most North-Western area of Central Kyoto, labeled 2 on the map, and consists of seven or eight different buildings around a central area. We chose to place a kyokubou in this area because it was far away. Most of the

other locations we looked into were in the central and South-Eastern area, and we wanted to spread out our kyokubou locations.

Myoshinji Temple, we decided, was a perfect place for one kyokubou because there are many long pathways in between the buildings, so kyokubou can be placed along these pathways and away from the main buildings. We decided to place our Akabeko, or red cow, kyokubou there, as the Akabeko is prevalent in Japanese Buddhist lore. The Akabeko, named Ushikyuu, enjoys watching the sun rise from the front steps of the main temple.

Figure 7: Ushikyuu

Nishi and Higashi Honganji Temple

Nishi and Higashi (West and East) Honganji Temple are about a 20-minute walk from Kyoto Station, and are labeled as number 3 on the map. Today, Nishi Honganji serves as the head temple of the Jōdo Shinshū organization. The temple is split into two separate sections about a block or two apart.

When visiting these sites, we found them to be places of quiet prayer and learning. In one location, people were quietly praying in the main building. In the other location, there was a ceremony being held. For both locations, we were required to remove our shoes before entering any building.

We decided to only have one kyokubou placed for both of these temples. In order to not disturb the peace, the kyokubou, our rooster, Osachi, will be placed near the entrances, far away from the main buildings. Osachi works at both temples, sweeping the stairs to the main areas of prayer and running messages between the East and West temple.

Figure 8: Osachi

Kiyomizu-dera

Kiyomizu-dera temple, named after the pure water the temple it's built around, is a very popular Buddhist temple. It wraps around Mt. Otowa, housing over 20 buildings in various places, including shops and small shrines, and is located at number 4 on the map. The shrine was build in 778, and was rebuilt after being destroyed by fire several times.

Figure 9: Ai

Aside from the water itself, one of the attractions to Kiyomizu-dera are the love stones. The love stones are two large stones that stand on opposite sides of a path, with shops on either side. The goal is to walk from one stone to the other with one's eyes closed. If someone can touch the other stone without help, then they will have luck in their interactions with other people. One of our kyokubou, Ai, is found near the Love Stones, where every day she tries to cross to the opposite stone, and proceeds to fail.

Another draw to Kiyomizu-dera is its popularity. Historically, the temple has been open to people of all social backgrounds, and the same is true today. Kiyomizu-dera teems with people of all ages, and the wait to drink its mystical water can be an hour or longer. The kyokubou Komiya is found around Kiyomizu-dera for this reason, as they like to be around people.

Figure 10: Komiya

Yasaka Shrine

Yasaka Shrine, also known as the Gion Shrine, is a popular location during the month of July when the Gion Matsuri is taking place. Yasaka Shrine is located at the number 5 on the map. Visitors often come to this shrine to pray for success in business, the well-being of their family, and protection from illness (Cali, Dougill, and Ciotti, 2013, 143).

During the month of July, especially on the 16th and 17th, there are multiple celebrations that take place around the Gion District, during Gion Matsuri. During this festival, locals are in charge of many preparations, including making floats for the parade on the 17th (Cali, Dougill, and Ciotti, 2013, 147).

Because of the popularity of this location during Gion Matsuri, as well as the public park located right next to the shrine, there was enough space to place zones for two different kyokubou.

Figure 11: Yasaka Shrine (Yanajin33, 2015)

The two kyokubou located at Yasaka Shrine are our two fox kyokubou, Kataaki and Kokoko. Kataaki and his daughter Kokoko often perform there together during Gion Matsuri.

Figure 12: Kataaki and Kokoko

Nijo Castle

Nijo Castle was built in 1603, and currently stands in the center of the Rakuchu region (Sheldon, 26). It is also easy to get to from Kyoto Station, just three stops away on the San-In Line, and is labeled as number 6 on the map.

We as a team decided that Nijo Castle would be a good place to place two kyokubou, as there is a lot of attractions inside the castle grounds, including gardens, ponds, and the castle itself. Unfortunately, it is not free to visit. Currently, it costs 600yen to enter the castle grounds. As a compromise, we decided to place one of the kyokubou near the entrance, so players could find that one without having to pay.

The two kyokubou located at Nijo Castle are Fuwari and Shoji. They both enjoy greeting visitors into Nijo Castle. Sometimes, when the weather is nice, they can be found in the gardens, where Shouji will tell tales and Fuwari will dance to them.

Figure 13: Shouji and Fuwari

Kyoto Imperial Palace

We chose Kyoto Imperial Palace due to its location, size, and affordability. It is labeled as 7 on our map Built in 794, the grounds stretch 110,000 square meters, the walled palace is surrounded by a large garden of grass and trees. The Kyoto Palace was the center of Japan's government until the capital was moved to Tokyo, and has been carefully preserved since. Entering the palace is also entirely free, which makes it a good spot to put a kyokubou because everyone will be able to access it without much difficulty or added costs.

Kyoto Imperial Palace is located in the middle of the city near a subway station. Its proximity to the city is why Kumagamine can be found there. He frequents it because his school is located nearby. Kumagamine enjoys taking naps in the open grassy areas around the palace.

Figure 14: Kumagamine

3 Visual Assets

Below is an explanation of the art assets that were made and used in *Step Symphony*. We used primarily 2D assets in the game.

3.1 Art Style

The 2D art assets in *Step Symphony* were made in Photoshop. Each asset was drawn by hand, from sketching to completion, using real animals found in Japan as a form of reference. The proportions of the kyokubou were based on Japanese mascot characters, most notably the ones used for baseball, like Buffalo Bell and Frep the Fox. Japanese baseball mascots enjoy a respectable amount of popularity, and are the face of their respective teams. The mascots often come in familial brother-sister pairs, like Buffalo Bell and Buffalo Bull from the Orix Buffaloes, or Tsubakuro, Tsubami, and Torukuya from the Tokyo Yakult Swallows. This is reflected in their designs with kyokubou sharing a species often being related to each other, like Kokoko and Kataaki, who are both foxes.

Figure 15: Buffalo Bull and Buffalo Bell, mascots for the Orix Buffaloes. (Wikimedia, 2011)

The art of the kyokubou reflect this with similar proportions and having the presence of families between some kyokubou. These design decisions were made to give the kyokubou a more grounded feeling due to their slightly more humanoid proportions working in harmony

with their animal features. The exceptions to this rule are the rooster and the salamander. Their unique features do not lend themselves as easily to anthropomorphism, and as such their proportions and bodies are much closer to that of their normal animal selves.

Figure 16: Kyokubou

Each kyokubou's pose and position was designed with the intention of having them look visual distinct from each other, while also showing off their animal features. This means each kyokubou's silhouette is different enough that the player can tell them apart. This also allows the kyokubou to have their own vibrancy and sense of life, as each pose intends to show as something alive or otherwise in motion somehow. Most of the kyokubou are given very distinct hands or arms, which creates the feeling that they can actually play their associated instrument.

The clothing of the Kyokubou also ties into the instrument they're associated with. The shamisen, which was at one time associated with the lower class Japanese, is most often played by kyokubou who are wearing more simplistic clothes. One of the Shisa Kyokubou is a farmer, which was once considered a low working class. The pig kyokubou plays a shamisen despite being more ornately dressed because he is intended to represent a kabuki actor, where the shamisen was often used as musical accompaniment. By comparison, kyokubou playing the flute often are dressed in fancier clothing, and in some cases have a religious connotation to them. One of the foxes plays a flute, and is dressed as a Shinto shrine maiden. Another is dressed as a Buddhist monk. The shinobue is more strongly associated with Shintoism than Buddhism, but the two religions tend to blend harmoniously in Japanese culture.

The main color scheme of the kyokubou pulls heavily from the colors of fall in Kyoto, which is a rich tapestry of reds, oranges, browns, yellows, and muted/desaturated greens. Kyoto's fall colors are considered iconic for their relative warmth and subtlety as compared to the well-

known bright pinks of the sakura blossoms in spring. The rich, warm colors of the fall season mix well with the temples in the Rakuchu district of the city where *Step Symphony* takes place, which is home to many temples that have color schemes that are rich in browns and reds. If the blooming of the sakura blossoms represents new beginnings and youth, then Kyoto's autumn can be considered to represent maturity and the new beginnings that come with adulthood. This serves our game well, since the target audience of our game are older people, most likely in their early-to-mid-twenties.

Figure 17: Kiyomizu-dera temple in the fall. (Wikimedia, 2008)

3.1.1 Color Meanings

The colors of our game are taken from the colors of fall in Kyoto. The game uses Kyoto's fall colors because of the popularity of the season. With many historical sites that sport warm shades of red and brown, the changing leaves adds golden yellow and vibrant orange, creating a view of natural beauty and warmth that mixes with the spiritual and historical feeling of Kyoto's many temples and castles.

Just like in other countries, colors have specific meanings in Japan. The meaning of these colors, as well as the natural coloration of the animals they're based on, were considered carefully when choosing the colors of the kyokubou, as well as the colors of the kimonos they wear.

Here's a list of the common colors that were used for the kyokubou, and their symbolic meanings in Japanese culture. We got this information from other students in the lab via a short interview:

Red:

Red is often associated with fire, communism, and embarrassment to Japanese people. Red is one of the four "basic" Japanese colors, symbolizing brightness and clarity in contrast with the color blue, which symbolizes "faintness". The color red is also tied to Japanese spiritualism and appears prominently in Shinto shrines, often on archways and worn by shrine maidens.

Figure 18: The color red is often used on these gates, which are used in shrines.

Brown:

Brown doesn't have any special meaning to Japanese people, though it is associated with the earth, and dirt.

Orange:

Orange also doesn't have any special connotation to Japanese people, though staff at businesses sometimes wear orange on their uniforms to separate them from customers.

White:

Often associated with innocence and purity, white is generally perceived as a good color. It is also contemporarily associated with doctors and scientists, but in a positive way. White is also the color of the summer uniform of middle and high school students, as a color of youth.

Black:

Black is a highly formal color. While it's worn for funerals, it can also be worn for weddings. As such, it's a color that has both a negative and a positive meaning. To Japanese people, black can also be “strong” or “sharp”. It also can mean “darkness” or “evil”.

Yellow:

Yellow can be used to mean something nice, fresh, and new. In contemporary Japan, it is most often used as a sign to warn other people. Children often wear yellow hats and have yellow backpacks, to make them easier for drivers to see.

Figure 19: Japanese children wearing yellow school hats (Wikimedia, 2005)

Green:

Green normally has a meaning of “natural” or “safe”. The color green tends to make one think of forests and nature.

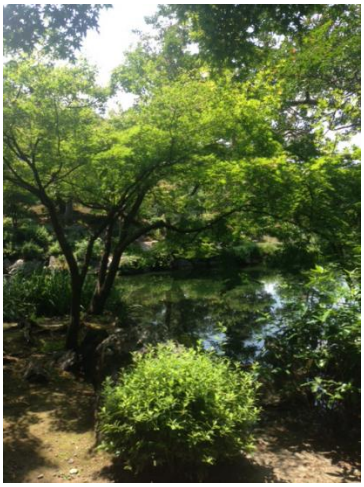


Figure 20: The color green often symbolizes nature.

3.2 2D Assets

The 2D art assets of *Step Symphony* were made entirely in Photoshop from concept to fully colored completion. Each kyokubou was referenced against a picture of the animal it is based off of, as well as a type of traditional Japanese clothing. For the UI elements, their colors and styles are based off making them seem like cuts of gemstones.

For kyokubou, the process begins with gathering references for the animal that we want the kyokubou to be. Gathering multiple references gives the artist a more complete picture of what they want to draw, and gives them the information they need to create with the fullest extent of their abilities. Using reference liberally, a page full of sketches are created.

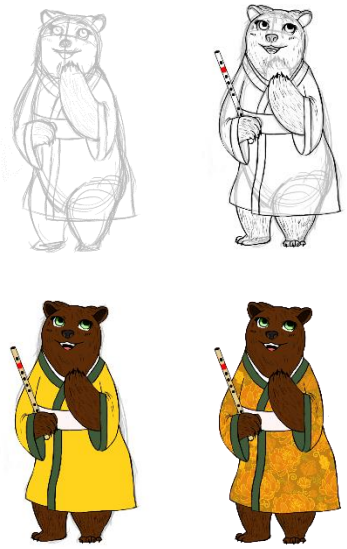


Figure 21: Creation of Yushiya

Making many sketches was important, because it allowed multiple quick iterations of the same basic idea. Afterwards, we decided which image from the seven or eight sketches we liked the best. That image would be further refined and eventually given color, and would be the final kyokubou asset used in-game. Figure 10 shows the process behind one of the kyokubou, from sketching to finalized, game-ready version. This process was also used for the UI.

Figure 22: Ringo (above) compared to Kun-Kun (below).

In some cases, a kyokubou would only be given a simple, flat color due to time constraints. However, other kyokubou were given more detailed designs on their clothing, to give them an added degree of polish. In some cases, the design was hand-drawn, like with Ringo's kimono, but in other cases it was another image overlaid onto the drawing, like with Yushiya and Kun-Kun. This was done to emulate Japanese woodcuts that we researched in the beginning, where the patterns had little to no deformation around the character's body.

Images were overlaid by being brought into Photoshop and placed on a separate layer above the basic color of the kyokubou. From there the layer style, opacity, color, and sometimes the design itself would be tweaked to suit the 'character' of the kyokubou.

For some kyokubou who did not get a fully detailed design for their clothing, shading was added. The shading was done using a layer above the basic colors set to multiply, with an opacity of about 30%. Adding shading to the kyokubou was tricky, since they would be seen in the real world, and therefore making the shading make sense was important. We decided to go with the light source being “above” the kyokubou, since the sun would most often be above the player. Some kyokubou were not given a design or a shadow on their clothing due to time and manpower constraints.

Figure 23: UI for the rhythm minigame.

How the creation of the UI differed was in moving from sketches to line art. Much of the game's UI is geometric: made of squares, circles, and triangles. When deciding which of the sketches to use as our UI, we would often pick elements of each iteration we liked, and put them together. The UI tended to have a more painterly look to them when compared to the kyokubou, which tended to be more simplistic in their colors. The most obvious example is the rhythm game screen, where you can see a kyokubou right next to the more UI, which looks more painterly. These styles, although they look different, are given some cohesion because they share a color palette consisting mostly of reds, yellows, oranges, and browns.

4. Audio

Step Symphony is reliant on having an inviting soundscape that surrounds the player with Japanese culture. In order to give the player a sense of what it feels like be in Japan, each interaction the player makes will be sound based. Within in this section will be an explanation of how the music for the game was composed, the style of music, and the instruments that were used within the production of the game.

4.1 Composition

Throughout the game, there are many opportunities to fully encapsulate the sound of Japan through musical composition. Each instrument has a specific way for which it is to be played within an ensemble. In 1964, an ensemble was created called *Pro Musica Nipponia,* which had a goal to incorporate a western orchestra to play Japanese instruments (Miki 2008). Based off of the research done with regards to how music is composed for each of the instruments, we have tried to keep the songs within the same key and time signatures as traditional Japanese music.

Because most of traditional Japanese music is taught orally and focused on smaller ensembles of instruments, this has challenged us to think about how each song should be written. Because European style of modes are taught in America, we had to completely change the style of composition. Instead of using traditional European Modes, we had relied on using five Japanese scales called *Yō, Ro, Ritsu, In,* and *Ryūkyū* (Miki, 2008). These scales all relied heavily on a natural mode and only had six notes including the octave. Luckily, each of these keys did not rely on using complex key signatures, thus making the music easier to compose than anticipated. Since most Japanese instruments are created for a specific key signature, each scale

has only six notes and at most use three octaves. The total amount of notes that can be used in a song is eighteen notes. If composed in a European style, most European instruments can change the key that they play in, making for a greater variation of notes that can be used in a song (Miki, 2008). With less notes to work with, the patterns in composing each song can remain similar in pitches used, but can become more complex with variations in the rhythm.

Each of the instruments that were picked in the sections below each have a unique element in the way that music can be composed for them. Music composed for the Taiko Drum was mainly inspired my Japanese Taiko Drum players. Music that has incorporated the use of the shamisen is influenced by the Tsugaru Shamisen style and also based off of artists such as the Yoshida Brothers and Agatsuma Hiromitsu. Since the styles of music are varied throughout the game, the player will receive a better understanding of the different styles of music that are located here in Japan.

4.1.1 Composition Style

Below is an explanation of the composition style used during the composition of the music in *Step Symphony*. The style that is predominately featured within the game is the Tsugaru Shamisen style.

Tsugaru Shamisen

Tsugaru shamisen is a style of play that was developed in 1877 by a blind entertainer called Nitabō in the Tsugaru region. Nitabō was able to transpose songs and narratives that were native to the Tsugaru region despite his disabilities. The most notable aspects of the development

of the style were the changes made to the instrument itself. Generally, most songs in the Tsugaru region were played on the hozozao (medium sized) shamisen. When Nitabō started to write, he transposed the music to the futuzao (large sized) shamisen. The other changes made to the instrument include a change to how the plectrum was used in the play of the shamisen; more specifically, the beating and slapping of the strings (Miku 2009, 110).

One of the main gameplay mechanics in *Step Symphony*, directional sound, is an homage to most of the famous musicians with visual disabilities, more specifically, Nitabō. Since the directional sound aspect does not rely on being able to see where you are in the game, we hope that the player can understand the way Nitabō has lived his life. All of the songs that will be featured in the capture portion of the game will be written with the style of Tsugaru shamisen in mind.

The Tsugaru shamisen style became popularized in the rest of Japan around the late 1990's with artists like the Yoshida Brothers and Agatsuma Hiromitsu. These artists also popularized the name of the style since the term *Tsugaru shamisen* was not used until the 1950s and 1960s within Japan (Johnson 2006, 84). This style is renowned worldwide due to a large emphasis on improvisation and energetic tempo, similar to that of the rock and roll guitar style. Because of the unique sound, Tsugaru shamisen became a large symbol of Japanese culture despite its early upbringing (Johnson 2006, 83).

4.2 Instruments

Listed below are the instruments that were used to compose the music found within *Step Symphony*. We have included four instruments within the soundscape of *Step Symphony*, the Sanshin, Shamisen, taiko drum, and the Shinobue. Due to time constraints, we would have liked to include more instruments such as the Ichigenkin, Wooden Fish and the Koto. These instruments are predominantly found within Japanese culture and also within *Step Symphony*.

Sanshin

The sanshin is a three-stringed lute, similar in appearance to a banjo, which is most commonly found in Okinawa. It is closely related to the sanxian, a Chinese instrument that also has three strings (Gillan, 2009). It is believed that that the sanshin came to Okinawa from China near the end of the fourteenth century. However, documentation showing the appearance of the sanshin does not show up until the mid-sixteenth century (Matsue, 2016).

This instrument is played by plucking each string individually with a pick, and is usually accompanied by vocals (In the Living Room, 2014). Sheet music is not written as notes and scales, but rather as Chinese characters (Craft, 2016).

Figure 24: Sanshin (Universidad Nacional de La Plata, 2012)

The sound that the sanshin makes depends heavily on the materials used, such as the type of wood, as well as how tight the snakeskin is stretched over the base. The neck is made from

one piece of timber. More expensive sanshin are made from the wood of ebony trees from Okinawa, and usually cost around $4,000 USD. Cheaper sanshin are often made from mulberry wood stained black, and cost around $200 USD. The body is made stretching two pieces of python skin over a wooden frame (Gillan, 2009).

While the sanshin became popular in the early 21st century, this wasn't always the case. Up until the mid-twentieth century, usually only the elite played the sanshin (Gillian, 2009). Because it takes many years to master playing the sanshin, the working class would see this as taking time away from 'real work', and would see sanshin players as undesirable and lazy. The sanshin would later develop into the shamisen, creating a cultural and musical bridge of sorts between Okinawa and the mainland (Matsue, 2016).

Shamisen

The shamisen was developed during the mid-sixteenth century, between the Eiroku and Muromachi periods. The shamisen is a later derivation of the Okinawan sanshin, which was created from a similar instrument from China called the sanxian (Miki, 88). The shamisen was said to be imported into a port in Osaka called Sakai in 1562 (Miki, 89). The shamisen are built

in three different sizes; hosozao (small), chūzao (medium), and futozao (large). The shamisen is labeled as both a string instrument and a percussion instrument.

The body and neck of the shamisen is made from the bark of the red sandalwood or mulberry trees (Fox, 2002). Dog or cat skin is then stretched over the hollowed wood body. Since the skin of the dog and cat are so fragile when stretched, modern shamisen use a plastic skin to cover the body for added durability.

Figure 25: Shamisen (Miri, 2008)

The strings are made from twisted silk and are looped over the end of the neck, or the nakagosaki (Miki, 89). The other end of the strings are wrapped around pegs, which are made from ebony or ivory. These pegs are used to adjust the tension of the strings in order to change the tonality of the instrument. There are three strings on the shamisen and there are 2 pegs extending to the right side of the neck and there is one peg that extends from the left side of the neck. The placement of the tuning pegs stays consistent between the different sizes of the shamisen.

The shamisen is played by using a device called a plectrum. A plectrum is a large pick that was primarily used in the mid-fourteenth century by European lute players. This variation of plectrum can be made from high-grade ivory, deer horn, tortoise shell, or water buffalo horn. Other substitutions include plastic, boxwood or oak (Miki, 88). The plectrum has to be made of a heavy material to ensure that the string is able to be plucked while the plectrum also hits the

body of the shamisen. The body of the shamisen is also known as a taiko drum, which is the reason why the shamisen is listed as both a percussion instrument and a string instrument.

In *Step Symphony*, half of the kyokubou will play the shamisen. This is shown by that kyokubou holding the instrument. Shamisen-playing kyokubou generally look more human and wear mostly reds and browns.

Taiko Drum

Taiko means drum, and refers to many variations of drum playing in Japan. Sizes of the drum range from handheld, to larger than a person (Fujie, 2001). The style of taiko drumming best known today has a short history only beginning in the late 1950's. However, taiko drums have been used in Japan for over 1400 years (The Taiko Resource, 2016).

Two of the first uses of taiko were as a battlefield instrument and everyday village life. On the battlefield, the taiko drum would be used to issue orders to the army, as well as intimidate the enemy, as it could be heard throughout the entire battlefield (The Taiko Resource, 2016). In everyday village life, the taiko drum was used to alert the town of incoming storms, the departure and return of hunters, etc. (Vogel, 2009). The loud volume of the taiko drum has been long associated with the voices of the gods (The Taiko Resource, 2016).

Figure 26: Taiko Drum (Feuchter, Jennifer. 2008)

Our mascot, Sarushi, will be our only Kyokubou to play a taiko drum. At the beginning of *Step Symphony,* Sarushi explains that all of the Kyokubou were together to perform. However, while they were practicing, Sarushi got too excited and started beating his drum loudly. As his drum is roughly the same size as the ones used to issue commands across battlefields, the loud drumming frightened the other Kyokubou away, and now it is the player's job to find them.

The Map Screen will feature a Taiko drum ensemble, as if Sarushi is playing while walking with the player. The tempo of the song is composed at a tempo of 105 beats per minute, which is considered to be andante, or walking tempo (Fallows, 2016). The tempo of this song allows the player to be able to walk in time to the song, without having the feeling of being rushed to find the Kyokubou.

Shinobue

The shinobue, also known as the fue or takebue, is one of the oldest folk instruments known to Japan. Little is known about the origin of this instrument, but there is evidence that different forms of transverse flutes have been used as early as the 8th century CE (Austin and Vetter).

The shinobue is made from shinochiku bamboo, which is also known as shino (Miri, 8; Austin and Vetter). The instrument is hollowed out on the inside to allow the air to flow through the instrument. There is a mouthpiece in the shape of a circle located on the topside of the instrument. Further along the body on the topside, there are six to seven holes carved and spaced

equally. On the bottom side, there is an indentation at the same position as the mouthpiece. At the end of the instrument closest to the mouthpiece, the shinobue is closed off with wax. The other end is left open in order for the sound to resonate. The ends of the instrument are wrapped in birch bark to ensure that the bamboo does not split and to add a decorative touch (Miri, 8, Austin and Vetter).

Figure 27: Shinobue

The shinobue can be produced in twelve different lengths and are usually numbered at the top of the instrument (Shimada, 2009). The lengths of the shinobue change the pitch of the instrument and amount of finger holes present. In orchestral performances, a shinobue player must have many flutes in order to play in different keys (Pro Musica Nipponia). The different flutes allow the shinobue player to be in tune with the other instruments and play a variety of songs (Shimada, 2009).

The shinobue is typically used during matsuri and nagauta performances. Matsuri is a festival setting where Shintō shrines are worshiped and celebrated (Fujie, 1983). During matsuri, the shinobue is played along with instruments that do not have to be tuned, such as the taiko drum. The instruments in this setting are not meant to be tuned. This is due to the fact that a single shinobue cannot be tuned to an ensemble of instruments. Nagauta, meaning "long song" in Japanese, is a musical accompaniment of Kabuki theatre and classical dances (Hughes, 29).

During a naguata performance, the shinobue plays an obbligato with both the shamisen and voice. Thus, the shinobue must be tuned properly with the shamisen and voice parts of a naguata.

Traditionally, the method in which the shinobue was taught was through word of mouth. Around the Meiji period, music for the shinobue became transcribed into a numbered score called sujifu (Shimada, 2009).

In our game, eight out of the sixteen obtainable kyokubou play the shinobue. These kyokubou tend to be related to Japanese spiritualism. Ushikyuu, for example, is an 'akabeko'. We learned while visiting Buddhist temples that an akabeko is a cow that, according to myth, helped build a temple before giving its soul to Buddha and turning to stone.

4.3 Incorporation into *Step Symphony*

Using the instruments, we were able to make our own tracks for multiple core components of *Step Symphony*, as well as each kyokubou's playable note. Finally, our planned final animation was going to have a twenty second track featuring all of the instruments to play along with the animation.

The directional sound portion of *Step Symphony* features three separate tracks. The first track is a fifteen second repeating track featuring only the taiko drum. This plays when the player is not currently in any kyokubou zone, and gives the feeling that Sarushi is playing is taiko drum alongside the player. The other looping two tracks would separately feature a shamisen or shinobue, depending on what type of kyokubou the player is near. If the player enters a kyokubou zone for a kyokubou with a shamisen, then the directional audio the player will hear is the shamisen track. The same goes for a shinobue kyokubou, but rather with a

shinobue track. Snapshot mode does not feature and audio. For our rhythm minigame, we have two separate tracks as well. If the player is trying to calm down a shamisen kyokubou, then a forty-five second shamisen track will play during the rhythm game, and vice versa for the shinobue.

When a kyokubou has been calmed down and collected, the player can listen to them play a note on the collection screen. Each kyokubou was given a note from their specific instrument. For the eight shamisen kyokubou, they were each given one of eight separate notes. The same goes for the shinobue kyokubou. Players can play all of these as many times as he or she wants, as well as overlapping all of the notes.

When a player has collected all of the kyokubou from a certain instrument, then a twenty second track is unlocked featuring solely that instrument. When the player collects all of the kyokubou, then all of those twenty second tracks that were unlocked become one track, and plays along with the final animation reward.

5 Technology

During the planning phase, we kept running into how large the scope of our game was. Because of this, it was important to implement a reusable and extensible code base. Two tech components used for the foundation of the game include the Unity 5 Engine and Android SDK because *Step Symphony* is a mobile application.

5.1 Unity

For development, we used Unity Engine 5, a free and open source game engine. We chose to use Unity over other engines for a number of reasons: the use of audio spatializers, the ability to code in C#, and the use of unique Unity plugins for additional programming functionality. These plugins include UniRX and a JSON parser, which allowed us to connect to our database to store users and kyokubou, as well as MapzenGo to gather data from OpenStreetMap for our player map. Because the game relies heavily on being able to use sound to find direction, an audio spatializer was necessary to be able to create 3D audio around the player. Given all of the game engines that could have been used, Unity was the only free game engine to include an audio spatializer add-on that fit our requirements.

5.2 Player Map and Augmented Reality

Step Symphony is a mobile application that is primarily focused on Augmented Reality or AR. The definition of AR, proposed by Ronald T. Azuma; "AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world." Along with this definition, there were also a few characteristics that specified the composition of an AR

system. These characteristics are as follows: Combines real and virtual, interactive in real time, registered in 3-D (Azuma, 1997). Most of the game incorporates all three of these factors, more specifically, the game's map and "Snapshot" mode.

The map that is displayed in the game encompasses all of the characteristics of an AR system due to GPS of the phone being updated in real time, the map is generated based off of real map data that is present in the world today and the rendering of the streets and buildings are generated in 3-D with a top-down facing camera. Successful games such as *Pokémon Go* (Niantic Inc. 2016) and *Ingress* (Niantic Inc. 2012) created by Niantic, Inc. also incorporate the use of a map but with a different camera perspective and amount of camera control within the application as seen in figure X.

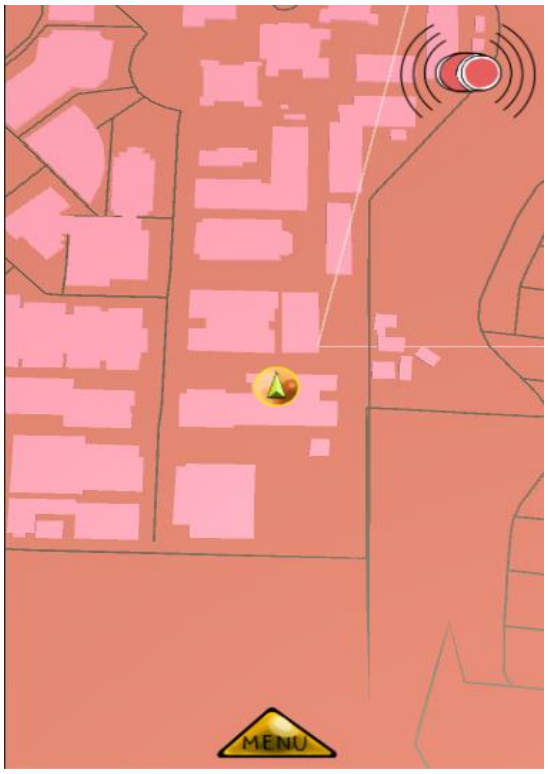


Figure 28: Map Screen

For this game, we wanted the player to be view a map to see the area around them as well as give a general idea of their location. However, we also did not want the map to be too detailed or to give the player hints for kyokubou locations.

To accomplish this goal, we decided to use OpenStreepMap, an open source map that is free to use under an open license, and MapzenGo. MapzenGo is a free to use/alter add-on written by Baran Kahyaoglu, a software engineer from Turkey. This add-on takes data from OpenStreetMap when given a latitude and longitude, parses it through a JSON add-on, and finds portions of the data that were labeled as roads, buildings, or water. These portions are then

stored in their respective factories, which then draws the information on-screen as lines (roads), 2D polygons (water), and 3D polygons (buildings).

The roads, buildings, and rivers are drawn dynamically. Nine tiles, which are fifty by fifty units, are drawn, with the player sprite in the center tile. When the player moves out of the center tile into a new tile, that new tile becomes the center tile and the other eight tiles are re-loaded given the new coordinates.

Unfortunately, at the time of implementation, MapzenGo did not take in GPS coordinates at runtime to update the map. Instead, MapzenGo could only read a manually created latitude and longitude for the player sprite. When the coordinate for the player sprite is set, the sprite can be moved around the map using the computer arrow keys. This was not what we needed for *Step Symphony*. In order to get what we needed, we had to alter the input of latitude and longitude as well as player movement.

The first portion of MapzenGo we altered was the input of player coordinates. We changed the manually entered location coordinate to now read the smartphone's last known latitude and longitude. MapzenGo then takes this coordinate and makes it the center of the middle tile on the map and places the player sprite there. This was a simple task.

The next step was to alter movement so the sprite would move in real-time on the map based on the player's change in latitude and longitude. This was done in multiple steps. First, the latitude and longitude for the center of the tile was stored. Then, the player's new latitude and longitude is taken in. After the player's new location is taken in, we find the difference in meters between the old and new locations, and convert the measurement in meters to units in Unity, based on the scale of the map. The player sprite is then moved that number of units, and

the player's new latitude and longitude becomes the center of the map. Wait a couple of seconds, and the process starts over again.

5.3 Directional Sound

One of the key gameplay components involves the use of directional sound to find kyokubou. Directional sound involves changing the direction the sound pans in the player's headphones. As seen in figure X, when the kyokubou is to the left of the player, the sound will pan to the left. When the kyokubou is to the right of the player, the sound will pan to the right. The farther the player has to turn, the more the music is panned to that one side. When the kyokubou is in front of the player, the sound will play smoothly in both ears.

Figure 29: Visualization of Directional Sound

To do this, the game uses Unity's compass function. Unity takes in the phone's direction and sets it to an angle between 0 and 360 degrees. If the player's phone is facing north, then the set angle is 0 degrees. Likewise, if the player's phone is facing directly south, the set angle would be 180 degrees.

After finding the phone's direction, we find the player's latitude and longitude as well as the closest kyokubou's latitude and longitude. Using the latitude and longitude of both the player and the kyokubou, we find the distance in feet between the two, as well as the direction, in degrees, a player will have to face to be directly facing the kyokubou. A player will only be able to hear the music to locate a kyokubou if they are within 500 feet of one.

As the player walks closer towards the kyokubou, the music playing in their headphones will get louder. When the player is within 100 feet of the kyokubou's location, a button resembling a camera will appear on the map screen (Figure X). Clicking on that button will bring the player to Snapshot Mode, which allows players to capture kyokubou in the real world.

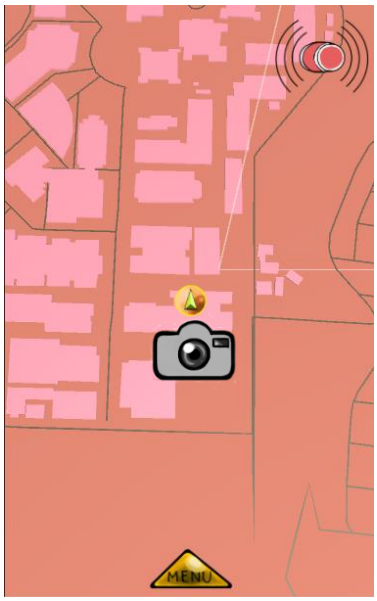


Figure 30: Entering Snapshot Mode from Map Screen

5.4 Snapshot Mode

The "Snapshot" mode is the part of *Step Symphony* where the player uses the back camera of the mobile phone to see where kyokubou are located in the real world. The expected player experience is somewhat similar to the visual effects in the hit 1988 film, *Who Framed Roger Rabbit?* In this movie, 2-D animation of famous cartoon characters interact with real life actors (Marshall, Watts & Zemeckis, 1988). In *Step Symphony*, the interaction between the real world and the virtual world is one between the player and the kyokubou, through the lens of the mobile phone.

The camera rotates around a central point using the gyroscope of the mobile phone. This rotation moves the camera using a quaternion. The reason for using a quaternion is to be able to move along each axis smoothly without the hassle of programming a matrix and Unity has built-in functions to work with quaternions. The rotating camera has a child game object of a display board with a material attached to it called "WebCamTexture." This is a material type within Unity that takes data from a device's camera and superimposes it onto the display board. As

shown in the Figure 31, the display board is placed farther from the rotating camera. In order for the player to see the kyokubou in the scene, the kyokubou must be placed between the camera and the display board, thus placing the kyokubou on a higher draw level than the display board.

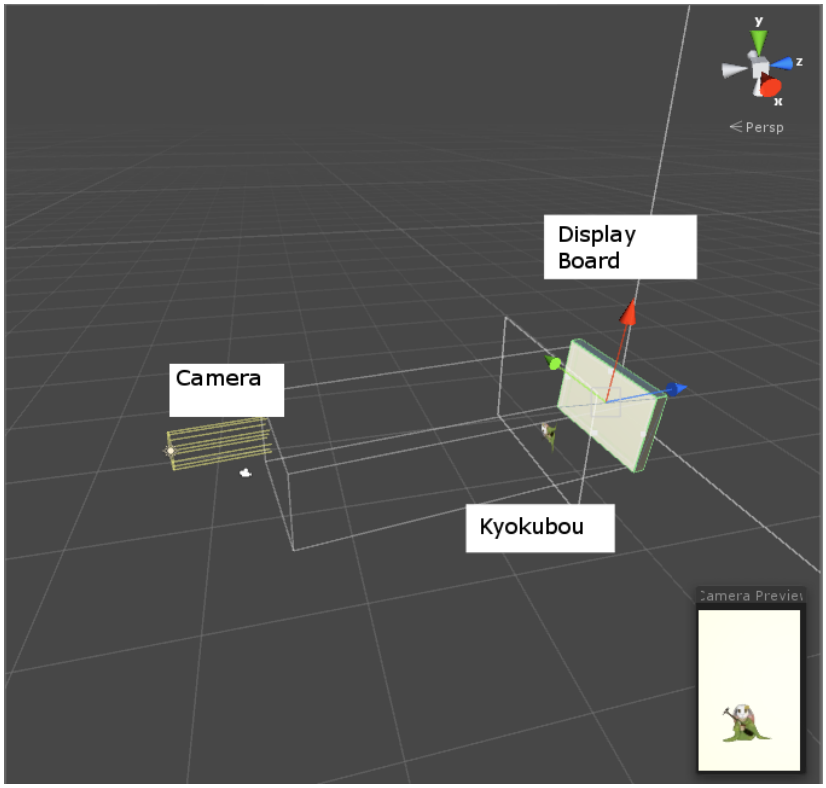


Figure 31: Abstraction of how "Snapshot" mode works in the Unity Engine

When the kyokubou appears on the screen, the way that the player interacts with the kyokubou is to touch it on the screen when the phone is rotated in a certain orientation. When the kyokubou is touched, the game will also move on to the rhythm-based portion of *Step Symphony*.

Since the one of the three characteristics of an AR system defined by Azuma requires the object to be rendered in 3-D, "Snapshot" mode within *Step Symphony* only draws the kyokubou as 2-D drawings in a 4-D space; therefore, this portion of the game is only a partially complete AR system.

Since the one of the three characteristics of an AR system defined by Azuma requires the object to be rendered in 3-D, "Snapshot" mode within *Step Symphony* only draws the kyokubou as 2-D drawings in a 4-D space, therefore this portion of the game is only a partially complete AR system.

5.5 Rhythm Capture Mechanic

When the player exits Snapshot mode, the player will enter the portion of the game where they will have to befriend the kyokubou. The kyokubou will befriend the player if they can successfully play music alongside the kyokubou. During this portion of the game, a song will play and icons which are shaped like gems, called notes, will move from the top of the screen and move towards the bottom of the screen. As seen in figure X, there are two other components called “Note Spawners” and “Note Catchers.”

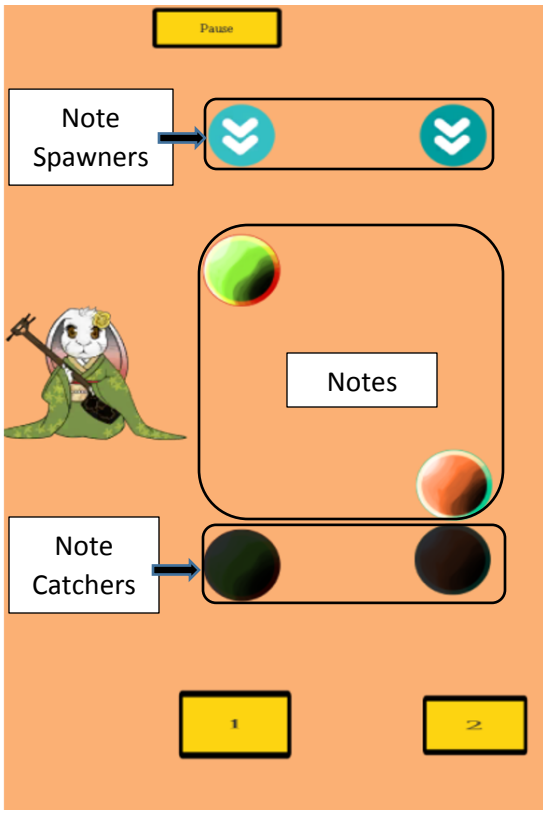


Figure 32: Overview of Rhythm Capture Mechanic

The “Note Spawners” are at the top of the screen and generate notes. These “Note Spawners” will generate these notes based off of XML data provided to the game based on the song that is currently playing. When the song reaches a certain point determined by the XML data, the notes will be placed randomly on either the first or second “Note Spawner”. The note will then move down the corresponding track until the player hits the note, or the note reaches the bottom of the screen.

In order for the player to hit the note, the note must pass over the object called the “Note Catcher.” This object is located above the button which the player has to press and is a darker shade than the note that appears on screen. When the player presses the button located below the “Note Catcher,” the object changes color and if a note is over it, then the note gets destroyed, records a successful hit, and moves the kyokubou closer towards the bottom of the screen.

Once the kyokubou has reached the bottom of the screen, it will become more comfortable with the player and will be added to the player's collection of kyokubou friends. This will be shown with a separate panel that will be overlaid onto the screen that will indicate success (Figure X). From this overlay, the game will save to both the device and game server. After saving, an option to either go to the kyokubou collection screen or go back to the map screen will appear. Each kyokubou can only be captured once.

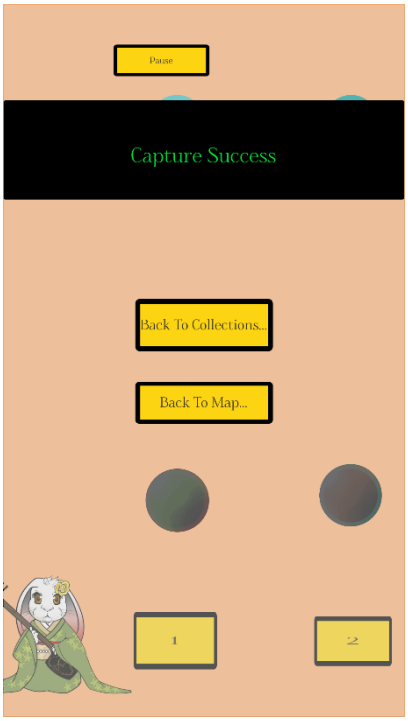


Figure 33: Successful Kyokubou Capture

If the player cannot successfully hit notes, the kyokubou will move towards the top of the screen, indicating that it is afraid of the player. When the kyokubou reaches the top of the screen, the kyokubou will be too afraid of the player, and the capture will be unsuccessful. From here, the player will only have the option to go back to the map screen (Figure X). In the current implementation of the game, you can try to capture the kyokubou right away. Our intention was to put a timed delay of when the kyokubou would appear again.

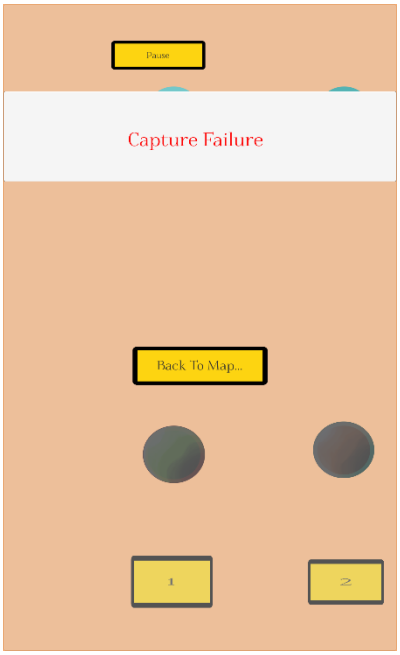


Figure 34: Unsuccessful Kyokubou Capture

5.5.1 Creation of Beat Maps

Since the game is centered on sound and music, we felt that the main gameplay mechanic also had to be related to music. While in Japan, the most popular games in the local arcade seemed to be around rhythm games. Games like *maimai* (2012) and *Taiko: Drum Master* (2004) seemed to capture attention and also have a distinct soundtrack. Unfortunately, these games require special instruments and would not be able to be played on a mobile device.

From here, we ended up looking at mobile rhythm games such as *LoveLive! School idol festival* (2013) and *Tap Tap Revenge* (2008). At the end of the design process for this game mechanic, we have decided on only using two separate note tracks, as opposed to three in *Tap Tap Revenge*. Another feature unique to *Step Symphony* is the use of custom music tracks and note placement, also known as beat maps.

The beat maps used in the game only keep track of the position where each note would be created at a certain time of the song. The only music based piece of data that could accurately keep track of where notes would appear in a song is MIDI data. As stated in a work by David Huber, "MIDI is a data protocol that communicates on/off triggering and a wide range of parameters to instruct an instrument or device to generate, reproduce, or control audio or production-related functions." (Huber, 2012)

These beat maps were created using Reaper Digital Audio Workstation and made use of the virtual keyboard tool. One track in the workstation contained the

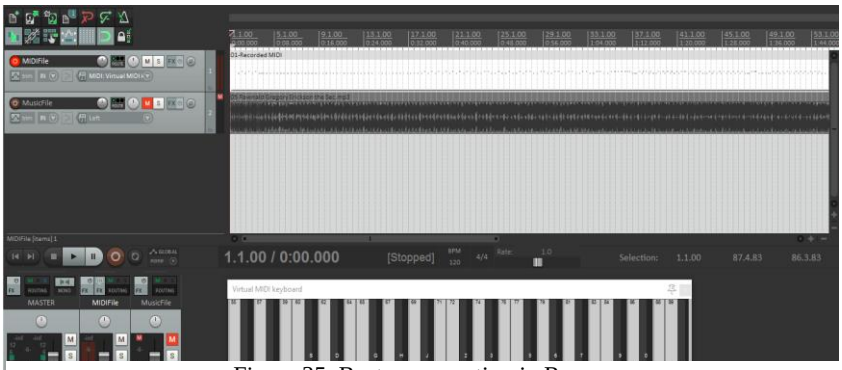


Figure 35: Beat map creation in Reaper

music file either in Wave File format or MP3 format, while another track would contain the MIDI data. The data was created when the record button is pressed and any note on the virtual keyboard was pressed during the duration of the music track. As seen in figure X, there are two tracks present in Reaper; the MIDI track and the audio track. The MIDI track is the first track displayed while the audio track is below the MIDI track.

Once the MIDI data has been created, it needs to be converted into a format that was understood by Unity and C#. Since Unity and C# are unable to interpret MIDI data but are able to quickly read in XML data, we had to find a way to convert MIDI data into XML data.

There is an open source code library called jMusic that is written in Java. This library contains functionality to interpret and analyze MIDI data. Given the toolset that the jMusic library has to offer, we created an application to take in a MIDI file and write the song duration, tempo, song title, song position and duration of each note to an XML file (Figure X). The application then saves the XML file to a place on disk that is specified by the user.

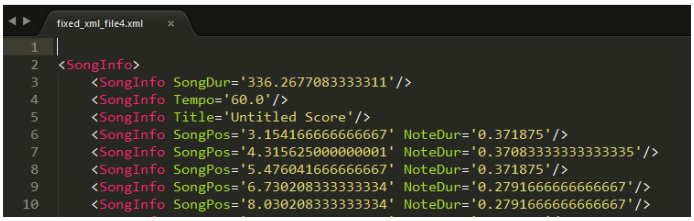


Figure 36: MIDI data interpreted into XML format

5.5.2 Creation of the Rhythm Based Mechanic

The rhythm based capture mechanic is broken down into multiple parts as depicted in figure X; the rhythm conductor, song information parsing, generating notes, activating note catcher and changing the position of the kyokubou.

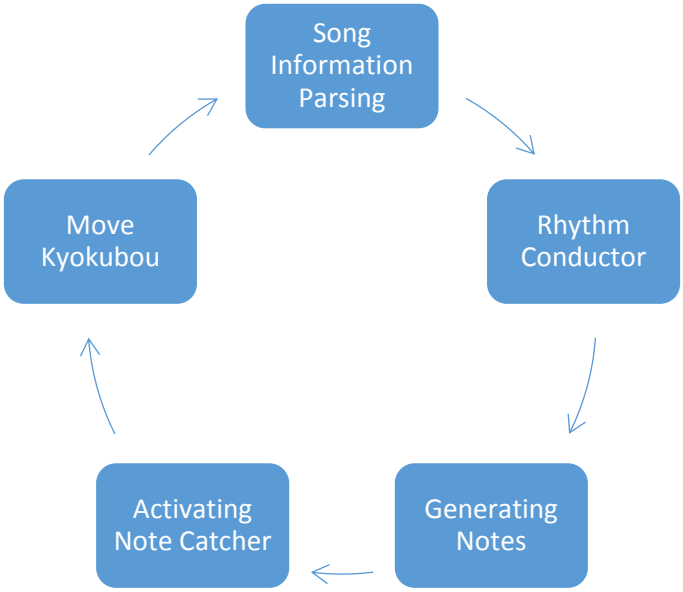


Figure 37: Cycle for Rhythm Based Capture Mechanic

Song Information Parsing

Using a library native to C# called Linq, we have the ability to take an XML file and parse it using the XDocument class. Using this class, we are able to parse through the entire XML document and separate data by attribute name. In figure X, the code below shows how the game iterates through the XML document based off of attribute names using a for each loop.

From here, the track speed of the song must be set given information from the XML file's tempo information. The track speed is set by dividing the bpm of the song and dividing it by the time duration of the beat in every measure of the song. This time duration can be adjusted depending on how difficult the song is intended to be.

If the attribute name is “SongPos”, it takes all of the song positions of the notes and adds them to a list of notes. From here, the information given from the XML data only tells the position where the note has to hit the note catcher.

```
//go through each XML node of song info to collect the note data for the file
foreach(var note in xmlFile.Root.Elements("SongInfo"))
{
    if(note.Attribute("SongDur") != null)
    {
        if (double.TryParse(note.Attribute("SongDur").Value, out songDuration))
        {
            if (conductorScript.durationInSamples != 0.0f)
                multValue = (conductorScript.durationInSamples/44100 - conductorScript.offset) / songDuration;
        }
    }
    if(note.Attribute("Tempo") != null)
    {
        if(double.TryParse(note.Attribute("Tempo").Value, out songTempo))
            trackSpeed = (60.0 / songTempo) / conductorScript.crotchetsPerBar;//set the track speed once it has been found
    }
    if(note.Attribute("Title") != null && note.Attribute("Title").Value.Length > 0)
        songTitle = note.Attribute("Title").Value;
    if(note.Attribute("SongPos") != null && note.Attribute("NoteDur") != null)
    {
        songInfo.Add(double.Parse(note.Attribute("SongPos").Value), double.Parse(note.Attribute("NoteDur").Value));
        songPositions.Add((float.Parse(note.Attribute("SongPos").Value) * multValue)-trackSpeed);
        noteDurations.Add(double.Parse(note.Attribute("NoteDur").Value));
    }
}
```

Figure 38: Parsing XML data in Unity C#

In order for the note to be spawned at the “Note Spawner” and end up at the “Note Catcher” is to first calculate the distance between the both of these objects and multiply it by the value that converts the MIDI song position into the Unity audio song position (the duration of song in samples divided by the sample rate of the music file). In order to find the conversion method, the duration of the song according to Unity must be found within the given sound file, subtract it from the offset of the song, and then divide it by the duration of the song according to the MIDI data, which is located within the XML file. For the scope of this game, we chose to make the sample rate for all of the songs equal to 44100 KHz for consistency and good audio quality. Once all of the values for the notes are converted properly, they are added to a list object which is then passed to the rhythm conductor.

Rhythm Conductor

The rhythm conductor is the master object within this mechanic, and controls most of how the music and notes are synched up without visual and auditory latency. This object keeps track of the beats per minute or BPM, time duration of a beat, position in time of the song, and the offset of the song.

The information about the variables is taken from the song object and is transferred into the rhythm conductor object. A visualization of how each element of the song is used while the game is playing is displayed in figure X. Once the song is received, the song position is calculated by dividing the number of time samples in the song by the sample rate of the music file. The importance of this game object is to ensure that the all other game objects that reference the song are only updated when the song position is changed. The rhythm conductor is the only game object that changes the song position and creates and moves the notes from the "Note Spawner" to the bottom of the screen. The information of the song is generated from another game object which reads in an XML file containing the data from the MIDI to XML parser.

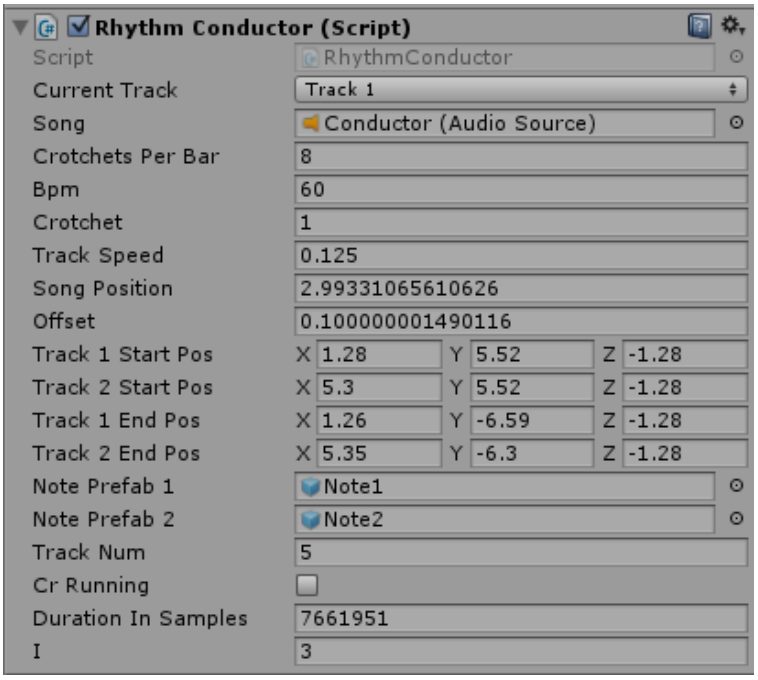


Figure 39: Rhythm Conductor Object in Unity

Generating Notes

Once the notes from the XML file are interpreted and put into a list of notes, the rhythm conductor now has the ability to play the song that is attached to the song information object and create new notes. Since the rhythm conductor is in charge of incrementing the song position, it will also check the list of notes and run a coroutine to randomly choose which track to place the note based on a random number generator. Once the note is created, another coroutine will run that will move the note from the "Note Spawner" to the bottom of the screen based on the calculated track speed.

Activating Note Catchers

Below the Note Catchers, there are yellow buttons that are represented by the track number. When one of these buttons is pressed, the color of the Note Catcher that was affected changes color, indicating activation. If a note is colliding and the Note Catcher becomes active, the colliding note will be destroyed and will not destroy other notes until the button has been released. When the note is destroyed, the game records this as a successful action and adds a point value to the current score.

Score System

There is no numerical way of keeping track of the total score in this part of the game. The way that score is kept is based off of the position of the kyokubou on the screen. When a successful hit is recorded, the kyokubou moves down the screen using a linear interpolation that decreases the y position. If the activation of the Note Catcher returns an unsuccessful press, the kyokubou moves towards the top of the screen using a linear interpolation that increases the y position. In figure X, there are box colliders which are invisible to the player that trigger the end of the game depending on which collider has been touched by the kyokubou.

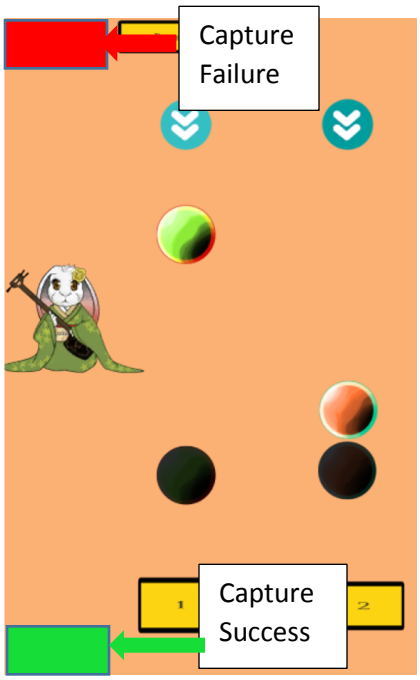


Figure 40: Ending positions for the kyokubou

Pause Button

The pause button allows the player to stop playing the game in the moment due to the inconsistent environment of the game world. When the pause button is pressed on the screen, the current song position that is remembered and the timescale of the game is set to zero. When the game is paused, another screen is drawn to the screen and will allow the player to continue to play the song or to go back to the map screen. When the player goes back to the map screen, the last kyokubou will still be in the same place and will not move again. If the player hits the resume button, the game will resume and the song will adjust to the last recorded song position and set the timescale back to one.

5.6 Databases

Step Symphony contains two databases that differentiate between game data and player information. Player account information and player progress are saved in the user database while location information and kyokubou information are stored within the kyokubou database. Both of these databases were created using a web framework called Django Rest Framework. This web framework would provide tools that help to create content from a web client and a game client. A large hurdle of this part of the development progress came from the integration between Unity and Django. Since Django is a web framework, there are many syntax discrepancies between a web based language and a game engine, such as the case of reserved headers.

In Django, there are certain commands that are placed at the head of the command that are called headers. These headers mean certain server commands based on the number. The main headers that are used during development are POST, PUT, DELETE, and GET. POST adds new information to the Django server. PUT replaces data that has already been written to the server. DELETE removes the data on the server that is contained within the response. GET writes a new response based on the information given from the send command to the server.

For development in Unity, we used the WWWForm class to send commands to the Django server. To send these commands from Unity, there are three parts which create a proper request to the server: the server url, additional data and the headers. The server url is combined with the server request command, which communicates with the Django server. The HTTP request that is created with the WWW object call in Unity can only be handled in a separate manner than usual HTTP requests. Using a delegate object called “onResponse”, the response status of the request is stored and can be retrieved from other classes within the code of *Step Symphony*. The responses that are recognized by the Django server are as follows: ClientError,

PageNotFound, RequestError, ParseError, and Success. Depending on what the response is, each function in the Unity project will activate other delegate methods that react to the Django server responses.

5.6.1 Player Database

The user database contains information about each player of the game and their progress. When the game is first opened, there is an option to either log into the server or sign up to be able to save your game to the servers. Interaction between the server and *Step Symphony* is represented by a singleton object called Player (Figures 18, 19). This game object allows for storage of data and communication between the game and the Django database.

Figure 41: Information about the current player in *Step Symphony*.

```
public class PlayerData : MonoBehaviour {
    public static PlayerData playerData;

    public string username;
    private int kyokuboCount;
    public List<Kyokubou> kyokuboCollection;
    private DateTime timeLastUpdated;
    SaveGame saveFile;
    public string saveFilePath;
    public string jsonData;
    public StepSymphonyBackend ssBackend;
```

Figure 42: Player Data singleton object in Unity C#

Signing up for *Step Symphony*

Figure 43: Sign up menu for *Step Symphony*

If the player presses the "Sign Up" button on the main menu, there will be a menu that prompts the player to input their email, username and create a secure password (Figure 19). To ensure that the player does not forget their password, there are two separate boxes to confirm that the password was entered correctly. On the upper left portion of the screen, there is a button to go back to the start screen, if the player already has an account and wants to log in instead.

Once the player has filled in all of the information correctly, they must press the button that says "Sign Up!" at the bottom of the screen. At this point, the scripts attached to the button will activate and check all of the information in the input boxes above. When the information entered is unique and has the correct format for each field, Unity will call a delegate function within the database scripts called SignUp. This function sends a POST command to the Django database with the information given. If the response from Unity to the Django server is successful, another function is activated which tells Unity that the command that was sent has been accepted and will then proceed to call the delegate function to log into *Step Symphony*. Once logging into the game returns a successful response, the map screen is loaded for the player. If any incorrect information is filled in one of the boxes, the sign up status message will change accordingly.

Logging into the Database

The first option that is given in the start screen is one to log into a previously made account for *Step Symphony*. The screen in figure 20 will prompt the player to enter the username and password that was made previously.

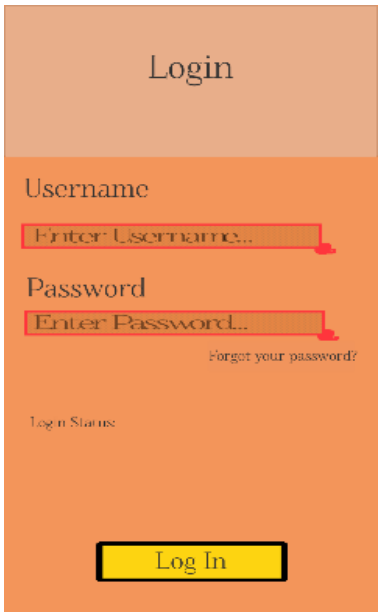


Figure 44: Log in menu for *Step Symphony*

When a valid username and password are entered into the boxes above and the "Log In" button on the bottom of the screen are pressed, Unity will call the delegate function in the scripts to log into the Django database. If the server's IP address is valid and matches the IP address that is called in Unity, a successful response will create an authentication token that will be a unique string for each user. Once the authentication token is created, the player is now able to send save data to the Django database and the map screen is loaded for the player. If incorrect information is given and the "Log In" button is pressed, the login status will change to show the reason for failure.

Saving the Game

When the player captures a Kyokubou, the player data object records the newly captured kyokubou in a list and saves the data in a JSON file that is written locally to the device. The locally written file is derived from the save data object seen in figure 22. Once the game has

saved locally, it is then sent as a POST request to the player database. If the response returned successfully, the latest local save data will overwrite the previous save data.

```
public SaveGame(string username, List<Kyokubou> collection, int count, DateTime lastUpdated)
{
    this.username = username;
    this.kyokuboCollection = collection;
    this.kyokuboCount = count;
    this.timeLastUpdated = lastUpdated;
}
```

Figure 45: Player save data in Unity C#

Player Database Model

This paragraph is about the structure of the player data within the Django database. Each user that has created an account in *Step Symphony* will store the user's credentials, a place on disk for the save data to be stored on the Django database. The way that the Django model is written is based off the variables that are within the Player object in the Unity project (Figures 18, 23).

```
models.py
13 class PlayerData(models.Model):
14
15     def update_savefile(instance, filename):
16         path = 'saves/'
17         format = '%s%s' % (instance.owner.pk, str(uuid.uuid4()))
18         return os.path.join(path, format)
19
20     owner = models.ForeignKey(User, null=True, blank=True)
21     saveData = models.FileField(upload_to=update_savefile)
22     create_date = models.DateTimeField(auto_now_add=True)
23     updated_date = models.DateTimeField(auto_now=True)
```

Figure 46: Player Django model in Python

5.6.2 Kyokubou Database

This database is connected to kyokubou information that is stored in the game. In Unity, there is a game object called the kyokubou data object. This is similar to the player data object

except, the only function of this object is to update the local information about kyokubou location information to the local game. The information is then saved within this persistent kyokubou data object in the JSON format (Figure 24).

Figure 47: Kyokubou Data object in Unity

Retrieving Kyokubou Information

Information about the kyokubou is stored on a separate database and contains a list of all of the collectable kyokubou that is present on the server. Each of the items in the database is written in the JSON file format and is passed into Unity as a game object of the class Kyokubou. This class is mirrored to the model of the kyokubou which contain information pertaining to the three locations that the kyokubou could be in as well as the name, instrument name and description of the kyokubou (Figure 25). Attached to this class is also the file path for the sprite, rhythm game track and collection screen sound. This information is recorded when the player logs into *Step Symphony* and can expanded upon in the future.

```
public Kyokubou(Location location_one = null, Location location_two = null,
                Location location_three = null, int id = 0, string kyoname = "",
                string instrument = "", string image = "", string music = "",
                string description = "", string reward = "")
{
    this.location_one = location_one;
    this.location_two = location_two;
    this.location_three = location_three;
    this.id = id;
    this.kyoname = kyoname;
    this.instrument = instrument;
    this.image = image;
    this.music = music;
    this.description = description;
    this.reward = reward;

}
```

Figure 48: Kyokubou constructor in Unity C#

Kyokubou Database Model

The structure of the kyokubou data within the Django database recognizes that each kyokubou has unique information that can be modified with ease through any web browser with administrative access. The information stored about the kyokubou is the same as the C# class in the Unity project, just in the form of a Django model.

```
class Kyokubou(models.Model):
    kyoname = models.CharField(max_length = 30)
    instrument = models.CharField(max_length = 50)
    image = models.CharField(max_length = 200, null = True, blank = True) #These will be just path references to the names of
    music = models.CharField(max_length = 200, null = True, blank = True) #This will be a path reference for the music file
    description = models.TextField()
    location_one = models.ForeignKey(KyoLocation, on_delete=models.CASCADE, related_name="Location1", null=True)
    location_two = models.ForeignKey(KyoLocation, on_delete=models.CASCADE, related_name="Location2", null=True)
    location_three = models.ForeignKey(KyoLocation, on_delete=models.CASCADE, related_name="Location3", null=True)
    reward = models.CharField(max_length = 200, null = True, blank = True) #This will just be a path reference for the name of
    create_date = models.DateTimeField('date_created')

    def __str__(self):
        return '%s %s %s %s %s %s %s %s %s %s' % (self.kyoname, self.instrument, self.image, self.music, self.description,
         self.location_one, self.location_two, self.location_three, self.reward, self.create_date)

    def was_created_recently(self):
        return self.create_date >= timezone.now() - datetime.timedelta(days=1)
```

Figure 49: Kyokubou Django Model in Python

As seen in figure 26, the information stored for each kyokubou can be wrapped as JSON data and become a Kyokubou game object in Unity with the unique fields as they appeared in the Django database.

There are is also a separate model class for the kyokubou locations in Django (Figure 27) that also mirror the kyokubou location class in the Unity project (Figure 28). This model keeps track of the location name, latitude and longitude of the location to the fifth decimal place and the date on which the location has been entered in the database.

```python
class KyoLocation(models.Model):
    locname = models.CharField(max_length = 50)
    latitude = models.DecimalField(max_digits = 8, decimal_places = 5)
    longitude = models.DecimalField(max_digits = 8, decimal_places = 5)
    create_date = models.DateTimeField('date_created')
    def __str__(self):
        return '%s %s %s' % (self.locname, self.latitude, self.longitude)

    def was_created_recently(self):
        return self.create_date >= timezone.now() - datetime.timedelta(days=1)
```

Figure 50: Kyokubou Location Django Model written in Python

```
[Serializable]
public class Location {
    public string locname;
    public double latitude, longitude;

    public Location(string locname = "", double latitude = 0.0, double longitude = 0.0)
    {
        if(locname != "")
            this.locname = locname;
        this.latitude = latitude;
        this.longitude = longitude;
    }
}
```

Figure 51: Kyokubou Location class written in Unity C#

If the game expands to locations greater than central Kyoto, new locations can be entered into the database and integrate with *Step Symphony* with relative ease. Overall, the integration of databases within the game are integral to future expansion and are aimed to keep away from third party use to store user information.

5.7 Collection Screen

Once the player has successfully befriended the kyokubou, the information about this player will be updated in the player database, allowing access to the kyokubou's picture, description, and payable note from that kyokubou object in the kyokubou database. Then, that kyokubou will also be added to the player's collection screen. The player will be able to access the collection screen from the menu, where they can see all of the

Figure 52: Collection Screen with one of the kyokubou pieces pulled out. The player would see a description here.

kyokubou they have befriended over the course of the game. The collection screen looks like a radial menu, with eight kyokubou around the outer sections of the circle, and a button in the middle that allows the player to switch what instrument the player is looking at.

On this screen, the player is able to tap the kyokubou buttons to play notes associated with the kyokubou's instrument. Each kyokubou has a unique sound, and the player can also tap multiple buttons at once to have more notes playing at the same time. The player can also swipe away from the circle to bring up the descriptions of the kyokubou they've found, allowing them to see information about the kyokubou, their relationship with other kyokubou in the game, and where they could find other kyokubou in the area around them.

We made the design decision to give the kyokubou descriptions because we felt that giving the kyokubou backstories would make them feel more realistic. We also felt that it was easy to see people getting lost and being unable to find other kyokubou. Putting hints as to where to find other kyokubou in their descriptions was the solution we decided to pursue. The musical aspect of the collection screen allows the player to have a degree of freedom with how they interact with the kyokubou they've found, and allows them to express creativity with the sounds that are native to Japanese culture.

6 Testing Methods for *Step Symphony*

Step Symphony's gameplay and environment required a slightly altered method of evaluation and testing. According to Desurvire (2009) and Wiberg (2009), existing game evaluation methods depend heavily on a set of events in a controlled environment that are easy to calculate and quantify, and must be broad enough to fit all genres of games. However, the events in *Step Symphony* as a whole exist solely in an environment that is unable to be controlled, the world around us. Because of this factor, aspects of *Step Symphony's* gameplay require that they be playable in an uncontrolled environment, often filled with numerous distractions.

In order to test *Step Symphony*, our evaluation method must be quantifiable and concise, as well as be able to test playability in both controlled and uncontrolled environments. It must also determine if the gameplay of *Step Symphony* in an uncontrolled environment is engaging, easy to follow, and if users would continue to choose to play *Step Symphony*. In order to fulfill these requirements, we made two separate playtests in order the properly test *Step Symphony*. One will take place in a controlled environment, and another will take place in an uncontrolled environment.

6.1 Challenges of Testing

Because our MQP took place at Ritsumeikan University in Kyoto, Japan, initial testing for *Step Symphony* also took place in Ritsumeikan University BKC. While we may have had clear cut goals for our tests with quantifiable measurements, because half of our tests will take place in an uncontrolled environment, there will always still be a possibility of error. However, this is crucial to refining *Step Symphony*, as there will always be that same possibility while

playing the game itself. We will also be testing a component of *Step Symphony* that has not been commonly used before, directional audio.

Another difficulty that we faced while testing, because we were working in a foreign country, was the present language barrier. Currently, *Step Symphony* is only available in one language, English. However, all of the preliminary testing will be held in Japan. Our target audience for *Step Symphony* was young adults visiting Kyoto to sightsee. This included tourists and residents alike. However, since we had no tourists to playtest *Step Symphony* at the time/location of testing, our playtesters will solely consist of residents/students of Kyoto between the ages of twenty and thirty.

Over the last twenty-five years, the Ministry of Education had attempted many reforms to change and improve English education in Japan. This included bring native English speaking Assistant Language Teachers, or ALTs, into classrooms, and putting a stronger emphasis on communication rather than grammar-translation (Matsura, 68). With these reforms, our team originally thought that more students would be able to speak and read basic English. However, through our time working in the lab, we learned that most students working there as well knew little to no English, and communication at times was extremely difficult.

A recent study was held at 480 high schools across Japan last summer. During this study, students were given an English written and oral exam. The results were lower than expected, with over 29 percent of students received a score of 0 on the written portion, and over 13 percent received a score of 0 on the oral portion. In fact, 80 percent of the students did not even attempt the speaking portion at all. These results fell short of the government's wish of having at least 50 percent of high school graduates scoring at Eiken Grade 2 (TheJapanTimes, 2015). More recently, the Ministry of Education released data recently showing that have classes taught solely

in English and have students start taking English lessons in elementary school have not produced any significant results. The data also shows that junior high and high school students in Japan failed to meet the government's target in English reading, writing, listening, and speaking (TheJapanTimes, 2016)

So, since little to none of our playtesters would be able to speak or read the English language, and our team spoke little to no Japanese, we needed to find another way to communicate with our playtesters so there would be no confusion during playtesting.

6.1.1 Working Around the Language Barrier

Because *Step Symphony* is currently only in English, a tutorial would not be equally helpful for all of our testers. Because of this, we chose not to implement a tutorial at the time of testing. We believed that, since our planned tutorial would have a mascot explaining what to do to capture a kyokubou in English, implementing said tutorial at this time would give skewed results, or not be useful in explaining the controls at all.

At first, we had wanted to write up both the surveys and test instructions in two languages, English and Japanese. Unfortunately, we did not have the time nor the resources to do this. Instead, we used simple English in the pre and post test surveys. Playtesters were able to answer written responses in either English or Japanese. For the instructions for each test, we had two versions, a written version and a visual version. The written instructions were written in simple English. The picture instructions included hand-drawn images of what the written instructions attempted to convey. We also gave small visual demonstrations on how to play

certain portions of *Step Symphony*. If the tester had any questions about wording or translations for either the surveys or the instructions, we had a professor at each test to translate.

6.2 Testing Directional Audio

Before looking at directional audio, we must look at spatial audio. For *Step Symphony*, we will be using He's (2016) definition of spatial audio. According to He, spatial audio, or three-dimensional audio, refers to the perception of sound in 3D space. He also states that humans have the ability to localize sound, or find the distance, direction, and elevation of an audio source, using both ears (He, 7). In *Step Symphony*, players will only be required to find the direction and distance of directional audio on the horizontal plane. According to He, the horizontal plane "refers to the plane that is horizontal to the ground at ear-level height" (He, 8).

When a player first enters a kyokubou's zone, the directional sound playing in their headphones will be quiet. Because of this, we will also have to take into account auditory masking. Auditory masking, according to He, is when a louder sound partly or fully masks a weaker sound (He, 14). Since *Step Symphony* takes place in the world around us, there will be outside noises that may mask directional audio when the player is farther away from the kyokubou.

Since spatial audio is usually only tested through subjective listening tests, the core component we will focus on during testing will be directional audio, and whether outside sound will affect gameplay. In order to not skew results, we will not be using noise-canceling headphones during testing.

6.3 Two Types of Tests

For *Step Symphony*, we ran two different tests, similar to an A/B test. The first test will take place in a controlled environment, while the second test will take place in an uncontrolled environment. These tests will run similarly to a psychological experiment, with one test as the control group, the controlled environment test, and one test as the treatment group, the uncontrolled environment test.

According to Robinson (2015), it is advisable to separate an A/B test into segments. This proved extremely successful for *Step Symphony's* test, as it was easier to compare results because we were able to separate *Step Symphony's* gameplay into our core components. We ran these tests with the following research questions in mind:

- Can playtesters correctly identify the source of the directional audio? (usability)

Since directional audio is the first step to being able to locate and capture kyokubou, it is crucial that playtesters are able to identify the correct direction. If a player is unable to correctly place the direction that the audio is originating from, then they will not be able to play the rest of *Step Symphony.* According to Dr. Nacke, a PhD in Digital Game Development and Affective Computing, usability has concrete, measurable aspects, and can be expressed in numbers. For *Step Symphony*, we will be able to measure the usability of directional audio by the length of time it takes to identify the direction the audio is coming from.

- Can playtesters easily learn to use the three core components? (learnability)

In *Step Symphony*, the three core components include directional audio, augmented reality, and the rhythm game capture mechanic. For this question, we will be looking at learnability for each component. According to Tullis and Albert (2013), learnability is the ability

to learn how to play a game or use a product over time, while gaining experience on how to use it. This learning, usually by doing the same task repeatedly, can be measured by looking at how much time is required to become proficient in the task. For this test, learnability can be measured by looking at the difference in time that it takes to complete the same task.

- Will using the three core components in an uncontrolled environment be more difficult? (distractions)

By testing in a similar fashion an A/B test format, we can compare the results of looking at usability of directional audio, as well as learnability of the three core components of *Step Symphony*, from both types of tests. We also look at the number of times the playtester was distracted by an outside source.

6.3.1 Testing in a Controlled Environment

Testing in a controlled environment, or the control group, focuses on the usability and learnability of each core component of *Step Symphony* with no possible outside distractions. Here, playtesters were given instructions on how to use each portion of the game separately. They were asked to complete each task a certain number of times. Testing for this section took place in a separate room. Each tester was given an Android smartphone loaded with the test, as well as a pair of over-the-ear headphones.

6.3.2 Testing in an Uncontrolled Environment

Testing in an uncontrolled environment, or the treatment group, focuses on the usability and learnability of each core component of *Step Symphony* with the possibility of outside distractions. Here, playtesters were given instructions on how to use each portion of the game in

sequential order to successfully collect a kyokubou. They were asked to capture two kyokubou located on campus at Ritsumeikan University BKC, where testing took place. Each tester was given an Android smartphone loaded with the test, as well as a pair of over-the-ear headphones.

6.4 Controlled Environment Testing

The first round of testing should consist of controlled environment testing, or the control group. This test will measure the usability and learnability of each component of *Step Symphony* separately.

For these tests, please keep in mind that the scripts we used during our testing were short, due to the language barrier. Our scripts explaining the test were also accompanied with hand-drawn images, as well as physical demonstrations. Since there are three core components of *Step Symphony*, there are three portions for the controlled environment test.

6.4.1 Preparation

The *Step Symphony* controlled playtest should take place in a separate suite. The suite should have at least one table and two chairs. Only one test administrator is needed to conduct this playtest.

Prior to the session, the test administrator should setup the testing area, as shown in Figure 26. This should all be done before the first tester arrives. Ideally, the testing area should remain consistent throughout all playtests. However, as long as the setup remains the same, it may be moved to a new location if necessary.

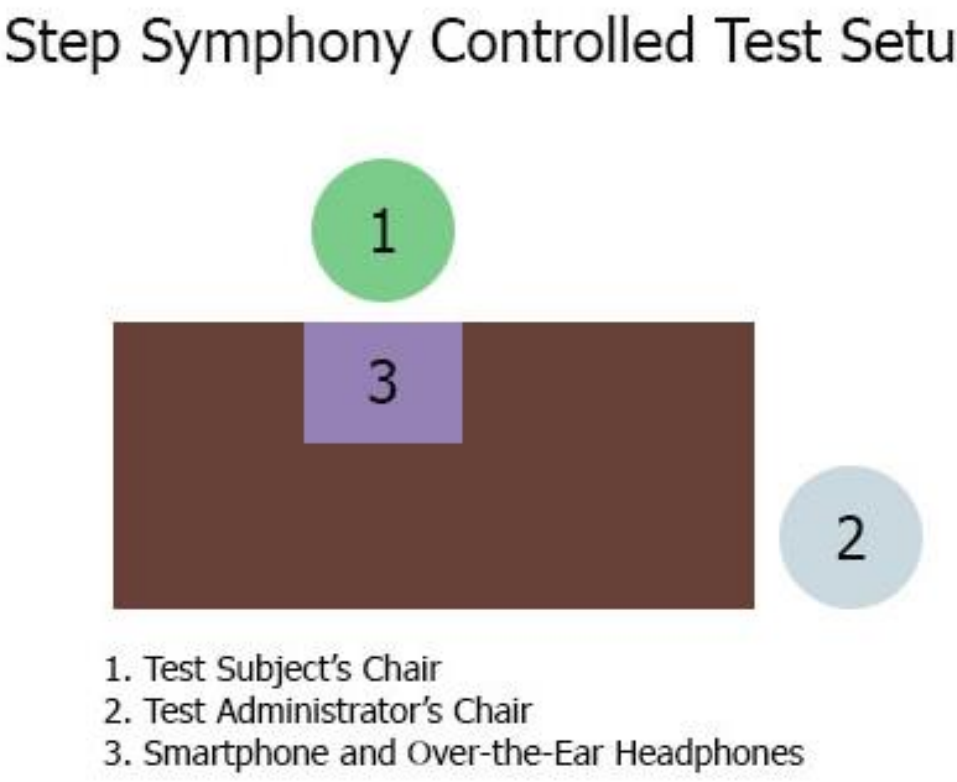


Figure 53: *Step Symphony* Controlled Test Setup

The test administrator must install the controlled test demo onto the smartphone before any testing begins. Once all of these steps are completed and the testing area is set up accordingly, wait until the first test subject arrives.

6.4.2 Introduction and Pre-test

When the test subject arrives, greet them at the door and direct them to the designated chair (1). When they are seated and ready, read them the following introduction.

Hello, and welcome to the *Step Symphony* playtest. My name is (insert name here) and I will be running your playtest today. Thank you for taking time out of your day to help us out.

Before we begin, I would like to ask you to fill out the following survey. Please write your answers in either English or Japanese. If you have any questions, please ask me.

At this time, hand the test subject a pen and the pre-test survey (Appendix F). According to Tom Miegs (2003), profiling playtesters is an excellent way to categorize and prioritize player feedback (pg. 179). Without these measures, it would be more difficult to prioritize the feedback to act on for *Step Symphony*.

The questions on the pre-test survey are designed to get a more detailed profile of the tester, to more accurately understand and prioritize their later feedback. One example of this would be whether a playtester plays mobile games or not. If a playtester is not used to playing mobile games, their feedback will be treated differently than a playtester who plays mobile games regularly.

When the playtester has finished filling out their survey, proceed to the second section of the controlled test.

6.4.3 Playtest

When the tester has finished with the survey, read the following while showing them the visual instructions.

Thank you very much. I will go over the playtest now. There are three parts to this playtest, which I will go over one at a time. If at any time you are confused or have a question, let me know.

Here we have visual instructions (Appendix E). I will go over part 1 now.

The goal of the first part of the test is to find the directional sound. During this part, you will hear sound in your headphones. Sometimes, the sound will only be in one ear. When you hear the sound on the left side, turn left. When you hear the sound on the right side, turn right. When you hear the sound in both ears, then the sound is coming from in front of you. Please point in that direction, tell me, then press the button on the center of the screen. We will do this three times.

Now, sit back and observe the playtester as they complete the first portion. Make sure to time how long it takes to find each sound, whether or not they identify the correct direction, and if they showed any confusion with the controls. The length of time it takes to identify the direction of each sound are used to measure learnability to answer the second research question. Whether the playtester could identify the correction direction, or if they were confused with the controls, are used to measure usability to answer the second research question.

When the playtester has pointed out the direction for the third music track, it is time to move on to the second portion. Read them the following while showing them the visual instructions.

Thank you very much, we will now move on to the second portion of the test. I will go over it now.

The goal of the second test is the find the kyokubou using the phone's camera. You can move the phone up, down, left, right, any way you wish. There are three kyokubou around you. When you find one, please tell me what animal it is, and then press it with your finger. Please find three kyokubou. When you have found all three, a button will appear on screen.

Now, sit back and observe the player complete the second portion. Make sure to time how long it takes to find each kyokubou, and whether or not they showed any confusion with the controls. Both of these observations are used to measure learnability to answer the second research question.

When the playtester has found the third kyokubou and the button has appeared on screen, it is time to move on to the third portion of the test. Please read them the following while showing them the visual instructions.

Thank you very much, we will now move on to the third portion of the test. I will go over it now.

The goal of the third test is to play a rhythm game. Please let me know when you have finished it. When you are ready, please press the button on screen.

Now, sit back and observe the player complete the final portion of the test. Make sure to time how long it takes to complete, whether they successfully completed capturing the kyokubou or not, if there was any confusion, and whether they completed the game before the song ended.

When the playtester completes the third portion of the test, proceed to the post-test.

6.4.4 Post-test

At the beginning of the post-test, read the following:

> Thank you very much. Before you go, I have one last survey for you to fill out. You can write your answers in either English or Japanese. Also, if you have any questions about the survey, or don't understand a question, please let me know.

Now, hand the test subject the *Step Symphony* post-test survey (Appendix G) and a pencil. Wait patiently until they have finished with the sheet.

The post-test survey contains 7 statements that playtester will rank with a number between 1 and 5. A rank of 1 means that the player strongly disagrees with the statement, and a rank of 5 means that the player strongly agrees with the statement. These statements are as follows:

Directional Sound (Part 1)

It was easy to follow the directional sound.

Directional audio has not been used to this extent for a mobile application before. Therefore, there is little to no existing documentation on the effectiveness of using sound in this format for mobile applications. This statement is used to identify the playtester's opinion on the ease-of-use of directional audio. If playtesters found this mechanic to be too difficult to follow, we would have to look into revising it or finding a new mechanic to find kyokubou.

I am interested in this as a game mechanic.

Since directional audio is not a commonly used game mechanic, we were interested on the playtester's view of it. Directional sound is the main mechanic that players will be spending

most of their time using while playing *Step Symphony*, so we wanted to make sure that it was welcomed by playtesters.

Finding the Kyokubou (Part 2)

It was easy to find the Kyokubou.

This statement is used to identify the playtester's opinion on the ease-of-use of the augmented reality potion of *Step Symphony*. If playtesters found this mechanic to be too difficult to use, we would have to look into revising it or finding a new mechanic.

I enjoyed finding the Kyokubou.

According to Federoff (2002), usability can also partially be measured by looking at the playtester's attitude, or satisfaction. If something is more satisfying to use or complete, it is often easier to learn. This statement was added to gain insight on the playtester's attitude towards the augmented reality portion of *Step Symphony*.

Capturing the Kyokubou (Part 3)

The controls were straightforward and easy to use.

This is based on Rehak's (2003) claim that difficult to use UI forces the player out of their immersion and back into the real world. In *Step Symphony*, this is the part that can be considered a 'twitch' component, or one that the player cannot easily look away from.

The game held my focus.

This statement was added to tie in with the statement above.

Capturing the Kyokubou was fun.

Again, if something is more satisfying to use or complete, it is often easier to learn. Similar to the statement for augmented reality, this statement was added to gain insight on the playtester's attitude towards the rhythm game capture mechanic.

Once the test subject has completed the post-test survey, please take the survey from them and say the following.

Thank you very much for your time. The test is now over.

The controlled environment test is now over. Lead the playtester out of the testing area. Once they are gone, clean up the testing area for the next playtester.

6.5 Uncontrolled Environment Testing

The second round of testing should consist of uncontrolled environment testing, or the testing group. This test will take place outdoors and will measure the usability and learnability of *Step Symphony* as a whole.

Again, for these tests, please keep in mind that the scripts we used during our testing were short, due to the language barrier. Also, our scripts explaining the test were also accompanied with hand-drawn images, as well as physical demonstrations.

6.5.1 Preparation

The *Step Symphony* uncontrolled playtest sessions should be conducted on Ritsumeikan University BKC. For the playtester, you will need a pencil, a clipboard with the pre and post test surveys, a smartphone with the uncontrolled environment test pre-loaded, and over-the-ear headphones. Two test administrators will be needed to run this test, one to record observations and one to explain and move the test along.

Because this test takes place in an uncontrolled environment, no setup is required, as any setup would skew the results of the gathered data.

The start of the test will take place outside the front doors of the Creation Core building. There, wait patiently for the playtester to arrive.

6.5.2 Introduction and Pre-test

When the test subject arrives, please greet them at the door and bring them to the sitting area on the side. Setup there is not needed as they won't be there long. Once they are settled, read the following.

> Hello, and welcome to the *Step Symphony* playtest. My name is (insert name here), and this is (insert name here). We will be running your playtest today. We thank you for taking time out of your day to help us out.
>
> Before we begin, I would like to ask you to fill out the following survey. Please write your answers in either English or Japanese. If you have any questions, please feel free to ask me.

At this time, hand the test subject a pen and the pre-test survey (Appendix C). This pre-test survey is exactly the same as the pre-test survey for the Controlled Environment Test. When the playtester has finished filling out their survey, proceed to the second section of the uncontrolled test.

6.5.3 Playtest

When the tester has finished with the survey, read the following while showing them the visual instructions.

Thank you very much. I will go over the playtest now. This test will take place outside on campus. During this test, you will attempt to capture two kyokubou. Here we have visual instructions on how to catch a kyokubou.

First, you will have to get close to kyokubou. To do this, you follow the directional sound. During this part, you will hear sound in your headphones. Sometimes, the sound will only be in one ear. When you hear the sound on the left side, turn left. When you hear the sound on the right side, turn right. When you hear the sound in both ears, then the sound is coming from in front of you, and you should walk in that direction.

As you get closer to the kyokubou, the music will get louder. When you get close enough to the kyokubou, a button will appear on screen. Do you have any questions so far?

At this point, you should pause for any questions, or translations if needed. When there are no questions, continue by reading the following while showing them the visual instructions.

Now, when you press the button on the center of the screen, it will open the phone's camera. You must use the camera to find the kyokubou. You can move the phone up, down, left, right, any way you wish. When you find the kyokubou, press on it. This will bring you to a short rhythm game. Successfully completing the rhythm game will capture the kyokubou.

You will need to do all of this twice to capture two kyokubou. Do you have any questions?

At this point, you should pause for any questions, or translations if needed. When there are no questions, bring the playtester to the front left entrance. Here, hand them the headphones to put on, and start the test on the smartphone. As the test is loading, hand the playtester the smartphone.

Now, one test administrator should walk by the playtester to check their progress, and the other should follow from a distance to take notes. The second test administrator should take note of the number of time the playtester gets distracted from outside sources, the number of hints needed to catch the kyokubou, and the time needed to complete each section.

Recording the number of distractions from the outside environment makes easier to review whether the uncontrolled environment had any effect on gameplay, or the third research question. By recording the amount of time it takes to complete each core component to collect a kyokubou, we can compare those times to the ones collected during the controlled environment test, or the control group, again the third research question. This can also be used to review whether the uncontrolled environment had any effect on gameplay as a whole, as well as learnability and usability for capturing a kyokubou, the first and second research questions. The number of hints required to complete a section of *Step Symphony* can be used to measure learnability to help answer the second research question.

If the playtester has been following the directional sound for ten minutes and has not found the kyokubou, have the first test administrator stop them and bring them to the location of the kyokubou. The first kyokubou is located south-west of the building entrance, near the benches. The second kyokubou is located west of the fountain on the other side of campus.

When the playtester has successfully found and caught the first kyokubou, read them the following.

> Well done, you found the first kyokubou. Now, the second kyokubou is located in the direction of the fountain. Please keep your headphones on while we walk that way. On the way there, you'll start to hear music playing again. When that happens, please attempt to catch the second kyokubou.

Now, walk the playtester in the direction of the fountain on the other side of campus, and have the second test administrator follow from a distance and should keep track of the same information as noted previously. Again, if the playtester has not found the kyokubou and ten minutes have passed, have the first test administrator lead them to that area.

Once the player has found and attempted to catch both kyokubou, take the headphones and smartphone from them. Then, bring them back to the Creation Core sitting area where they took the pre-test survey. When there, proceed to the post-test.

6.5.4 Post-test

At the beginning of the post-test, read the following:

> Thank you very much. Before you go, I have one last survey for you to fill out. You can write your answers in either English or Japanese. Also, if you have any questions about the survey, or don't understand a question, please let me know.

Now, hand the test subject the *Step Symphony* uncontrolled post-test survey (Appendix G) and a pencil. Wait patiently until they have finished with the sheet.

The post-test survey contains 6 statements that playtester will rank with a number between 1 and 5. A rank of 1 means that the player strongly disagrees with the statement, and a rank of 5 means that the player strongly agrees with the statement. These statements are as follows:

The game controls and interface were easy to use.

This is based on Rehak's (2003) claim that difficult to use UI forces the player out of their immersion and back into the real world. During the uncontrolled environment test, we want make sure that the player is not distracted by anything in the world around him or her.

The game held my focus.

This statement was added to tie in with the statement above.

I kept getting easily distracted by other things while playing.

This statement was added to tie in with the statement above.

I had no trouble following the music to find Kyokubou.

Again, directional audio has not been used to this extent for a mobile application before. Therefore, there is little to no existing documentation on the effectiveness of using sound in this format for mobile applications. This statement is used to identify the playtester's opinion on the ease-of-use of directional audio. If playtesters found this mechanic to be too difficult to follow, we would have to look into revising it or finding a new mechanic to find kyokubou.

I'm interested in seeing all of the other available Kyokubou.

This statement was added to see the player would be interested in continuing to play this game.

I would recommend *Step Symphony* to a friend.

This statement was added to tie in with the statement above.

Once the test subject has completed the post-test survey, please take the survey from them and say the following.

> Thank you very much for your time. The test is now over.

The uncontrolled environment test is now over. Lead the playtester out of the sitting area. Once they are gone, head to the main entrance to wait for the next playtester.

6.6 Revisions Based on Results

Due to a small sample size, 7 – 8 for each test, as well as the language barrier posing some hindrances, additional future playtesting will be needed. However, we can draw a few possible conclusions from the limited testing for revisions to *Step Symphony*.

Unfortunately, during the uncontrolled environment testing period, we ran into a game-breaking error during the directional sound portion of the test. This error took place when the playtester was tasked with collecting the second kyokubou, and may have skewed some of our conclusions from that portion of the test, so future testing in other locations will be required. This error will be explained later on.

6.6.1 Revision of Snapshot Mode

During playtesting, both controlled and uncontrolled, the playtester would have to point the phone down to the floor in order to find the kyokubou. Originally, this was done to show that kyokubou were very small, so the player would have to look towards the floor in order to find them. However, it was found that during testing, for both the control group and treatment group, that playtesters tended to look at eye-level and above to find the kyokubou. Because of the observation that all playtesters would look for the kyokubou with the phone's camera at eye-level and above, we decided to instead have the kyokubou spawn around eye-level instead of on the ground.

6.6.2 Other Forms of Feedback for Players

During controlled testing, most playtesters from the control group were able to correctly identify all three directions during the directional sound portion of the test. Also, during the post-test survey, 6 out of 7 agreed or strongly agreed with the statement that "it was easy to follow the directional sound." Test administrators also observed that the playtesters were very surprised by the directional audio, excited to use it, and enjoyed the custom audio tracks in the test.

During uncontrolled testing, it appeared that the opposite was the case. Test administrators observed that playtesters constantly second-guessed themselves when finding the second kyokubou. Most playtesters had to be led to the source after the ten-minute mark, and all playtesters disagreed or strongly disagreed with the statement in the post-test survey that they "had no trouble following the music to find kyokubou." However, this difficulty was only seen

when finding the second kyokubou, and was later credited to a game-breaking error that flipped the panning for the directional audio.

Even though the difficulties the playtesters faced were likely due to a game-breaking error, we would like to run another A/B test in the future. The control group for this test would have the same directional audio and player map in a new location, while the treatment group would have more player feedback on the map. For example, if a player in the treatment group is heading in the correct direction, the player icon on the map would blink, or the phone would vibrate. Even with these new forms of feedback, we would still like to have directional audio used in *Step Symphony* to locate kyokubou, as the audio itself allows the player to explore the real world by following the music the kyokubou play, which is an experience goal for *Step Symphony*.

6.6.3 Testing Error

During the uncontrolled test, there were two paths the player could take to reach the second kyokubou. If the player took the left path, everything was fine and the directional sound worked correctly. This path was taken when we tested our uncontrolled environment test before actual testing, and by our first playtester. That was the only test where the playtester was not confused by or did not require additional assistance with the directional sound portion. If the playtester took the right path, then the directional sound stopped working correctly. The directional sound would lead the playtester in the opposite direction of the kyokubou, so the playtester would think they were going in the correct direction. However, the volume would get quieter, signifying that they were going in the wrong direction. Besides the first playtester, everyone else took the path on the right.

On further inspection, it was not *Step Symphony* that crashed, but rather the smartphone. When tested, all smartphones in this area did the same. Directional audio works using a smartphone's magnetic heading reading. It is through this reading that the phone knows which cardinal direction it is facing. When there is something in the area that makes its' own magnetic pull strong enough to mess up that reading, then the directional audio no longer functions correctly. We believe that something in the area caused the magnetic reading in each smartphone to malfunction.

Currently, because this is a location-based hardware problem, we have no way of overcoming this error. We also believe that this hardware error skewed our results from our uncontrolled test. To rectify this, we will have to do further testing in different locations to fully identify the usability of directional sound.

6.7 Discussion of Results

While additional research is definitely needed, *Step Symphony's* testing method seems to be successful to provide the foundation of a good playtesting rubric for testing in uncontrolled environments. Minus the testing error that took place in the treatment group, each section of the test performed as we expected them to. However, since we only had eight playtesters for the controlled environment test, and seven playtesters for the uncontrolled environment test, we did not have a large enough sample size to fully prove the success of this playtesting method.

Even with our small sample size, testing error, and language barrier, we were able to find some successful points to this playtesting method.

6.7.1 Uncontrolled Environment did not Cause Distractions

During the uncontrolled environment test, playtesters wore over-the-ear headphones for the entire test itself. However, these headphones were not noise-cancelling, so we were unsure as to how the outside world would affect overall gameplay. Fortunately, the results were better than we anticipated. Out of the seven playtesters in the treatment group, none of them were distracted by outside sources while playing any portion of *Step Symphony*.

This may not always be the case for all mobile applications using the same test. For *Step Symphony*, playtesters reported not being distracted because the headphones allowed them to focus solely on the game. However, in order to confirm or deny this claim, further playtesting using portions of *Step Symphony*, or another mobile application, without headphones will be required.

6.7.2 Confusion Between Pauses or Dead Zones

Another point we observed during this playtest was the confusion between kyokubou locations. For the uncontrolled test, there was a pause where the playtester would have to walk across campus to find the second kyokubou. During this pause, before he or she got within 500 feet of the second kyokubou, there was no music playing in the headphones. During this time, most playtesters thought that the demo had stopped working correctly, and would tell the test administrators. Even after assurances that the demo was functioning correctly at that point in time, most playtesters were still concerned. It was only after we entered the zone for the second kyokubou, and the directional audio started playing again, that playtesters stopped worrying.

Unfortunately, we did not have the time to fix this point so late in the game development stage. For mobile applications in uncontrollable environments, especially those that use augmented reality like *Step Symphony* (player map and Snapshot mode), testers agreed that something is needed on screen to keep them occupied. For *Step Symphony,* it wasn't enough to have just the map of the area. A few playtesters asked for more notifications to appear on the map screen, just so they would know that the game was still running.

6.7.3 The Importance of a Control Group in a Controlled Environment

The final point we noticed in testing *Step Symphony* was the importance of a control group in a controlled environment, and how finding results from our playtests would have been impossible without it. The purpose of the control group was to make sure playtesters could understand and use core components of *Step Symphony*. If a playtester could not understand how to play portions of the game, then that playtester would not be able to play *Step Symphony* as a whole in the uncontrolled environment.

The purpose of a controlled environment test, in this situation, is to make sure playtesters can easily follow portions of the games they are testing. If a core component is confusing, fix it and do more controlled environment playtesting. Uncontrolled environment playtesting should only happen after most, if not all, core components in the game being tested are understandable and usable. Then, test administrators should begin uncontrolled environment testing, and compare results to the final controlled environment tests.

7 Post-Mortem

At the end of development of *Step Symphony*, we sat together as a group to discuss the overall results of our development process. We talked about what we learned as a group, as well as what we learned separately. We moved on to what went right during the development phase, as well as what went wrong. Lastly, we talked about what we would have changed about our development process looking back on it now.

7.1 What we Learned

Since we were abroad at the Japan Project Center, we had around 12 weeks to design and implement our game. During this timeframe, our team of three had to work fast, and we had to get things done right. When something didn't work as intended, it was usually fixed by the next work day.

In order to keep on schedule to finish our application in time, we learned that we had to set goals and deadlines for everything. From concept art for kyokubou to adding information to our database, everything had a deadline on our shared calendar. We also had group meetings set at the end of each day. These meetings were to go over the game's design, the portions of the game due that day, what everyone had completed that work day, and if anyone had any questions.

Out of the three of use, we had never worked as a team before. To overcome this, we had a meeting during PQP. During this meeting, we all discussed our strengths and weaknesses. We also went over what we wanted to get out of our MQP, as well as how we usually worked best. We continued these meetings weekly during MQP as well.

For *Step Symphony*, we used Unity as well as Python and Django for databases. Out of our team of three, one of us had never worked in Unity before, none of us had ever created a database before, let alone using Django, and we had all only coded in Python once during our freshman year. Two of us had never used Source Tree either, only using GitHub in the past. All of these things, we had to learn how to use in the first week.

During our stay in Kyoto, we were able to view cultural and historic sites such as Fushimi Inari and Kiyomizu-dera. By visiting certain sites, we were also able to learn about their history, popular internal attractions, and what kinds of animals certain shrines and temples held sacred. We were able to attend Gion Matsuri, a festival that took place near the beginning of our stay. We were also able to learn about why Fall in Kyoto is so popular for its' beauty. We also learned from our sponsor Professor Noma, as well as some fellow lab students, about different meanings behind certain colors. All of this new knowledge about cultural Kyoto and more can be seen in the final designs of our kyokubou, with their different types of dress, poses, and what animal they are.

Most importantly, we learned to be patient. When something went wrong, we fixed it. When someone was having trouble, we helped. We learned to work as a team, not as individuals.

7.2 What Went Right?

When we looked at what went well in *Step Symphony*, we were able to identify many positive outcomes. The first positive we identified was the design process. The design process itself went very smoothly, only taking about four weeks. Our design process was fast, yet

concise, and effectively outlined the scope of our game. Whenever a team member had a question about *Step Symphony's* overall design, it was usually resolved within minutes.

When first designing *Step Symphony*, our design and personal goals were to have a completely playable game. This playable game included:

- Working directional audio to guide the player, as well as an accompanying map screen with GPS tracking
- Augmented reality using the phone's camera to find the kyokubou in the real world
- A mechanic to capture the kyokubou to add to the player's collection
- A collection screen where the player could view their collected kyokubou, as well as play their respective notes
- Two different instruments for the kyokubou, as well as eight kyokubou to collect for each instrument
- Database to hold the kyokubou as well as their information and locations
- A second database to hold accounts, allowing players to sign up and log in

All of these design goals were met, and even exceeded, within our set timeframe. The Directional Sound and Snapshot (augmented reality) portions of *Step Symphony* were completed within a week, a lot less time than we anticipated. The player map to accompany the directional sound was slightly more difficult. Originally, MapzenGo, which the map is based on, was not compatible for GPS tracking or mobile applications. This took slightly longer to fix, two or three weeks. However, we were able to complete it.

The collection screen took less time, being completed within three or four days. Also, besides being able to view collected kyokubou as well as listen to their notes, we also added the

ability to play multiple notes at a time, allowing the player to create symphonies of their own. We also added the opportunity to learn more about each kyokubou and the location they were found in through a short description the player can choose to read.

The rhythm game took slightly longer to complete, as there was much more that needed to be implemented than we had originally thought. However, that was completed within two weeks. There was no lag, the game mechanic worked as intended, and it was very easy for playtesters to pick up with little to no instruction.

We also made sure that the kyokubou and UI assets for *Step Symphony* would be able to be completed within our time limit. At first, we were unsure if all of the necessary art assets would be completed in time, as we had lost a team member early in the design process. However, we were able to pull through and have all of the necessary assets completed with time to spare. All sixteen of the kyokubou, including Sarushi, and their respective descriptions and musical note tracks were completed to meet our high standards.

Our initial design called for two databases. The first database would hold all of the kyokubou, including their names, instruments, descriptions, and locations. The second database would hold user accounts, including their usernames, passwords, and collected kyokubou. While we hit many obstacles, we were able to deploy these servers.

7.3 What Went Wrong?

After going through what went right with *Step Symphony*, we then moved on to what went wrong. One key problem we identified was the scope of the game in later stages. At first, when we started as a group of four, the scope was within our limitations. However, when we

lost a team member in the early stages of making *Step Symphony*, we had to re-evaluate our scope. Unfortunately, this meant that we had to cut out user settings, map settings, more audio tracks, as well as the final animation reward for capturing all kyokubou. If given more time, we would have liked to implement these.

A second problem where we struggled was when we lost some of our art assets. Kun-Kun, our raccoon kyokubou, as well as some UI, was lost near the end of our timeline. This was a huge blow, as Kun-Kun was the final kyokubou that needed to be created. Fortunately, we were still able to redo all of the lost assets without losing too much time.

A third key problem we faced was deploying the server. This took weeks to do as we had never worked with databases before and were learning on the fly. Creating this database using Python and Django meant we had to use multiple frameworks within frameworks. If Python and Django had not been a requirement from our sponsor, we would have used a different method for our database.

Our final issue we ran into was one we could not fix, as it was a defect in the hardware, which we found during testing. When playtesting our game, we noticed that sometimes the directional sound would start leading the player in the opposite direction of where the kyokubou was placed. Upon further inspection, we learned that it was the phone's internal compass flipping, and it only happened in specific locations and always at those locations. This was for all phones and all applications that required GPS tracking. As this was a result of a hardware issue, we were unable to fix this problem.

7.4 What we would have changed

If given the chance to start *Step Symphony* over again, there would be a couple of things we would change. First of all, we would have used a different language when creating our databases. Using Python and Django for our databases made it more complex than it should have been; we had frameworks on top of frameworks. If it had not been a requirement, we would have created our databases differently.

After creating and implementing our capture mechanic, the rhythm game, we found a simpler parser, called MIDI to XML parser, for the creation of notes. Unfortunately, we did not have enough time to change to this simpler parser. Given more time, we would have liked to switch to this simpler parser for our rhythm game. We were not also completely happy with how the rhythm portion of the game looks, color-wise and position-wise. Given more time, we would have moved the note tracks to each end and the kyokubou to the center of the screen.

One final thing we would have done differently was start working on a settings screen earlier in the development process. We originally were going to have a settings screen, where the player could choose to turn snapshot mode on or off, reset their collection, and so on. Unfortunately, by the time we got around to working on implementing these settings, it was too late to do so. This will be implemented later in the year, but we would have liked to have done this earlier.

7.5 Future Plans

Step Symphony is far from complete. Given more time, there is much more we would have liked to add. During our initial design, we had more audio tracks and a final animation

reward for collecting all of the kyokubou. Originally, we were planning on having each kyokubou have their own unique audio track play during the rhythm game portion of *Step Symphony*. These unique tracks would feature the kyokubou's played instrument, as if they were playing the rhythm game track themselves. However, due to time constraints, we were unable to complete this goal. However, this will be implemented in the future.

We had originally wanted to award the player with a final animation when he/she had successfully captured all of the existing kyokubou. In the animation, Sarushi would be happily reunited with all of his kyokubou, and they would perform a short portion of their upcoming concert performance together to reward the player for collecting all of the other kyokubou. Compared to the rest of the game, the final reward would be partly rendered in 3D rather than 2D, so it felt more gratifying. At the time, we had planned to have only Sarushi be fully 3D, while the other kyokubou would be animated in 2D. This decision was made due to restraints in time and manpower, though animating 17 different characters would also likely be too graphically taxing for a phone.

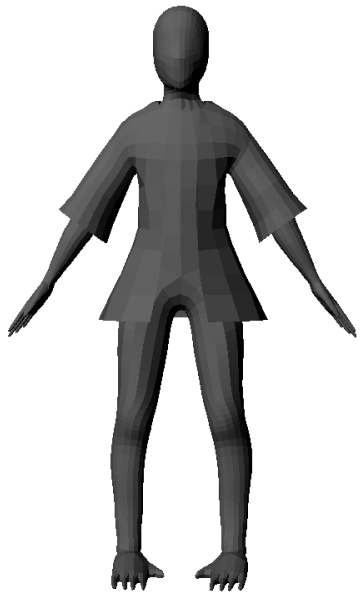


Figure 54: Incomplete Sarushi Model

Besides parts of *Step Symphony* that we were unable to implement due to time restraints, we have other plans in store. As a reminder, out user goals for *Step Symphony* included:

- To collect all of the different notes and instruments in multiple zones
- To discover places the user would have never thought to go to before
- To experience joy

Where *Step Symphony* currently is, we believe that we have met all of these user goals. However, there is still much we can do. For example, we would like to increase the scope of the game in the future. Why only play in Central Kyoto? Why only have Japanese instruments. In the future, we would like to expand to other places in the world, such as Washington D.C. and Paris. We would make more kyokubou to collect, with instruments that are applicable to the culture in these new locations. By expanding the game world of *Step Symphony,* as well as add more kyokubou, we would give the player more opportunities, as well as easier access to finding kyokubou, to complete the user goals we had in mind above.

Based on observations and feedback from playtesters, we decided that we would also want to implement some sort of visual feedback during the directional sound portion of *Step Symphony* to let the player know that they are heading in the right direction. Some of the non-sound cues would either be the phone vibrating when the player moves to face the kyokubou, or the player icon on the map screen blinking when the player is walking in the right direction. We would give the player the option of disabling this future feature.

We would also like to make a separate playable version for the visually impaired, a very underserved audience. We believe this to be possible because *Step Symphony's* core unique feature, directional sound, does not require sight. Instead of using the phone's camera, or augmented reality, to find the kyokubou when in range, we would instead have the phone vibrate when the player was near the kyokubou's location, and have the player tap the screen. We also would make a different variation of the rhythm capture mechanic. Instead, the rhythm game would have a drum beat, and the player would tap the screen in time with the drum beat. This way, *Step Symphony* could fully be played without any visuals.

8 References

AgoraVoxFrance. “Kodo - "O-Daiko" - HD (japanese drummers - Taiko - tambours géants Japon).” YouTube video, 08:24. Posted March, 2011. https://www.youtube.com/watch?v=C7HL5wYqAbU

Austin, Toby, and Vetter, Roger, “shinobue,” *Grinnell College Musical Instrument Collection*, accessed August 30,2016, https://omeka1.grinnell.edu/MusicalInstruments/items/show/315.

Azuma, R. T., (1997) A survey of augmented reality. Presence: Teleoperators and Virtual Environments 6, 4, MIT Press, 355–385.

Bertrand, Sebastien. *School children at Kiyomizu-dera*. April, 2005. Kyoto, Japan. https://commons.wikimedia.org/wiki/File:School_children_at_Kiyomizu-dera.jpg.

Cali, Joseph, Dougill John, and Ciotti Geoff. "Yasaka Jinja (Gion-sha)." In *Shinto Shrines: A Guide to the Sacred Sites of Japan's Ancient Religion*, 143-47. University of Hawai'i Press, 2013. http://www.jstor.org.ezproxy.wpi.edu/stable/j.ctt6wqfhm.25.

Craft, Lucy. “3 Strings And A Snakeskin: Okinawa’s Native Instrument.” nprmusic. December 4, 2012. Accessed August 1st, 2016. http://www.npr.org/2012/12/06/166416782/3-strings-and-a-snakeskin-okinawas-native-instrument

David Fallows. "Andante." Grove Music Online. Oxford Music Online. Oxford University Press, accessed September 6, 2016, http://www.oxfordmusiconline.com/subscriber/article/grove/music/00854.

Desurvire, Heather and Wiberg, Charlotte. *Game Usability Heuristice (PLAY) for Evaluating and Designing Better Games: The Next Iteration.* Springer Berlin Heidelberg. 2009.

Feuchter, Jennifer. *Signal Drum 01*. Taken May 8th, 2008. Accessed August 30th, 2016. https://www.flickr.com/photos/jennerosity/2571105659/in/set-72157606509856273

Fox, Stacey. Encyclopedia of Modern Asia. New York: Charles Scribner’s Sons, 2002. http://ic.galegroup.com.ezproxy.wpi.edu/ic/whic/ReferenceDetailsPage/ReferenceDetailsWindow?disableHighlighting=false&displayGroupName=Reference&currPage=&scanId=&query=&source=&prodId=WHIC&search_within_results=&p=WHIC%3AUHIC&mode=view&catId=&u=mlin_c_worpoly&limiter=&display-query=&displayGroups=&contentModules=&action=e&sortBy=&documentId=GALE%7CCX3403702633&windowstate=normal&activityType=&failOverType=&commentary=

Federoff, Melissa A. *Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games.* Published December, 2002. Accessed September 15th, 2016.

Fujie, Linda. "Effects of Urbanization on Matsuri-Bayashi in Tokyo." *Yearbook for Traditional Music* 15 (1983): 38-44.

Fujie, Linda. "Japanese Taiko Drumming in International Performance: Converging Musical Ideas in the Search for Success on Stage." *The World of Music* 43, no. 2/3 (2001): 93-101. http://www.jstor.org.ezproxy.wpi.edu/stable/41699367.

Gillan, Matt. *Songs from the Edge of Japan: Music-making in Yaeyama and Okinawa*. New York: Ashgate Publishing, 2009.

He, JianJun. *Spatial Audio Reproduction with Primary Ambient Extraction.* Springer Singapore, 2016.

Huber, D. M. (2012). The MIDI Manual, 3rd Edition. Burlington, MA: Focal Press

Hughes, David W. "*Traditional Folk Song in Modern Japan: Sources, Sentiment, Society"*.

Johnson, Henry. "Tsugaru Shamisen: From Region to Nation (and Beyond) and Back Again." Asian Music 37, no. 1 (2006): 75-100. http://www.jstor.org.ezproxy.wpi.edu/stable/4098489.

In the Living Room. "Beautiful Okinawa Music | Sanshin Folk | Jun Nishimoto | In the Living Room". YouTube video, 02:37. Posted September, 2014. https://www.youtube.com/watch?v=lkC7k2MHcYM

Marshall, F, Watts, R (Producers), & Zemeckis(Director), R. (1988). *Who Framed Roger Rabbit?* [Motion Picture]. United States: Touchstone Pictures Amblin Entertainment.

Matsue, Jennifer Milioto. *Focus: Music in Contemporary Japan.* New York: Routledge, 2016.

Matsuura, Hiroko & Chiba, Reiko & Hilderbrandt, Paul. *Beliefs about Learning and Teaching Communicative English in Japan.* 2001. Accessed August 31st, 2016. http://jalt-publications.org/archive/jj/2001a_JJ.pdf#page=66

McMullin, Neil. "Oda Nobunaga." In Buddhism and the State in Sixteenth-Century Japan, 59-94. Princeton University Press, 1984. http://www.jstor.org.ezproxy.wpi.edu/stable/j.ctt7zv18m.7.

Meigs, Tom. (2003). Ultimate game design: building game worlds. Emeryville, CA: The McGraw-Hill Companies

Nacke, L. (2009). From playability to a hierarchical game usability model. Proceedings of the 2009 Conference on Future Play on @ GDC Canada. Vancouver, Canada: ACM

Miki, Minoru, and Regan Marty. *Composing for Japanese Instruments*. Edited by Flavin Philip. Boydell and Brewer, 2008. http://www.jstor.org.ezproxy.wpi.edu/stable/10.7722/j.ctt14brw5x

Oilstreet. *Kiyomizu-dera(清水寺), Kyoto, Kyoto prefecture, Japan*. Dec 04, 2008. Kyoto, Japan. https://commons.wikimedia.org/wiki/File:Kiyomizu-dera_in_Kyoto-r.jpg.

Pro Musica Nipponia. "shinobue," Pro Musica Nipponia. 2002, accessed August 30, 2016. http://www.promusica.or.jp/english/shinobue.html

Rehak, B. (2003). Playing at being: Psychoanalysis and the avatar. In Wolf, M. J. and Perron, B (eds.), The Video Game Theory Reader. London: Routledge.

Robinson, Mark. "5 top tips for A/B Testing in free-to-play games" *Gamasutra.* Published November 2nd, 2015. http://www.gamasutra.com/blogs/MarkRobinson/20151102/258075/5_top_tips_for_AB_Testing_in_freetoplay_games.php

Sheldon, Charles. "The Nijo Castle, Kyoto." *History Today*.

Shimada, Akiko. 2009. *Cross-cultural music: Japanese flutes and their influence on western flute music*. Vol. 34. USA: National Flute Association, Inc.

STB-1. *Buffalo-bell and Buffalo-bull, mascots of the Orix Buffaloes, at Chiba Marine Stadium*. Aug 06, 2011. Chiba, Japan. https://commons.wikimedia.org/wiki/File:Buffalo_bell.jpg.

TheJapanTimes. "Disappointing levels of English." *TheJapanTimes*, Published March 28th, 2015.

TheJapanTimes. "Let's discuss English language education in Japan" *TheJapanTimes*, Published February 15th, 2016.

The Taiko Resource. "Overview and History". Taiko.com. Accessed August 2nd, 2016. http://www.taiko.com/taiko_resource/history.html

Tullis, Thomas and Albert, William. *Measuring the User Experience, 2nd Edition.* Waltham: Elsevier Inc. 2013.

Universidad Nacional de La Plata. Sanshin. Accessed August 30, 2016 http://www.unlp.edu.ar/articulo/2012/4/13/texto_castellano_azzarini_sanshin

Vogel, Brian. *Transmission and Performance of Taiko in Edo Bayashi, Hachijo, and Modern Kumi-daiko Styles.* Houston, 2009.

"Wind Instruments." In *Composing for Japanese Instruments*, edited by FLAVIN PHILIP, by MIKI MINORU and REGAN MARTY, 6-70. Boydell and Brewer, 2008. http://www.jstor.org.ezproxy.wpi.edu/stable/10.7722/j.ctt14brw5x.8.

Yanajin33. "Yasaka Shrine". Wikimedia. Published July 20th, 2015. Accessed September 20th, 2016. https://commons.wikimedia.org/wiki/File:Yasaka_Shrine_-_Romon3.jpg

Appendix A: Art Asset List

Kyokubo Assets

Asset Name	Instrument	Animal	Level of Done-ness
Kataski	Shamisen	Fox	
Kokoko	Flute	Fox	
Ai	Shamisen	Bunny	
Tomaru	Shamisen	Bunny	
Osachi	Flute	Rooster	
Hi	shamisen	Shisa	
Komiya	Flute	Cat	
Sarushi	Taiko	Monkey	
Shouji	Flute	Giant Salamander	
Ushikyuu	Flute	Akaibeko	
Fuwari	Flute	Shisa	
Tega	Flute	Squid	
Yushiya	Flute	Bear	
Kumagamine	Shamisen	Bear	
Inokoza	Shamisen	Pig	
Ringo	Shamisen	Salamander	
Kyokubo	Shamisen	Raccoon	

Done-ness Chart	
Not Started	
Started	
Roughed	
Finalized	

UI Assets

Asset Name	Screen	Function	Done-ness
Menu Button	Map	Opens the menu	
AR Button	Map	Enters AR mode when the player is near a Kyokubo	
Central Collection Button	Collections	Switches the kyokubo instrument type that the player is looking at	
Exit Button	AR Screen	Exits AR and returns the player to the map screen	
Outer Collection Buttons (1-8)	Collections	Plays a sound relevant to the kyokubo it belongs to	
Log In Button	Start Screen	Used to log in to Step Symphony	
Sign Up Button	Start Screen	Used to create an account in Step Symphony	
Splash Screen	Splash Screen	Start Screen	
Username textbox	Sign Up	Textbox for the player to enter their username	
Password textbox	Sign Up	Textbox for the player to enter their password	
Go button	Sign Up	Proceeds to the game	
Player Arrow	Map Screen	Shows the player's location on the map	
Go Back	Menu Screen	Takes the user back to the last screen they were on	
Note	Capture Screen	The note that will appear on the track	
Note Spawner	Capture Screen	The object which the note will come out of	
Sarushi Yelling Loading	Loading Screen	Tells the player that it is LOADING!	

Appendix B: Kyokubou Descriptions

Kyokubou	Description
Ai	Ai spends most of her time at Kiyomizu-dera. Every week, she visits the love stones and tries to walk between them with her eyes closed. Every week, she fails. But that's fine. She'll just go play a few tunes, drink to her health from the waterfall, eat some shaved ice with Komiya, and try again next week. Shopping doesn't hurt either.
Hii	Hii and Kokoko have been best friends since they were small. In fact, it was Kokoko's father that taught Hii to play the shamisen. However, Hii's true passion is dancing. If you want to find Hii, there are two places to look. Either she is around Nishiki Market, singing and dancing, or she is with Kokoko.
Inokoza	You may have seen Inokoza at the Kabuki Theater. He's been in many performances there over the past few years, always sitting on stage next to his friend Kataaki. Inokoza lives close by, near the park by Yasaka Shrine. He likes to spend time there, paying for visitors and tourists alike. On rare occasions, you may even see him dance.
Kataaki	Kataaki grew up playing the sanshin in Okinawa. His parents never approved, seeing him as lazy for spending less time helping around the house to play. When he was old enough, he moved to Kyoto and learned to play the shamisen. During Gion Matsuri, Kataaki can be seen near Yasaka Shrine eating yakitori and playing his shamisen with his daughter Kokoko.
Kumagamine	Kumagamine loves taking naps in the trees at Kyoto Imperial Palace after class. It's easy for him to get there as well, as he attends a high school nearby. Kumagamine is learning to sing and play the shamisen, just like his grandfather. If you do happen to find him asleep, please wake him gently. He may play a song for you.
Kun-Kun	Every afternoon, Kun-Kun goes to visit Honnoji Temple. While there, he feeds the fish, then sits and waits. Every single day, Kun-Kun sits there until the sun is long set, only to leave the way he came with a sad smile on his face. No one knows why Kun-Kun does this. Is he maybe waiting for someone?
Ringo	Ringo is a big girl now, she just turned six! She loves running around Kyoto Aquarium and Umekoji Park, chasing after Uncle Tega. Tega always gets upset when Ringo catches him, and Ringo doesn't know why. Ringo just started learning to play the shamisen. She's not very good yet, but will get better. Maybe one day she'll get to perform during the dolphin show!
Tomaru	Tomaru loves to spend time in Okazaki park. On some days, she plays her shamisen. On other days, she paints. Some day, Tomaru hopes to have one of her

	paintings displayed in the Museum of Modern Art. Until then, she'll spend her days sitting in the shade, painting and playing.
Fuwari	Fuwari often accompanies Shouji to Nijo Castle. When he tells his tales to visitors, Fuwari performs these stories by dancing and playing her Shinobue. Sometimes, when the sun is hot in the sky and she needs a change of scenery, Fuwari will take Shouji to visit Osachi at Shosei-en Garden.
Kokoko	As a child, Kokoko loved going to the Minamiza Kabuki Theatre. Her mother would take her every weekend to watch her father, Kataaki, play. Now, Kokoko spends much of her time performing on stage, as well as playing the Shinobue with her father on the streets of Gion.
Komiya	Komiya likes people. People are great. They pet Komiya. take pictures with Komiya. In return, Komiya helps the people. Shows them the best shops walking up to Kiyomizu-dera. Shows them the best places to take pictures. Komiya also spends time with Ai, cheering her on as she tries to reach the other love stone. Komiya is pretty happy.
Osachi	Osachi is always near Nishi and Higashi Honganji Temple. Every morning as the sun rises, he sweeps the front gates for visitors. During the day, he carries messages between the two. When the sun is bright in the sky and the weather is nice, Osachi will visit the Shosei-en Garden with his friend Tega, where they will play their shinobue as they watch the fish swim by.
Shouji	If you want to find Shouji, always look near Nijo Castle. Check the entrance first, he may be welcoming guests in. Check the water next, he may have gone for a swim. When all else fails, check the gardens. That's where Shouji tells his stories. And trust me, Shouji's tales are wonders that you do not want to miss.
Tega	If you know what's good for you, never mention the Kyoto Aquarium in front of Tega. Every day, he tries to escape. Every time, he's always caught in either Umekoji Park or Shosei-en Garden and brought back. Tega doesn't want to be there, he wants to spend time with Osachi. So don't send him back to the Aquarium. He'll never forgive you.
Ushikyuu	Ushikyuu is rarely seen away from Myoshin-ji. If you want to find him, look outdoors. In the mornings, Ushikyuu watches the sun rise. At dusk, he watches the sun set. During the day, look near the temple bell. You'll find him there, playing his shinobue for anyone who cares to listen.
Yushiya	Whenever Ai has questions about the latest fashion trends, she always goes to Yushiya for help. Yushiya works at a clothing store in Kyoto Station, and is always excited to talk about new makeup and mail polish. If you need advice on coordinating outfits, or just want company going to the spa, then never hesitate to ask Yushiya. She will never say no.

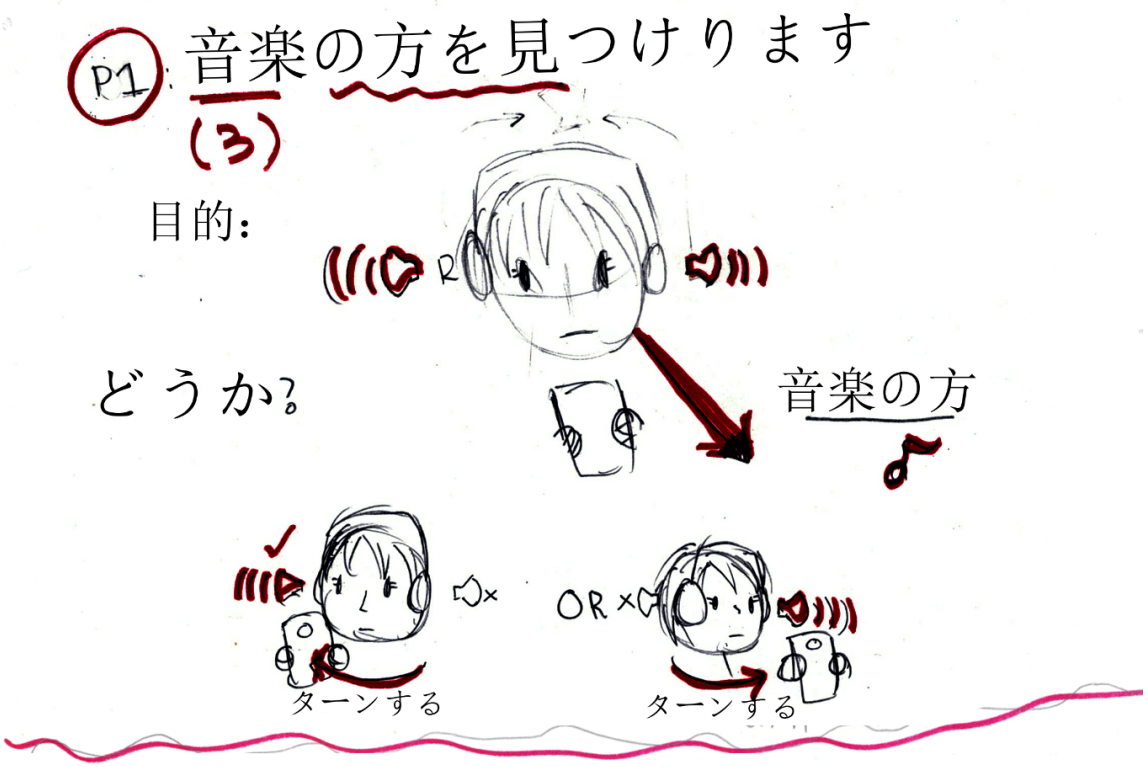
Appendix C: Kyokubou Images and Names

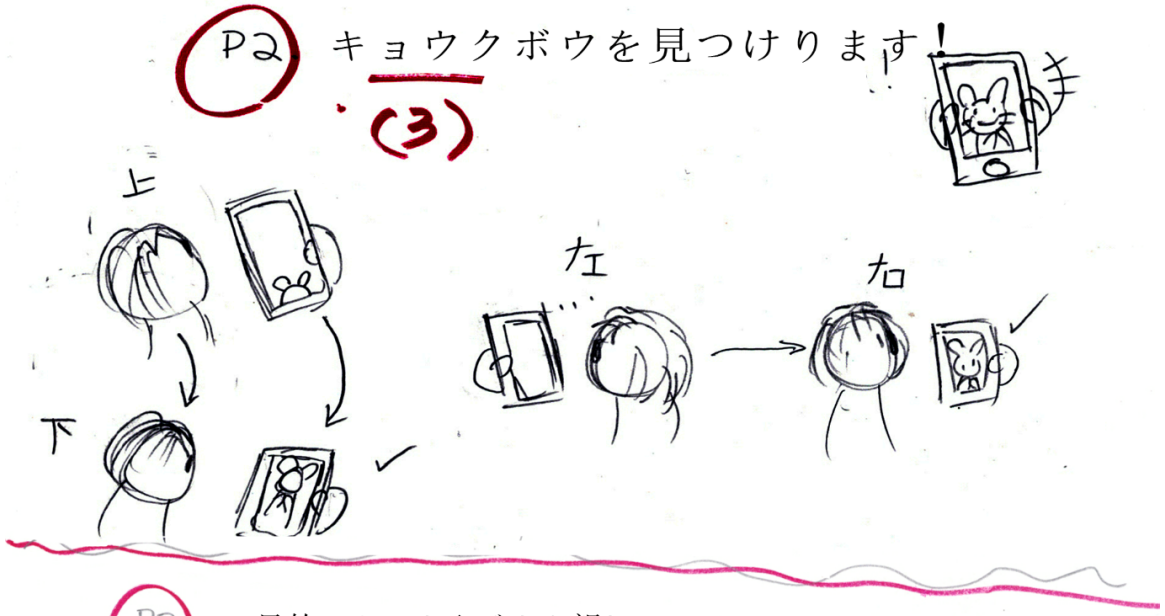
 **Ai** *Rabbit* *Shamisen*	 **Hii** *Shisa* *Shamisen*	 **Inokoza** *Pig* *Shamisen*	 **Kataaki** *Fox* *Shamisen*
 **Kumagamine** *Bear* *Shamisen*	 **Kun-Kun** *Raccoon* *Shamisen*	 **Tomaru** *Rabbit* *Shamisen*	 **Ringo** *Giant Salamander* *Shamisen*
 **Fuwari** *Shisa* *Shinobue*	 **Kokoko** *Fox* *Shinobue*	 **Komiya** *Cat* *Shinobue*	 **Osachi** *Rooster* *Shinobue*
 **Shouji** *Giant Salamander* *Shinobue*	 **Yushiya** *Bear* *Shinobue*	 **Tega** *Squid* *Shinobue*	 **Ushikyuu** *Akabeko* *Shinobue*

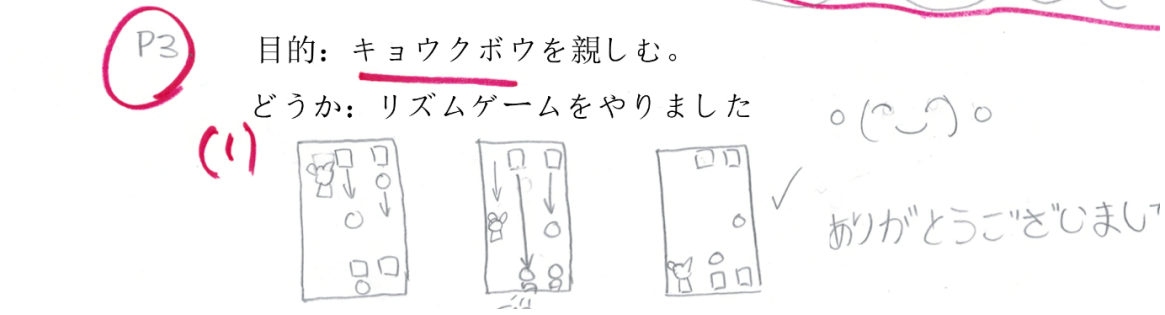
Appendix D: Audio Asset List

Asset Name	Type	Instrument	Length(min:sec)	Completion
col_shamisen_01	Collection	Shamisen	0:04	Done
col_shamisen_02	Collection	Shamisen	0:04	Done
col_shamisen_03	Collection	Shamisen	0:04	Done
col_shamisen_04	Collection	Shamisen	0:04	Done
col_shamisen_05	Collection	Shamisen	0:04	Done
col_shamisen_06	Collection	Shamisen	0:04	Done
col_shamisen_07	Collection	Shamisen	0:04	Done
col_shamisen_08	Collection	Shamisen	0:04	Done
col_shinobue_01	Collection	Shinobue	0:04	Done
col_shinobue_02	Collection	Shinobue	0:04	Done
col_shinobue_03	Collection	Shinobue	0:04	Done
col_shinobue_04	Collection	Shinobue	0:04	Done
col_shinobue_05	Collection	Shinobue	0:04	Done
col_shinobue_06	Collection	Shinobue	0:04	Done
col_shinobue_07	Collection	Shinobue	0:04	Done
col_shinobue_08	Collection	Shinobue	0:04	Done
bgm_final_reward	BGM	Shamisen,Shinobue,Taiko	0:15	Not Started
bgm_shamisen_reward	BGM	Shamisen	0:15	Not Started
bgm_shinobue_reward	BGM	Shinobue	0:15	Not Started
bgm_menu	BGM		0:30	Done
bgm_map_screen_01	BGM		0:30	Done
bgm_map_screen_02	BGM		0:30	Not Started
bgm_capture_success	BGM		0:05	Not Started
bgm_capture_failure	BGM		0:05	Not Started
sfx_selection	SFX		0:02	Not Started
sfx_ar_appear	SFX		0:02	Not Started
sfx_button_press	SFX		0:02	Done
trailer_music	Trailer		0:30	Done
cm_shamisen_01	CaptureMusic	Japanese Medley Solo Shamisen	0:45	Not Started

Appendix E: *Step Symphony* Test Instructions







Appendix F: *Step Symphony* Pre-Test Survey

Pre-Test Survey

Do you currently own a smartphone? ☐ Yes ☐ No

Do you play any mobile games? ☐ Yes ☐ No

If yes, how many hours a week do you think you play mobile games?

☐ 0-5 ☐ 6-10 ☐ 10-15 ☐ 16-20 ☐ 20+

Do you play mobile games with the sound off or on? ☐ Off ☐ On

Do you play any rhythm games? ☐ Yes ☐ No

If yes, please list your favorite rhythm games.

_____
_____
_____

What are your 3 favorite game genres? (Please list your top 3 in descending order)

1._____
2._____
3._____

Do you like to listen to music while walking? ☐ Yes ☐ No

Appendix G: *Step Symphony* Post-Test Surveys

Post-Test Survey (Controlled Testing)

Please read each statement and circle the number that best represents your thoughts on it.

1 = Strongly Disagree 5 = Strongly Agree

Directional Sound

It was easy to follow the directional sound.	1	2	3	4	5
I am interested in this as a game mechanic.	1	2	3	4	5

Snapshot Mode (Finding the Kyokubou)

It was easy to find the Kyokubou.	1	2	3	4	5
I enjoyed finding the Kyokubou.	1	2	3	4	5

Capturing the Kyokubou

The controls were straightforward and easy to use.	1	2	3	4	5
The game held my focus.	1	2	3	4	5
Capturing the Kyokubou was fun.	1	2	3	4	5

Would you be interested in playing the full game? Why or why not?

_____
_____
_____
_____
_____

Additional Comments:

_____
_____
_____
_____
_____

Post-Test Survey (Uncontrolled Testing)

Please read each statement and circle the number that best represents your thoughts on it.

1 = Strongly Disagree 5 = Strongly Agree

The game controls and interface were easy to use.	1	2	3	4	5
The game held my focus.	1	2	3	4	5
I kept getting easily distracted by other things while playing.	1	2	3	4	5
I had no trouble following the music to find Kyokubou.	1	2	3	4	5
I'm interested in seeing all of the other available Kyokubou.	1	2	3	4	5
I would recommend *Step Symphony* to a friend.	1	2	3	4	5

What did you like best about *Step Symphony*? Why?

_____
_____
_____
_____
_____
_____

What did you like least about *Step Symphony*? Why?

_____
_____
_____
_____
_____
_____

What part of the game did you find the easiest? What part did you find most difficult?

_____
_____
_____
_____
_____
_____

Would you play *Step Symphony* again? Why or why not?

_____
_____
_____
_____
_____
_____

Additional Comments:

_____
_____
_____
_____
_____
_____