# ~Konbini Konnection~

## A Competitive Location-Based Multiplayer Game

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 and
Degree of Bachelor of Arts

By:
Erik Cerini
Laurie Mazza

Advised by
Jennifer deWinter
Clifford Lindsay

# Abstract

*Konbini Konnection* is a mobile social interaction game with the goal of producing an experience that encouraged communication between players in the same area as each other. In this paper, we discuss the overall design process we followed to implement a game of this nature. From a gameplay design perspective, we discuss expansive, modular elements and their use in the construction of a positive, social experience. From an artistic design perspective, we discuss the process of creating a character creation system, as well as forming a single, cohesive art style out of several disjointed ones. Lastly, from a technical design perspective, we discuss the design of databases for player data and analytics, as well as the construction of a gameplay server. While there is still room for improvement, we were generally successful in producing an enjoyable experience for those who interacted with the game.

# Acknowledgements

We would first like to thank our advisors Jennifer deWinter and Clifford Lindsay for their patience, understanding, and support in the creation of this project. They gave strong creative feedback, without which *Konbini Konnection* would not be the game that it is today.

We would also like to thank Noma-sensei, Lopez-sensei, and Ruck-sensei and their students at Ritsumeikan University for offering us lab space to work at during our stay in Japan. They created a welcoming environment and provided us with any tools we needed to create this project.

In addition, our thanks go out to the staff at the Ritsumeikan Biwako-Kusatsu Campus International House. They provided us with safe, clean, and comfortable living space, as well as advice on good places to visit wherever we went in Japan.

Next, we would like to thank our families for their love and support, without which *Konbini Konnection* would not exist.

Finally, we would like to thank our playtesters. They played an instrumental part in the development and refinement of *Konbini Konnection* by locating bugs and offering ideas for potential improvements.

# Table of Contents

# List of Figures

# Code Listings

# Table of Authorship

| Section | Author |
|---|---|
| 1. Introduction | Erik Cerini |
| 2. Design and Gameplay | Erik Cerini |
| 2.1 Hardware & Platform | Erik Cerini |
| 2.2 Inspiration | Erik Cerini |
| 2.3 Target Audience | Laurie Mazza |
| 2.4 Solo Play | Erik Cerini |
| 2.5 Microgames | Erik Cerini |
| 2.6 Battle Mode | Erik Cerini |
| 2.7 Minigames | Erik Cerini |
| 2.8 Feedback Systems | Erik Cerini |
| 2.9 Proof of Concept | Erik Cerini |
| 3 Artistic Design Process | Laurie Mazza |
| 3.1 Overall Style/Inspiration | Laurie Mazza |
| 3.2 Character Creation | Laurie Mazza |
| 3.3 Games | Laurie Mazza |
| 4 Technical Design | Erik Cerini |
| 4.1 Login System and Password Encryption | Erik Cerini |
| 4.2 Databases | Laurie Mazza |
| 4.3 Location Services | Erik Cerini |
| 4.4 Network Communication | Erik Cerini |
| 4.5 Modular Microgames | Erik Cerini |
| 5 Testing Process | Laurie Mazza |
| 5.1 Challenges | Laurie Mazza |

| | |
|---|---|
| 5.2 Methods | Laurie Mazza |
| 5.3 Sessions | Laurie Mazza |
| 6 Conclusion | Both |
| Appendix B Session 2 Info Packet | Both |
| Appendix C *Dash Conductor* | Laurie Mazza |

# 1. Introduction

*Konbini Konnection* is a location-based, social interaction game through which players can challenge each other to battles composed of smaller, disjointed games to form a cohesive experience. Players can use their mobile devices to find others in their area that are also playing the game and opt to compete against them. In *Konbini Konnection*, players take charge of their very own コンビニ (Konbini), or Convenience Store. They perform various tasks, such as stocking shelves and checking customers, by playing microgames.

The goal of our design was to provide a positive, social experience that players would continuously engage in and enjoy. This was done by providing players with cute visuals, extrinsic motivators, and frenetic gameplay that would provide positive reinforcement of play.

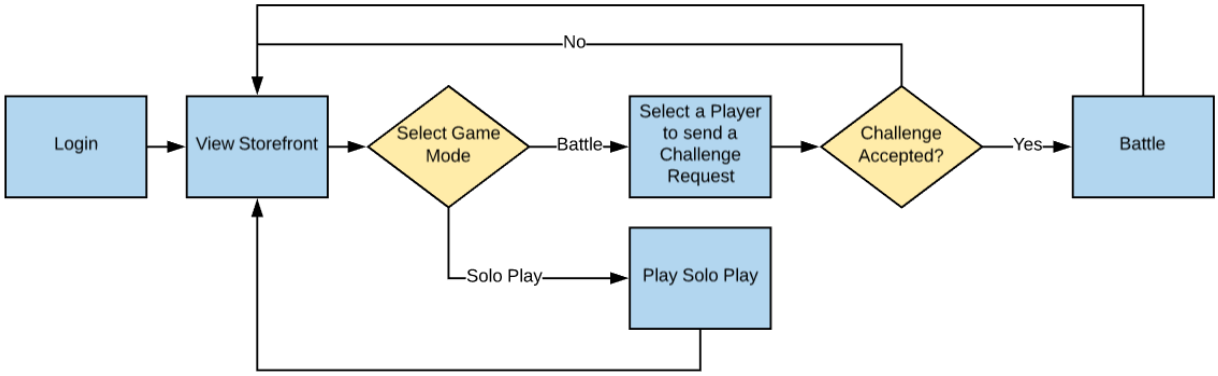



Figure 1.1: Overview of *Konbini Konnection*'s gameplay

Figure 1.1 shows a brief overview of the gameplay offered by *Konbini Konnection*. Players begin by logging into their account and accessing the Storefront. From here, they can choose to either play on their own, or play with other, nearby players. If they choose to play by themselves, they will be brought to the Solo Play menu, where they can select one of the Microgames that they've unlocked and play through it. Doing this will grant them proficiency points, explained further in section 2.8.1 Proficiency Points, that will aid them in battle.

Alternatively, they may select to play our Battle mode, which is meant to be the core experience of our social game. They can do this by selecting one of the avatars that appear on the Storefront screen and sending a challenge request to its associated player. Upon being accepted, they will be brought into a battle, where both players will race to see who can complete more Microgames in order to earn an advantage in the final, winner-takes-all, head-to-head Minigame. Figure 1.2 outlines all of the Micro and Minigames that have been designed for *Konbini Konnection*.

| Game Name | Game Type | Description |
|---|---|---|
| Bento | Microgame | Slash ingredients out of the air to make a delicious Bento Box Lunch. |
| Cats | Microgame | Follow the clues hidden behind objects on the bookshelf to catch the mice. |
| Doors | Microgame | Open a series of doors to get to the end of the hallway. |
| Dumplings | Minigame | Face off against your opponent to see who can pick the most flowers. |

Figure 1.2: All Micro and Minigames in *Konbini Konnection*

## 1.1 User Story

Karen is a 20-year-old college student sitting on the Quad between classes. She has a few minutes to kill, so she opens up *Konbini Konnection* on her phone and logs in. On her Storefront (Figure 1.3), she sees a couple of other students in the area that are also playing the game. After looking at the users in her Storefront, she sees her friend Jake's avatar and that he's not currently in a battle, so she taps on his avatar and sends him a challenge. He presses the big, green "accept" button and the prepares for battle.

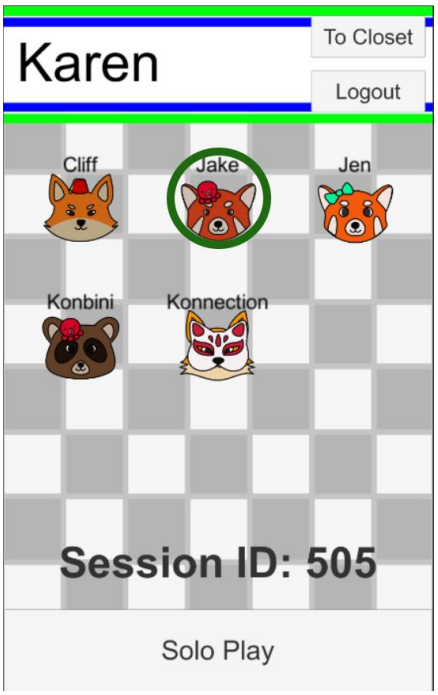



Figure 1.3: Karen's Storefront

The battle begins once both players are ready. A 3x3 grid of products fills the screen, representing the various Microgames that are available for them to play this time (Figure 1.4).

Since an advantage will be given based on who can complete the majority of the 9 games, Karen quickly taps the Cats microgame in the center of the top row, her personal favorite, and completes it with ease. Then she completes the Doors microgame on the left side of the middle row. She and Jake continue completing games until all 9 have been played.

Figure 1.4: Karen and Jake's Battle screen

Going into the Minigame phase, Karen completed 4 microgames and Jake completed the other 5. This means that in the upcoming, randomly-selected game, Dumplings, Jake will have a slight advantage over Karen. The two go on, picking flowers and eating dumplings until they both run out of Stamina and the game ends. Karen looks at the scores and see that hers is higher, meaning that she wins the battle! Her Punch Card gets a big, green stamp and the social level between her and Jake goes up on both players' devices. Jake walks up to her after the battle, thanks her for the game, and the two walk off together to their Biology class.

## 1.2 Paper Structure

In this paper, we detail the design and testing of a portion of the full vision of *Konbini Konnection*. Throughout the paper, we discuss various artistic, gameplay, and technical design challenges that arose and how we solved them. We also detail the process and findings of testing in our game, as well as plans for future improvements.

In Chapter 2, we discuss the full intended design of *Konbini Konnection* and how this design works to realize our experience goal. We discuss the Platform and Target Audience, Inspiration material, and feedback design. In addition, we discuss the various gameplay elements involved in a play session, such as game modes, Microgames, and Minigames. Finally, we define the aspects of the design that were included within the scope of this project.

In Chapter 3, we describe the artistic design involved in developing the proof of concept. This includes topics of cuteness and art style involved in producing a visually appealing experience. We discuss the process of designing a Character Creation system and how the

animals used in it were selected based on topics of color, body-type, and cultural significance. Finally, we discuss the artistic design of the games and how each one differs from the overall design of the game.

In Chapter 4, we talk about the technical challenges associated with the development of the game. These challenges included designing a secure login system, setting up location services, communicating between devices, and designing modular game blocks to build the experience. In addition, we discuss the challenge of structuring a database to house our player information and analytics data.

In Chapter 5, we go over the testing process of *Konbini Konnection*. This includes our testing methodology, as well as an analysis of the results. We also discuss the problems that occurred and the changes that we made as a result of these sessions.

Finally, in Chapter 6, we discuss our overall experience designing *Konbini Konnection* through a postmortem. We go over what we learned, what went right, and what went wrong. We also discuss future plans for development of the game.

Our Appendices feature extra materials that are referenced in our paper. This includes a project that was worked on alongside the MQP with a team of students at Ritsumeikan University. This project, named *Dash Conductor*, explores the use of Amazon Dash Buttons as a novel user interface and can be read about in Appendix C.

# 2. Design and Gameplay

The experience created by *Konbini Konnection* is one with heavy emphasis on creating positive, competitive encounters between players through social interaction. The design parameters that we outlined to facilitate this experience goal were:

- High player-to-player interactivity.
- Location-based gameplay.
- Easy expandability.

In addition, since *Konbini Konnection* was completed as an on-site MQP at Ritsumeikan Universit's Biwako-Kusatsu Campus in Shiga Prefecture, Japan, we were expected to include elements of Japanese Culture into the game. We decided to focus on the popularity of Konbinis and Mascot characters, as seen through the art, as well as various idiomatic phrases, which were used to inspire the design of our Micro and Minigames.

In this chapter, we discuss the design of *Konbini Konnection*, as well as the research that informed that design.

## 2.1 Hardware & Platform

We selected mobile devices as the platform for *Konbini Konnection* because the portability, control scheme, and haptic feedback systems that they employ foster the creation of meaningful social encounters. The portable nature of phones allows players to relocate to areas with a high saturation of players. Richard Bartle says in his paper on Multi-User Displays that in order to emphasize player interaction, it is important that players are able to find and meet each other (1996). Allowing players to move to create these highly populated areas was an important reason for *Konbini Konnection* to utilize the mobile platform.

In addition, mobile devices use gesture controls, which are easy to learn and facilitate positive, enjoyable interaction sessions for users. Kyle Sanders of Smashing Magazine (2017) presents Figure 2.1 below on the topic of gesture-based interaction.

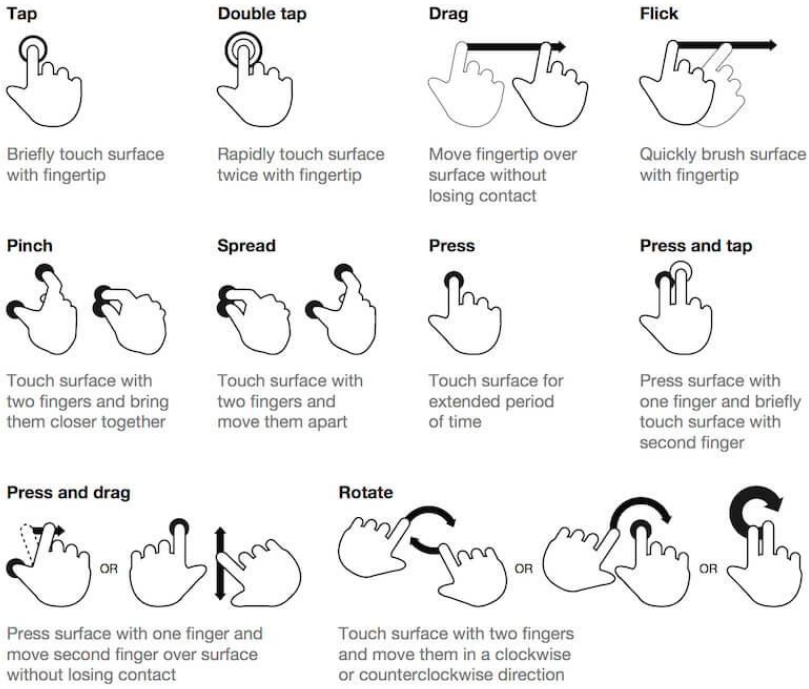



Figure 2.1: Several baseline gestures for touch interfaces (Sanders, 2017)

All of the gestures in Figure 2.1 are common in most mobile applications and stem from the simplification of real world actions. Sanders discusses that their low learning curve makes them enjoyable to use and encourages a positive reaction from the user. By making use of gesture controls, we were able to make an interface that was easy to operate within, which would allow players to focus more on the overall experience.

Finally, the delivery of feedback is important to the design of a cohesive and enjoyable experience. Research on the emotional feedback associated with mobile devices shows that "mobile devices such as mobile phones are very personal items that are used every day. The sense of touch in such devices creates a physical and emotional bond with the user" (Symeonidis). The haptic feedback associated with directly interacting with objects on your screen elicits a personal and emotional response that is important for encouraging players to continuously interact with your game.

## 2.2 Inspiration

*Konbini Konnection* was heavily inspired by the *WarioWare* (2003) game franchise, specifically its visuals, modular gameplay, and interesting multiplayer game modes. The whimsical visuals included throughout the series elicit a positive, lighthearted experience and inspired much of our artistic design. The "microgames" that make up the core gameplay loop were simple, accessible, and easy to implement in a modular fashion. The multiplayer game modes involved in *WarioWare Inc.: Mega Microgame$* (2003), *WarioWare Inc.: Mega Party Game$!* (2003) and *WarioWare: Smooth Moves* (2006) provided interesting and enjoyable player-to-player experiences composed of these "microgames." The models of gameplay that the *WarioWare* franchise introduced were in line with the experience we wanted to create, encouraging us to use it as a jumping off point for *Konbini Konnection*.

The secondary inspirations for *Konbini Konnection* were mobile social interaction games, such as *Streetpass Mii Plaza*(2011) and *Pokemon Go* (2016). Looking back at our goal to create a social interaction game that encouraged interaction outside of the game, it seemed important to look at other games that did this with some level of success.

*Streetpass Mii Plaza* is a social interaction game developed by Nintendo that encouraged players to always have their 3DS system with them, wherever they went. The game itself relies on the transfer of data between two 3DS systems that get within a certain range of each other, causing them to engage in a "Streetpass" (Robertson, 2013). When two systems "Streetpass" each other, their player avatars, referred to as a "Mii," are shared, as well as the data for any games that have Streetpass functionality enabled. These Miis can then be found in the *Streetpass Mii Plaza* game, where players can interact with them and play games with them. By playing these games, players can earn hats and speech bubble designs for their character that will be shared the next time they Streetpass with someone. The sense of personal representation, created by the character customization system, combined with extrinsic motivators to encourage continued use of the system helped to inspire those same systems in *Konbini Konnection*.

*Pokemon Go* is a social geocaching game developed by Niantic, Inc. following the success of their previous geocaching game, *Ingress* (2012). The game uses the nostalgic characters from Nintendo's *Pokemon* franchise to encourage players to travel to locations outside of the game world in hopes of receiving extrinsic rewards. The game's ability to get people to travel outside and condense in areas containing "Pokestops" caused increased player-to-player interaction outside of the game. Player Max Gayler reports that *Pokemon Go* is "a revolutionary new way we can engage with one another, at a time where human interaction is becoming ever more scarce" (2016). The use of location-based systems to encourage real world social interactions made the game highly successful and inspirational with regards to the experience that we wanted to create.

## 2.3 Target Audience

The goal target audience of *Konbini Konnection* is smartphone users within the 14 to 23 age range. This is similar to games such as *Miitomo* (2016) and *Pokemon Go (2016),* as they would be market competitors. *Miitomo* was designed to be similar to popular social media applications such as Instagram in order to target millennials (Harmon, 2016). Similarly, *Pokemon Go* had an average player demographic of 25-year-old females at two weeks after it's initial launch (Sonders, 2016). The age range of our target audience is the typical age range of high school to college students, who typically spend most of their time on a campus with many others. A location based game that relies on others to play would work well with this audience, as the market competitors show. A game with short play sessions would benefit this audience as they favor using public transportation, meaning they have short bursts of time to kill waiting (Goodyear, 2014). Keeping the knowledge of the goal target audience in mind, *Konbini Konnection* was designed to keep in line with its market competitors.

## 2.4 Gameplay Overview

The focus of *Konbini Konnection*'s gameplay is to produce a positive social experience that players will continuously engage in. To do this, we broke our game down into a set of components: The Battle, Solo Play, Microgames, Minigames, and Feedback. Each of these components fulfills a different role in generating this experience. The Battle Mode (Section 2.5) is the core social interaction of our game, making it the most important part of the experience. In

the Battle, the players will play Microgames (Section 2.7) and Minigames (Section 2.8), which offer frenetic and exciting gameplay to keep players engaged, with the goal of defeating their opponent. In Solo Play, players can play individual Microgames for the sake of practicing and gaining proficiency points that will give them an advantage in a battle. Finally, the Feedback system integrated into the design provide players with a set of extrinsic motivators that reward them for playing the game.

## 2.5 Battle Mode

The Battle system is the core of the *Konbini Konnection* gameplay experience, and it is designed to emphasize the creation of a positive, social experience. From the storefront, Players can challenge anybody within a 50 meter radius of them to a battle of microgames. This allows for players to locate people to play with that they may not know, encouraging a new social experience. As shown in Figure 2.2, a Battle consists of two phases: the Microgame Phase and the Minigame Phase.

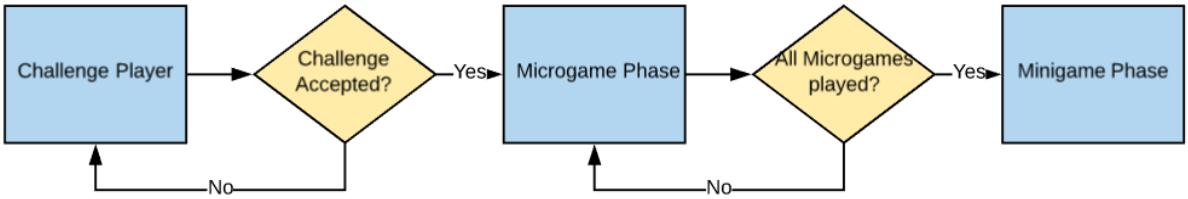



Figure 2.2: Flow of a Battle

### 2.5.1 The Microgame Phase

In the Microgame Phase, players will compete head-to-head to try to complete as many of the nine available games as they can before their opponent. The interaction cycle of a typical Microgame Phase is seen in Figure 2.3. The microgames are lined up in a 3x3 grid of items that a customer is trying to purchase, such as chips for the Cats Microgame and a Magazine for the Doors Microgame. A player can tap any of them that have neither been checked nor are currently being checked, causing that Microgame to begin. If the player is successful, he checks that Microgame. If not, the item is returned to the grid and is able to be selected by either player. Once all Microgames are checked successfully, the Minigame Phase will begin.

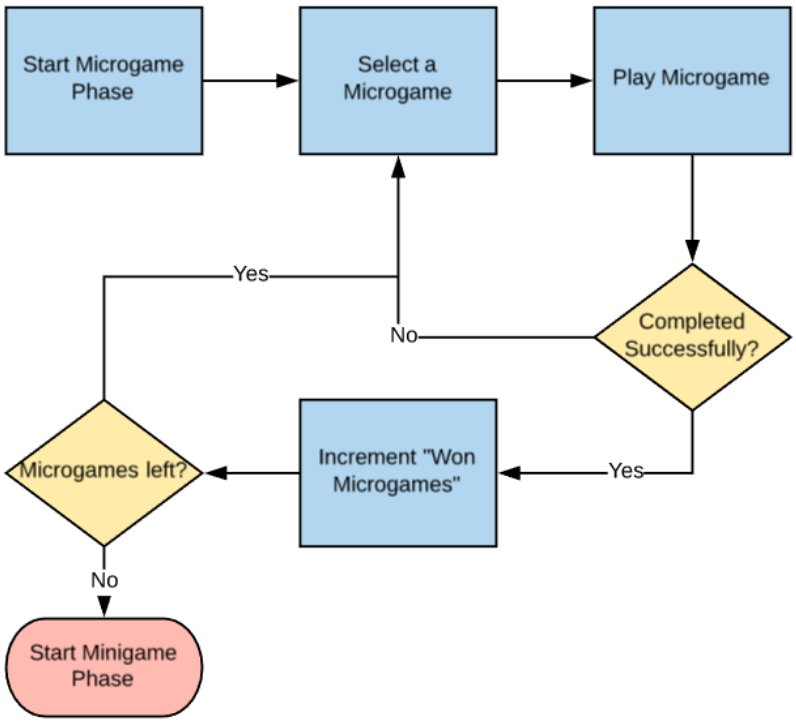



Figure 2.3: Flow of the Microgame Phase

Figure 2.4 shows the current implementation of the Microgame Phase. The player is presented with a grid of Microgames, as mentioned previously, and receives feedback as to which player has completed each Microgame. A Microgame that is currently not selectable will appear grayed out because it is either completed or being played by the other player. When a Microgame is completed, it is surrounded by a circle of a different color depending on the player that completed it: Blue circles represent the player, Red circles represent the opponent.


Figure 2.4: A screenshot of the Microgame Phase

During the Microgame Phase, players can utilize items that they have purchased from the Supplier in order to gain an advantage. These items are applied to Microgames and can augment them in various ways, such as decreasing the time limit or increasing the number of objectives that exist within the context. For example, if a player were to apply a “50% Off Coupon” to a Doors Magazine, her opponent would have half the normal amount of time to open all of the doors.

### 2.5.2 The Minigame Phase

In the Minigame Phase, players will compete head-to-head in one, larger competitive challenge. The winner of this challenge will be crowned the winner of the Battle. In order to tie the Microgame and Minigame phases together, every Minigame has at least one value that is variable based on how many Microgames the player completed in the previous phase.

## 2.6 Solo Play

A desire to keep players continuously engaged with *Konbini Konnection*, even without nearby opponents, informed the design of our Solo Play mode. A single play session in Solo Play is explained in Figure 2.5. When players enter Solo Play mode, they are able to select any of the microgames that they have unlocked. Each game has a short “Info” screen that provides a general overview of the game’s mechanics that the player can access at any time from the Solo Play menu.

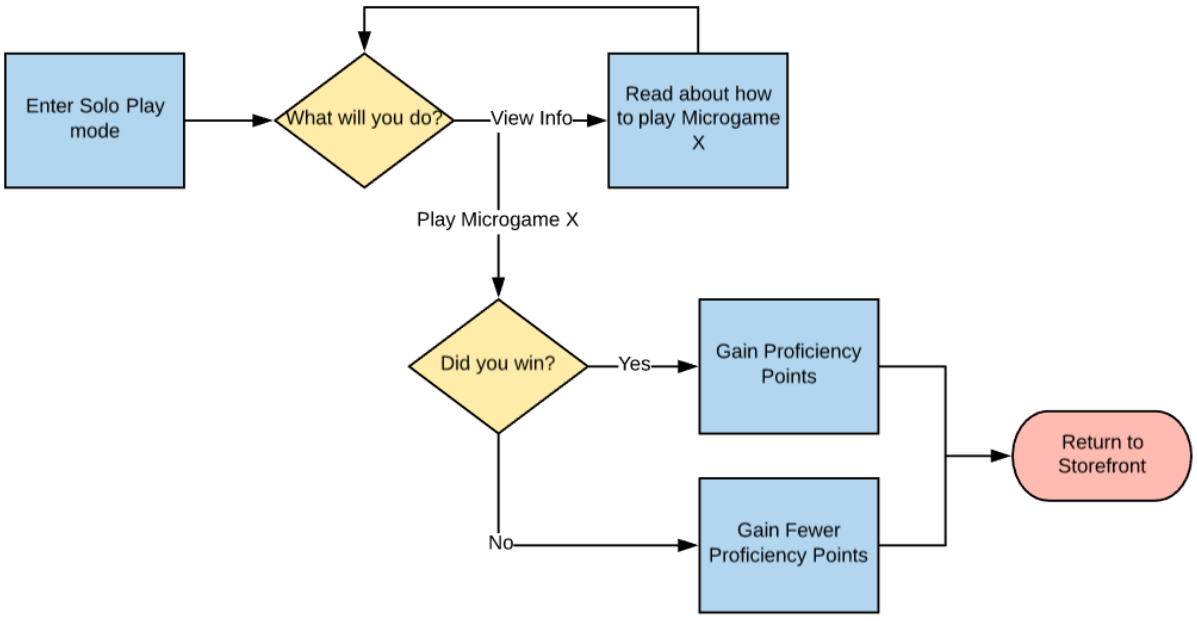



Figure 2.5: A flowchart of the interactions of Solo Play

Figure 2.6 features a screenshot of the Solo Play menu as it is currently implemented. On each row of the shelf a different Microgame is featured. On the left is the aforementioned “Info” button, through which players can view details on the game for that row. To the right is that

game's "Product," which is how it will be represented in the Battle mode. This allows players to get accustomed to associating that picture with the game to improve their recognition when competing against another player. Lastly, in the center, is the button to start the Microgame. From this screen, players can also choose to return to the Storefront.

Figure 2.6: A screenshot of the Solo Play menu

Through continued use of Solo Play, players will accumulate proficiency points depending on their skill. Each microgame has its own proficiency score, which they can increase through continuous, successful play. Having a high proficiency score will cause the associated microgame to appear more often in the multiplayer "battle." This encourages players to continuously play games that they enjoy so that they can have an advantage over their opponents. We discuss proficiency points further in section 2.8.1 Proficiency Points.

## 2.7 Microgames

The term "Microgame" was coined by Nintendo in their development of the *WarioWare* franchise. The term refers to a small minigame that generally lasts fewer than 5 seconds, or 4-16 beats based on WarioWare's beat based timing. These Microgames are the foundation of a play session in all games of the *WarioWare* franchise and they are broken up into two main types: Accomplish and Survival (*WarioWare Inc.: Mega Microgame$!*, 2003). In Accomplish Microgames, the player must complete a simple task successfully in the given time constraint. In Survival Microgames, the player must prevent something from happening until the timer runs out.

Jeff Gerstmann of Gamespot reports that "[*WarioWare Inc.: Mega Microgame$]* tosses its huge slew of often hilarious minigames at you [...] and in the end, it all comes together in a nearly magical fashion" (2003). The quick and dynamic nature of Microgames, as well as their ability to fit together like building blocks to form a single, enjoyable experience was something that was very appealing to us as designers. With the goals of a positive competitive experience and easy expandability of content in mind, we adopted these "Microgames" and made two major changes to better suit the experience we wanted to create: 1) we increased the time limit for each game, and 2) we removed Survival Microgames.

We decided to expand the time limit of each Microgame from the traditional 5 seconds to 10 seconds, for the sake of being able to increase the complexity of each game. Since it was not always guaranteed that a player would be able to challenge an opponent in his area, we wanted players to still be able to enjoy playing *Konbini Konnection* on their own. An increase in the complexity that was allowed for each of our microgames made them more appealing to play casually.

We removed Survival Microgames because they impeded the player's ability to complete them as fast as possible. Our focus was on developing positive, multiplayer experiences, and seeing as the Battle system we decided on was, in its simplest sense, a race for points, we did not want to force players to wait for a timer to run out in order to progress. If there were three games left and the player had the option to complete two of them quickly or one slowly, it would make sense for him to try to complete two games instead of one. With a dilemma like this in mind, it did not make sense to include Survival Microgames.

The design of *Konbini Konnection* includes three Accomplish microgames, loosely inspired by Japanese idioms: Bento, Cats, and Doors.

### 2.7.1 The Bento Microgame

Bento is a fast-paced twitch microgame inspired by the idiom "小打も積もれば大木を倒す" (Shōda mo tsumoreba taiboku-wo taosu), which translates to "Many small strokes fell the great tree." The idiom talks about performing many small tasks to complete one large task, which is the focus of this game. In Bento, players are given a list of ingredients that they need to slice out of the air in order to make a nice Bento Lunchbox.

Figure 2.7: A screenshot from *Fruit Ninja* (Alastor, 2016)

The frenetic, exciting gameplay of *Fruit Ninja* (2010), shown in Figure 2.7, was something that we strove to replicate in designing this particular microgame. In *Fruit Ninja*, fruit is thrown across the screen at various speeds and trajectories, and players are encouraged to slash them for points. Bento was heavily inspired by this idea, but gave players a clear goal that was more akin to an Accomplish Microgame than a High-score Arcade game.

### 2.7.2 The Cats Microgame

Cats is a logic puzzle inspired by the idiom “鳴く猫はねずみを捕らぬ” (*Naku neko wa nezumi o toranu*), which translates to: “A loud cat catches no mice.” Much like the idiom, this game focuses on the player exercising patience in her search for “mice.” Players take on the role of a cat in their owner’s study, and have ten seconds to locate three mice that are hiding on the bookshelf.

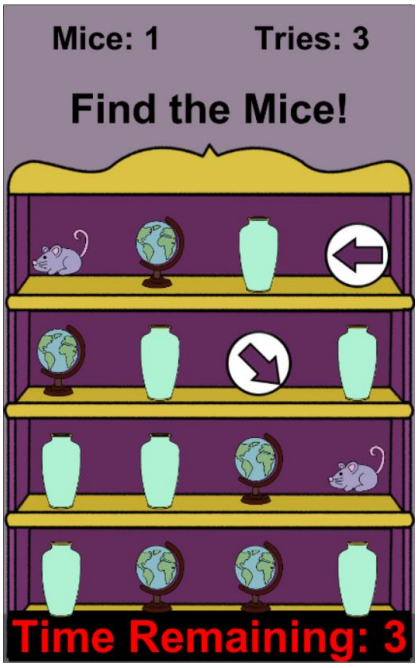


Figure 2.8: A screenshot from the Cats Microgame

Whenever a location is searched that does not contain a mouse, one of the player’s three tries will be removed and an arrow will be revealed, pointing in the general direction of the closest mouse to that location that is currently not found. This can be seen in the top-right corner of Figure 2.8, where the lower arrow is pointing directly to the mouse in the bottom-right. Then, when the higher arrow was discovered, it indicated the location of the mouse that was in the top-left corner. Lastly, over time the player’s tries will regenerate up to 3. This aspect is meant to tie in the patience associated with the idiom. A player with no tries, cannot search for mice and must wait for her tries to regenerate. This general interaction loop is shown in Figure 2.9, in which the area labeled “Player Interaction Loop” encompasses the core game loop; the area labeled “Try Regeneration” follows the regeneration of tries over time; and the area labeled “Game Timer” follows the flow of the Time Remaining display at the bottom of the screen.

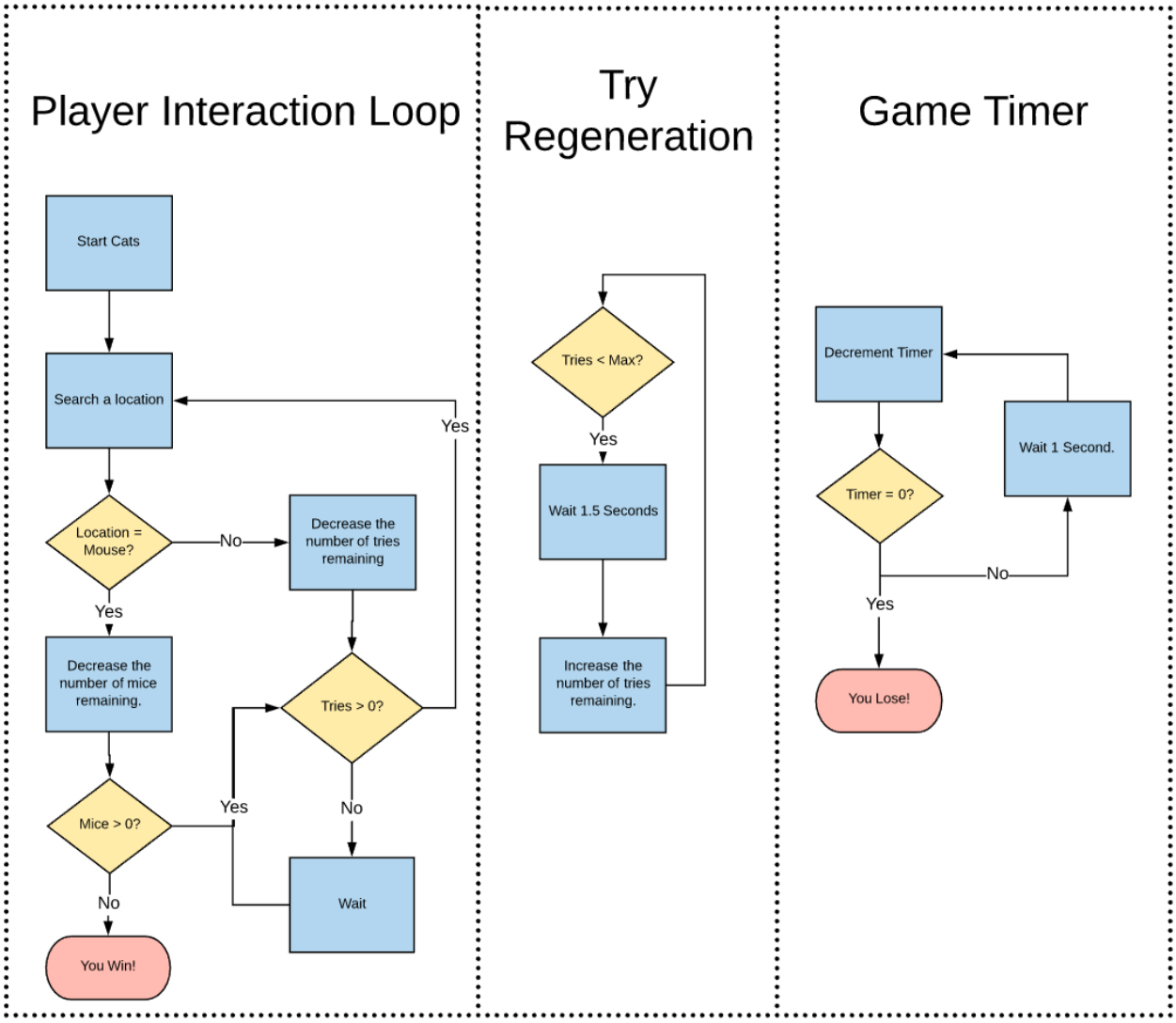



Figure 2.9: A Flowchart of the gameplay for Cats

### 2.7.3 The Doors Microgame

Doors is a twitch-based microgame inspired by the Japanese idiom “押してもダメなら引いてみな” (*Oshite-mo dame-nara hiite mina*), which translates to: “If pushing doesn’t work, try pulling.” The design of Doors is an almost literal interpretation of the idiom, in which players will be shown a series of doors and asked to open them in sequence.

Figure 2.10: A screenshot of the Doors Microgame

Players can interact with the doors by using a "Swipe" motion in one of four directions, representative of different ways that a door can be opened in real life: Push (Up), Pull (Down), Slide Left (Left), and Slide Right (Right). For example, the door in Figure 2.10 indicates that the player should slide it to the left in order to open it, as seen by the handle on the right and the sign on the door. If the player swipes correctly, he will move on to the next door in the hallway, otherwise he will be brought back to the previous door. The player wins when he has opened all of the doors correctly, and loses if he runs out of time to do so.

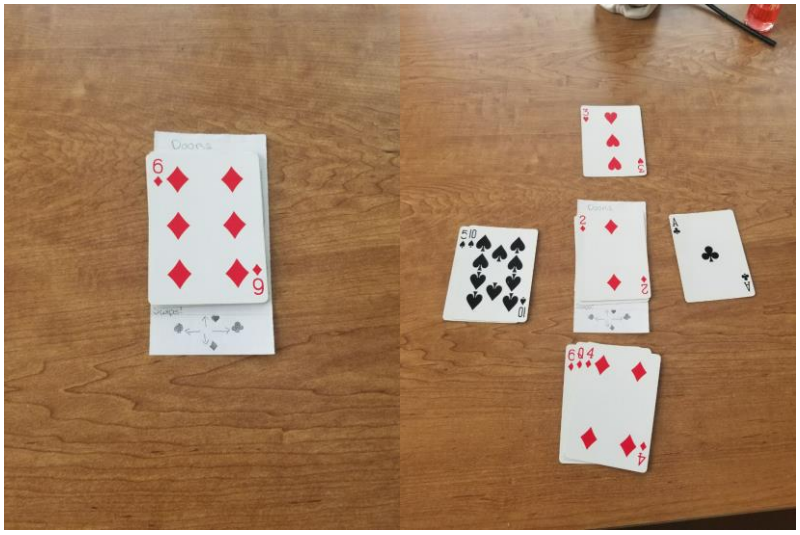


Figure 2.11: The paper prototype for Doors

Doors began as a paper prototype, shown in Figure 2.11, using a deck of cards and a timer. We presented players with a stack of ten cards and a ten second timer, and challenged them to sort all of the cards, one at a time, into four directional piles based on suit. Hearts were collected at the top of the play area, Clubs were placed to the right, Diamonds below, and Spades

to the left. This simple system of play allowed us to visualize the mechanics of the game in action, as well as make some initial thoughts on how to implement the digital game.

## 2.8 Minigames

"Minigames" are smaller games that take place inside of larger games. There are many games that are incredibly popular for their Minigames, such as the *Mario Party* franchise, and *Final Fantasy VII*. These games are typically characterized by players doing a series of very simple actions before the timer runs out in order to complete a goal. In Konbini Konnection, Minigames exist for the purpose of wrapping up a Multiplayer battle with a unique head-to-head experience. In designing the Minigame system, we wanted them to be simple and interesting, while also being clearly separate from the Microgames.

To differentiate Minigames from Microgames, we decided to make two major distinctions as to what defined a Minigame: 1) the length of the game, and 2) they had to build off of the results of the players' session up to this point. One of our main goals in making the Minigames was that we wanted to make them last longer than the Microgames. As a result, we needed to add more complexity to them so that they could hold the player's attention for longer. We did this by adding an advantage system that takes place during the initialization step of the game. In each Minigame, there is some value, such as Stamina, that is variable depending on how well the player did during the Microgame Phase of the battle.

### 2.8.2 Dumplings

Dumplings is a twitch-based Resource Management Minigame inspired by the idiom "花より団子" (Hana yori dango), which translates to "Dumplings rather than flowers" (Most similar to "Function over form" in English). In this game, players go head-to-head working in a garden that grows two types of plants: Edible (Dumpling) Plants, and Flower Plants.

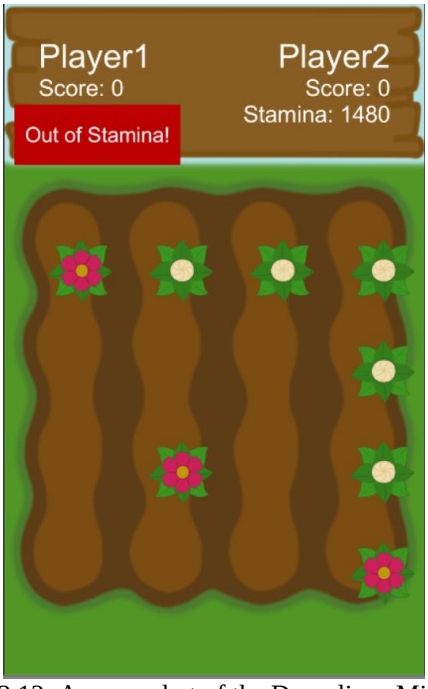



Figure 2.12: A screenshot of the Dumplings Minigame

Plants will quickly grow through three stages as the game progresses: sprout, bud, and bloomed. Figure 2.12 shows a play session in which the field is in full bloom, with the maximum of eight plants. Players gain points by collecting the bloomed flowers that grow in the field. The player with the most points at the end of the game wins.

Players begin with a variable amount of stamina based on how many Microgames they had completed in the Microgame Phase (200 Stamina per Microgame completed successfully). This stamina level decreases over time and eventually hits 0, causing the player to be eliminated and unable to participate further. When this occurs, the “Out of Stamina” sign appears over the player’s stamina total. In order to combat this however, players can choose to pick and eat the Dumpling plants in order to regain some of the energy that they lost and keep picking flowers. These interactions culminate into a model similar to the one displayed in Figure 2.13.

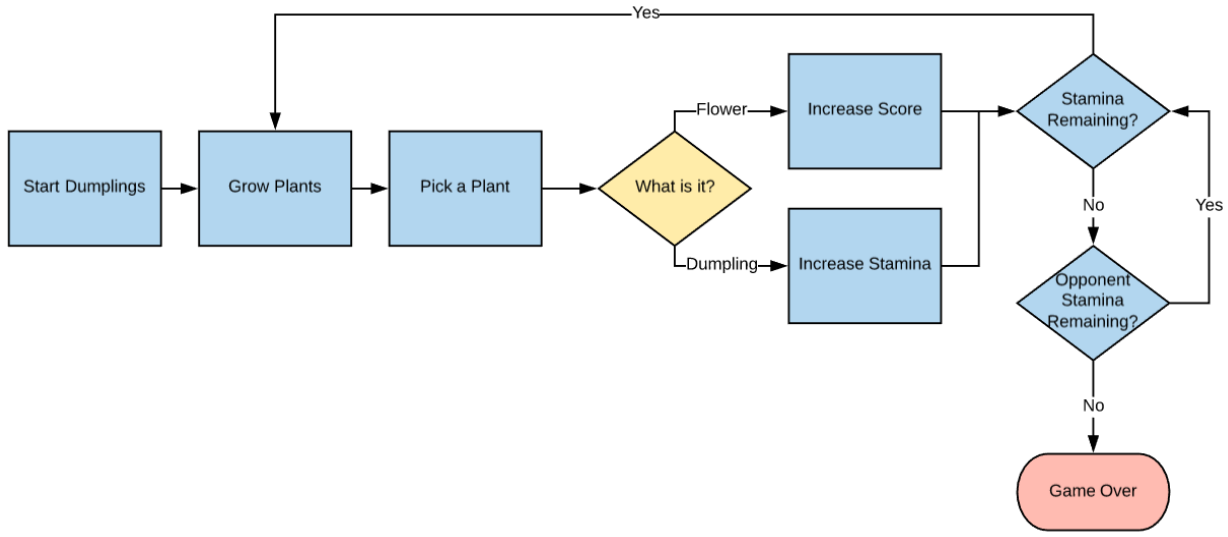



Figure 2.13: Basic game loop for Dumplings

The goal when designing this game was to encourage players to occasionally forsake picking flowers to instead pick dumplings. We did this by increasing the importance of dumpling plants, and decreasing the frequency at which they appear. Since stamina is a crucial element to being able to participate in the game, having the collection of dumplings be the only way that players can regain stamina made them instantly desirable. On top of this, we lowered the spawn rate of dumplings to slightly below that of a flower and made them finite. After the 50th dumpling was spawned, players would no longer be able to collect dumplings. This discouraged the strategy of players collecting as many dumplings as they could early on and just trying to outlast their opponent, and instead made the choice to pick a dumpling more conscious and meaningful.

## 2.9 Feedback Systems

In order to motivate people to play *Konbini Konnection* and engage in social interactions, we needed to supply them with extrinsic motivators. In order to do this, we came up with a set of positive reinforcement models that encouraged players to interact with every part of the system. The important part of designing these systems was that even though they explored all sections of

the game, in the end they needed to relate back to creating a cohesive, positive social experience for players. An overview of the feedback loop used for *Konbini Konnection* is seen below in Figure 2.14.

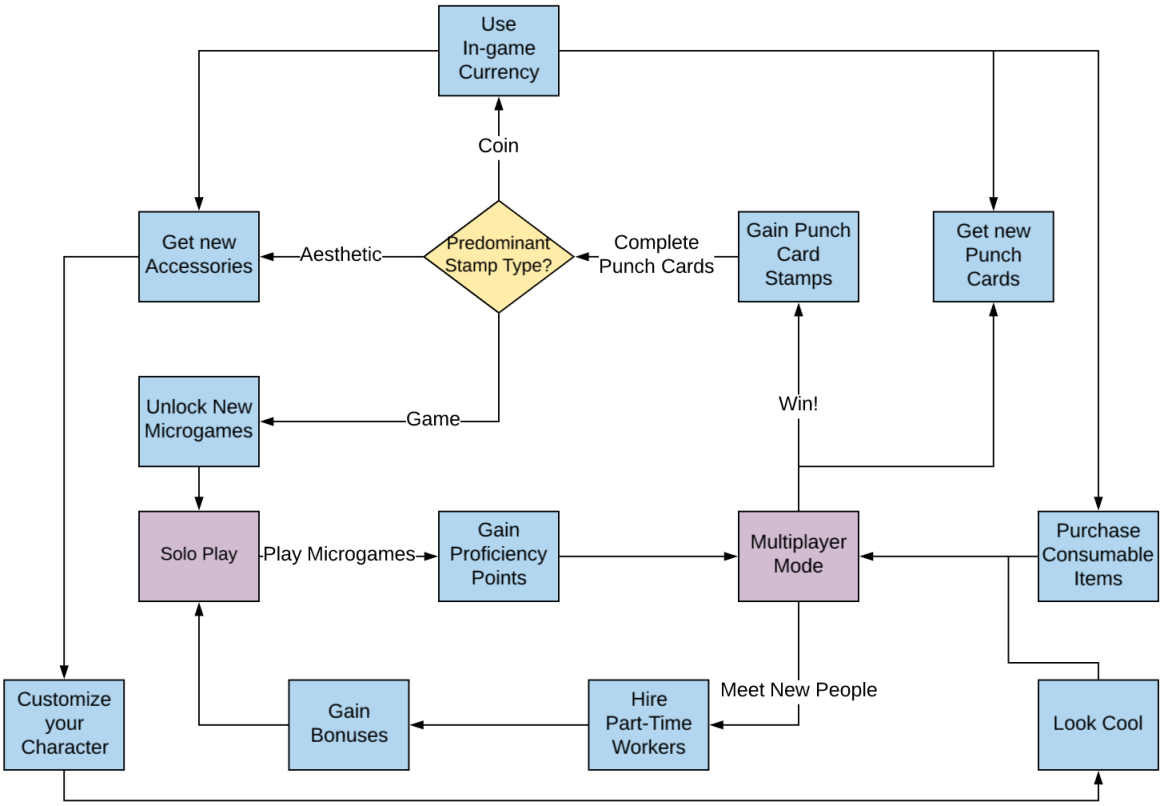



Figure 2.14: *Konbini Konnection*'s Feedback Loop

## 2.9.1 Proficiency Points

Even though *Konbini Konnection* is meant to be a social multiplayer game, it is inevitable that players may occasionally have no opponents nearby. This is the reasoning for the design of our Solo Play mode. However, we wanted players participating in the Solo Play mode to receive some kind of in-game bonus for continuing to play even though nobody is around. For that reason, we implemented a Proficiency Point system.

Inspired by Role Playing Games (RPGs) and Gamification concepts, our Solo Play mode allows you to gain tangible advantages in the Battle mode through continued play. As players complete microgames in Solo Play, they gain Proficiency Points, which act similarly to Experience Points in an RPG. Each Microgame has a different Proficiency Point total associated with it, allowing players to see how often they play each game.

These points tie into the Battle mode by altering the probability that certain games will appear. At the start of the battle, each player will be assigned 50% of the probability. Then, all of their Proficiency Points will be totaled and divided out into that 50%. Finally, the two sets of probabilities will be brought together and used to determine the nine items that appear at the checkout counter for that battle.

### 2.9.2 Part-Time Workers

Since *Konbini Konnection* is meant to encourage social interactions with players outside of the game world, we decided that we wanted to implement a way for players to benefit from meeting the same players multiple times. Inspired by *Streetpass Mii Plaza* (2011), our game allows the player to employ other users that she has met to work in different sections of her convenience store. Based on the level and classification of the player, she will gain different benefits, such as increased Proficiency Point gain when playing certain Microgames, or increased Coins obtained at the end of a battle. Similar to the games of *Streetpass Mii Plaza*, every day that a player meets their Part-Time Worker, the social bond between the two of them will level up and increase the bonus that they give each other, up to a maximum level of 10.

### 2.9.3 Punch Cards

The Punch Cards in *Konbini Konnection* are the main method of providing players rewards after the Battle. Players can hold up to 3 Punch Cards that contain either 3, 5, or 8 spaces. They can receive these Punch Cards either through the Supplier, or via random drop after a battle. When a player wins a battle against another player for the first time that day, he will receive a "Stamp" on his active Punch Card based on the opponent's classification. Then, if the Punch Card is full, it will be instantly exchanged for a reward based on the classification of stamps on it. An example design of a Punch Card is seen in Figure 2.15.

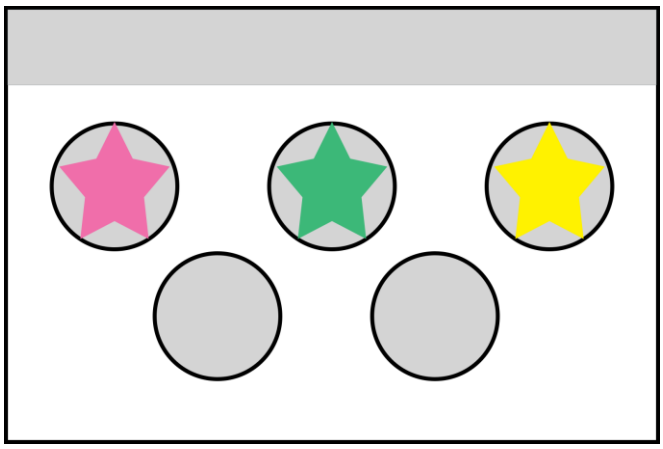


Figure 2.15: A Mockup of a Punch Card

There are three different classifications of players: "Game," "Coin," and "Aesthetic." These classifications are determined by the player upon the creation of his character, much like the Team System in *Pokemon Go* (2016). This decision is relevant to the player because every day he may choose to stamp one of his Punch Cards using his own team's stamp as a daily reward. This idea of player classification was largely inspired by *Streetpass Mii Plaza* (2011), where, depending on the color of the avatar's shirt, its role in the minigames would change.

When a player fills a Punch Card, he will be given a reward based on its contents and size. If a majority of the Stamps on the card are of a certain classification, the reward will be of that type. In the event of a tie, the reward type will be chosen at random between the tied classifications. Then, the size will determine the rarity or quantity of the reward depending on what it is.

Players who fill out their Punch Cards with “Game” (Green) stamps will make progress towards unlocking more microgames to play in Solo Play. They obtain a special form of currency called “Tokens,” which can be used to obtain new products that can be sold in their store. The use of unlockable products allows players to become comfortable with the microgames at their disposal before having to learn more. The size of the Punch Card will determine the number of tokens that are received upon its completion.

Players who fill out their Punch Cards with “Coin” (Yellow) stamps will obtain in-game currency that can be used at the Supplier. They can use this currency to purchase a variety of goods that can assist them in the battle mode, or otherwise improve their experience. The details of the Supplier are outlined in section “2.8.4 The Supplier.” The size of the Punch Card will determine the amount of currency that is received upon its completion.

Players who fill out their Punch Cards with “Aesthetic” (Pink) stamps will gain access to new items that they can use to customize their character. These accessories have no impact on the actual gameplay, but give players more options to express themselves in the game space. When an “Aesthetic” Punch Card is completed, the reward is selected from one of three prize pools: common, uncommon, and rare. The goal of this system is that common accessories would have relatively less appeal than the rare ones, so that players are encouraged to fill out larger Punch Cards and play a part in more social experiences.

### 2.9.4 The Supplier

“The Supplier” is the in-game shop where players can purchase a variety of items to enhance the experience of running their very own convenience store. In this shop, players can use in-game currency obtained through either “Coin” Punch Cards or Microtransactions to buy Punch Cards, Consumable Items, or Accessories. All three tiers of Punch Cards (3, 5, and 8 slots) are available for purchase, but can only be bought if the player has an available space to hold them. The Consumable Items are usable in the Battle mode of *Konbini Konnection*. They allow players to affect the items brought to the checkout counter, such as applying “Discounts,” which decrease the amount of time both players have to complete the microgame.

The Accessories are obtained using a Gacha Machine instead of being purchased outright. This model is meant to mirror the popular system that appears in most social mobile games, where players can spend currency to get a chance of obtaining new and rare gameplay elements (Toto, 2012). The Gacha system in mobile games has been proven to be extremely successful, as many companies have reported that over 50% of their income comes from players spending money to turn the wheel.

## 2.10 Proof of Concept

Since the full extent of our vision for *Konbini Konnection* was too large for the scope of this project, we were only able to produce a portion of the original design. Implemented in this proof of concept is:

- Solo Play Mode, without the Proficiency Point system
- The Doors and Cats Microgames
- The entirety of the Battle framework, minus Consumable Items
- The Dumplings Minigame

This excludes most of our designed feedback system. The following sections discuss the Artistic Design, Technical Design, and Testing with regards to this proof of concept

# 3. Artistic Design

*Konbini Konnection* uses 2D art assets that were made using Adobe Photoshop. The following sections detail the process in which they were made, from the inspiration to the finalized assets.

## 3.1 Overall Style/ Inspiration

Japan's cultural, economic, and political dynamics have shifted over the past three decades, turning Japanese pop culture into a global phenomenon. Contemporary Japanese pop culture embraces a youthfulness and is appealing to people globally (Shearin, 2011). This cuteness is booming around the world as a fashion element and is rapidly becoming the global image of Japan (Kageyama, 2006). The cuteness culture of Japan, or kawaii aesthetic, emerged out of university students in the late 1960s protesting against what was being taught by reading children's comics instead of going to lectures (Kerr, 2016). Kawaii is the romanization of かわいい, which translates to cute. Kawaii has remained a symbol of resistance as it has become a way of resisting the adult world and denying female sexuality and all the subjugation it implies. It is everywhere and claimed by everyone, regardless of age, gender, and nationality (Kerr, 2016).

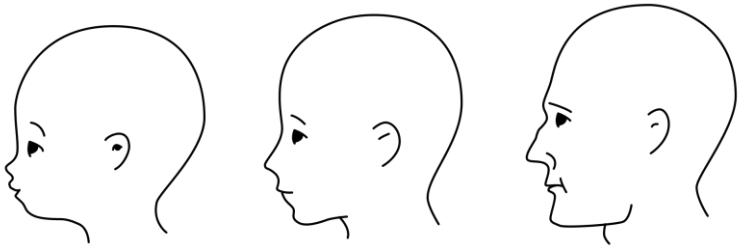


Figure 3.1: Regression of Kindchenscheme (Neitram, 2007)

Cute things are typically popular because they create positive feelings in people (Nittono, Fukushima, Yano & Moriya, 2012). Cuteness is a subjective term that describes a type of attractiveness that is associated with youth. Konrad Lorenz was the first to introduce cuteness as a scientific concept and analytical model in ethology. He proposed the concept of Kindchenschema, or baby schema, as a set of features that can commonly be seen in young animals such as a large head relative to body size and large eyes (Lorenz, 1970). Figure 3.1 illustrates the regression of the baby schema as one ages. This schema acts as a stimulus to capture attention and induce motivation to care for the thing deemed cute. A study conducted in 2012 found that a cuteness-triggered positive emotion caused a narrowed attentional focus (Nittono, Fukushima, Yano & Moriya, 2012). The motivation for using a cute art style is that it could increase player motivation and positive feelings.

Two games played a major role in the inspiration of the art style of *Konbini Konnection*, *WarioWare, Inc.: Mega Microgame$!* and *Rhythm Heaven*. Ko Takeuchi was the art director for both games and has a style that is commonly seen as cute. He is best known for simple designs that use bold lines (Bertoli, 2016). Both *WarioWare, Inc.: Mega Microgame$!* and *Rhythm Heaven* use Takeuchi's simple designs paired with bright colors. These games tend to have

objects in the foreground, such as characters, outlined in black, while the background is done in basic colors with simple shading, as shown in Figure 3.2.

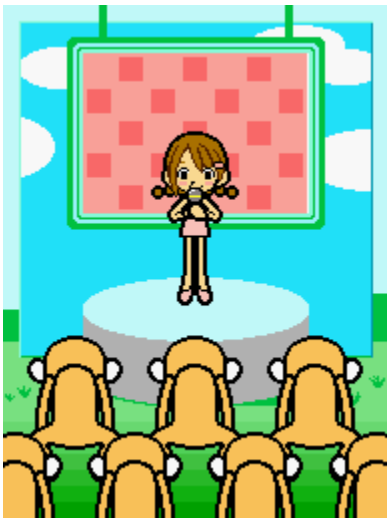


Figure 3.2: Rhythm Heaven Fan Club Game (WarioWare Inc.: Mega Microgame$, 2003)

## 3.2 Storefront

The narrative of *Konbini Konnection* is that the player takes control of their own konbini and manage customers by playing games. Konbini is the romanization of コンビニ, which is the shortened version of the Japanese word for convenience store. Convenience stores in Japan are often open 24 hours a day and offer more services than typical western convenience stores, such as payment services for bills and ticket services for concerts to airlines (Convenience Stores, 2017). This narrative was chosen because of how many there are around Japan and the variety of elements offered from them.

Areas where the player is navigating between the closet and gameplay are called the store. Within the store, the area where the player is sent after logging in or exiting the closet is called the storefront and this is where other nearby players can be seen. The other area of the store is the solo play area and this is where players can practice microgames. These areas of *Konbini Konnection* are where the narrative of running a konbini are visually present. Konbini stores use bright florescent lights which highlight cooler colors, as they produce light that is blueish (Smith, 2016). The use of these lights influenced the use of cool colors for the color pallette of the store. The row based layout of these stores, as seen in Figure 3.3, inspired the use of a row layout in the storefront and battle system (Convenience Stores, 2017).

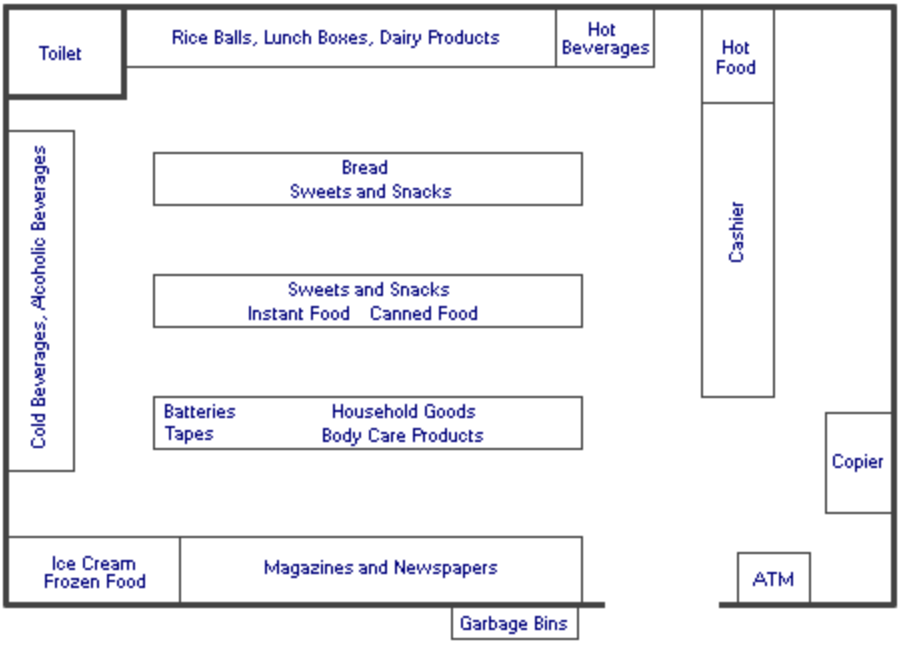


Figure 3.3: Layout of a Typical Konbini (Convenience Stores, 2017)

## 3.3 Character Creation

In order to allow for not only a personal connection between players, but also a more personal connection between the player and the game itself, players are able to create their own mascot character from predetermined assets. This gives players the opportunity to show off their personalities and get to know other players by having characters that represent themselves. The goal of the character creation system is to allow the player to create a ゆるキャラ or yuru-kyara for their Konbini.


Figure 3.4: Kumamon on household products (Lavendei2, 2015)

Yurui masukotto kyarakutā, or its contracted form yuru-kyara, is a term used to represent a species of mascot in Japan and roughly translates to mean “light character.” This term is believed to have been coined by illustrator Jun Miura back in 2002 (Brassor, 2008). Yuru-kyara are created with the goal of attracting attention, as they are used to promote everything from regions of Japan, soap, to train lines and even prisons (McKirdy, 2014). These characters have had a presence in Japanese advertising for decades and generated nearly $16 billion in sales in 2012 (Ripley & Henry, 2014). The popularity of some yuru-kyara is spread beyond the territory they represent as Kumamon, a yuru-kyara for the Kumamoto Prefecture, has the status of a celebrity throughout Japan, appearing on everything from posters advertising Kumamoto to household goods, as seen in Figure 3.4 (McKirdy, 2014). Part of the appeal of yuru-kyara is the amateurish nature of their concepts and designs as out of the several hundred of them throughout Japan, most were designed and chosen by residents. Getting to play a role in their creation makes people feel closer to these characters (Brassor, 2008). Having players create their own yuru-kyara for their konbini will allow for them to feel closer to the character that represents them to others in the game.

Figure 3.5: Design Process of Character Creation Parts

The process of designing a character creation system shown in Figure 3.5 started with creating base characters that could be broken down into parts and allowing players the ability to mix and match those parts to create their characters. The proportions of the base characters were based on yuru-kyara, notably Nobu-sama and friends. Since each piece needed to be able to fit together without clashing, the base characters needed to be somewhat similar but still visually distinctive. Three animals similar in size were chosen for reference and then drawn in a yuru-kyara style to create the base characters.

Figure 3.6: Tanuki figurine (Hall, 2006)

The search for animals started by looking through animals that play a role in Japanese culture. When walking around Japan, figurines of the pudgy creature seen in Figure 3.6 can be seen outside businesses, beckoning for patrons to enter. These figurines are of an animal known as a tanuki and are a modern depiction of them with a big belly, straw hat, giant scrotum, sake flask, and confused facial expression (Schumacher, 1998). These figurines are meant as a prosperity charm to stretch one's money and bring good fortune, as metal workers in the Kanazawa Prefecture would wrap gold in the scrotum of a tanuki before hammering it into sheets that were supposedly thin enough to cover eight tatami mats (Gordenker, 2008).

A tanuki, or a Japanese raccoon dog, is both an actual species of animal in Japan and a Yokai known for shape-shifting. Yokai are Japanese supernatural beings or phenomena that are not worshipped. The Tanuki is a mischievous Yokai, illustrated in Figure 3.7 by Toriyama Sekien, that is notorious for causing people to lose their way in familiar places by altering features of the landscape (Foster, 2008). Tanuki are a part of Japanese pop culture, appearing in video games such as Super Mario Bros. 3, and Animal Crossing; however, they are often mistranslated as raccoons in exported versions (West, 2009).

Figure 3.7: Yokai Tanuki (Foster, 2008)

Figure 3.8: Tanuki Base Characters

Their popularity in Japanese culture lent the tanuki to be the first animal selected for the character creation system. When designing the tanuki as a base character, the build of the character was decided to be between the chubby build of the figurine and the slimmer actual animal. The color palette was picked to be similar to that of the figurine tanuki and the ones used for depictions of Tanuki in pop culture.

Within the Kinmozui, the first illustrated encyclopedia in Japan, the tanuki shares a page with the Kitsune, or fox. Similar to a tanuki, a kitsune is both an actual species of animal found in Japan and a Yokai. However, their page in the Kinmozui fails to mention any supernatural characteristics for both animals, categorizing them with animals that are similar in appearance. As a Yokai, a kitsune is known for shape-shifting and a form of possession called “fox possession,” which is the oldest recorded and most common found in Japan (Foster, 2008). A major difference between a tanuki and a kitsune in Japanese lore is that a kitsune has a divine connection (Schumacher, 1998). Kitsune had an unavoidable presence in religion and culture in Japan from the ninth century to the nineteenth century. The cultural and religious significance is difficult to capture, like the animal itself. In modern language, kitsune seems to be synonymous with *Vulpes vulpes* or the red fox (Bathgate, 2003).

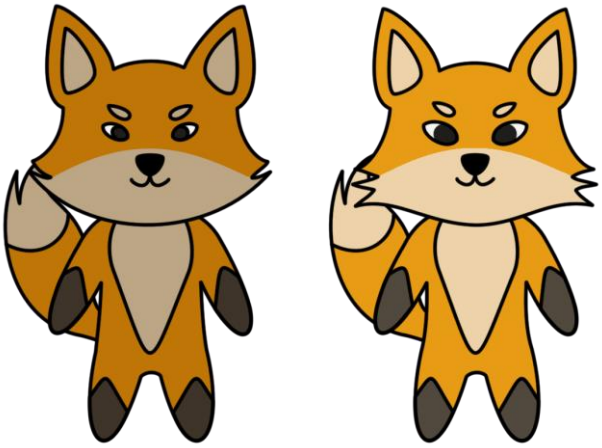


Figure 3.9: Kitsune Base Characters

The cultural significance of Kitsune, along with their similarity in proportions to a tanuki, made them a good choice for the second base character. Since the word “kitsune” can mean “red fox,” it was used for the design of the second base character shown in Figure 3.9. While it is called a red fox, the color palette for this character is a yellow-orange to match the color of an actual red fox shown in figure 3.10.

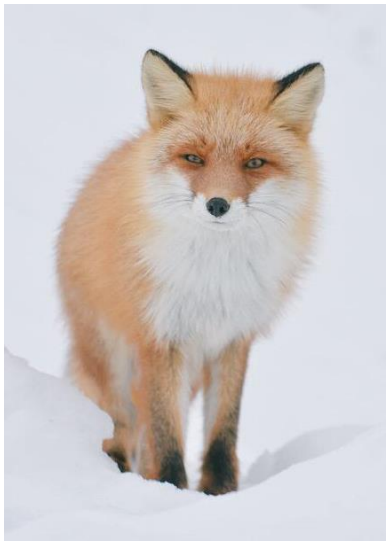


Figure 3.10: Hokkaido Red Fox (Zhang, 2015)

The final animal was chosen using the limitations set by the first two animals, such as size, color, and the ability to be broken down into three parts. Other animals that were native to Japan were tried but did not fit together as well as needed. Instead of forcing another native Japanese animal to fit as the third animal, the search turned to looking at animals that are similar to the two. Since they are both members of the Canoidea suborder, looking through a list of animals in that suborder started this search (Carnivore, 2018). The red panda was found through this search.

Figure 3.11: Red Panda (Beardsley Zoo, 2018)

The *Ailurus fulgens,* or red panda, is a reddish brown, long-tailed mammal shown in figure 3.11. When searching through members of the Canoidea suborder, the red panda stood out, as it is the only member of the Ailuridae family (Red panda, 2018). It was chosen as to be used as a base character because of its similarity in size to the tanuki and red fox, similar color palette to the red fox, and ability to be broken down into a head, body, and tail component.

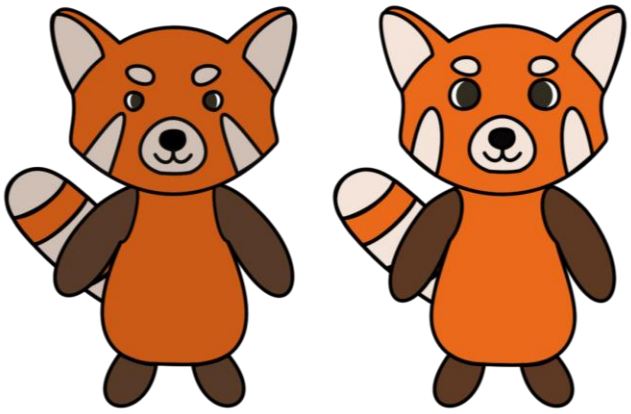


Figure 3.12: Red Panda Base Characters

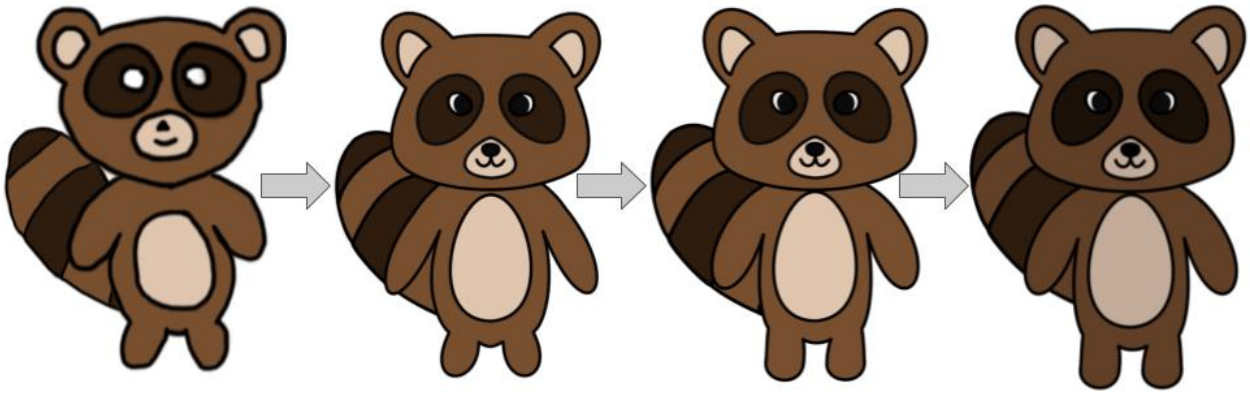


Figure 3.13: Tanuki Iterations

Every character part went through multiple iterations before the final was created as shown in figure 3.13. The first changes were the adjusting of head shapes and proportions

between the body and limbs. These changes provided the head shape but left the arm length too long. Changing the arm length and adjusting proportions between the head, body, and tail was the focus for the next iteration. The changes made to provide the final version of each base character was the adjustment of color palettes to better distinguish between the male and female looking versions.

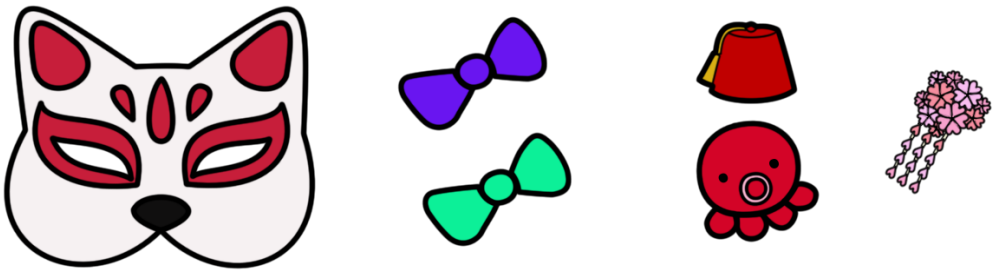


Figure 3.14: Accessory Options

Figure 3.14 shows all of the options for accessories in the character creation system. These accessories range from simple objects such as bows, to a half-face kitsune mask. Originally, the kitsune mask accessory was meant to be unlockable after winning a certain number of games. This was cut because not enough time to implement an unlockable system. Rather than not including the kitsune mask in the game, it was downgraded to a starting accessory.

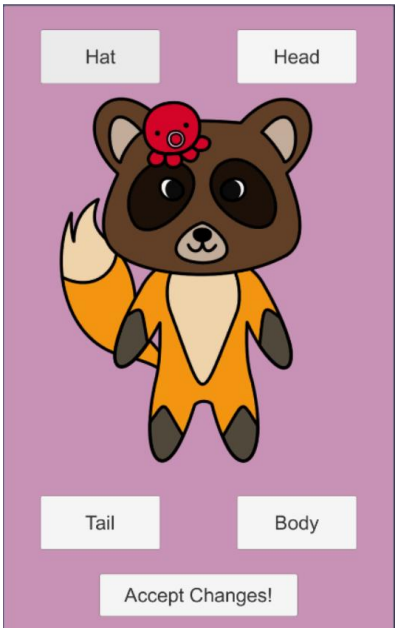



Figure 3.15: Character Creation System

All together the character parts and accessories allow for players to create their own yuru-kyara using the system shown in Figure 3.15. This system has six different options for each part, allowing for a possible 1,296 unique combinations.

## 3.4 Games

Each sub-game has a different art style as a tribute to the *WarioWare* franchise. The microgames within these games have different art styles based on the designer (Alfonso, 2006). These styles branch off of the overall art style of *Konbini Konnection* and help to visually distinguish that each game is different with a distinct set of rules.

### 3.4.1 Doors

The microgame Doors has various door assets that indicate which way the player has to open it. Each door has a sign and visual hints to tell the player which way to swipe it. This microgame follows the overall art style by using bright colors and a black outline, but it varies by having a textured outline, giving the art a childish feel. Doors are different colors for different directions to avoid confusion as shown in Figure 3.16. The signs are done in the complementary color of the door to make it easy to see, allowing players to quickly react.

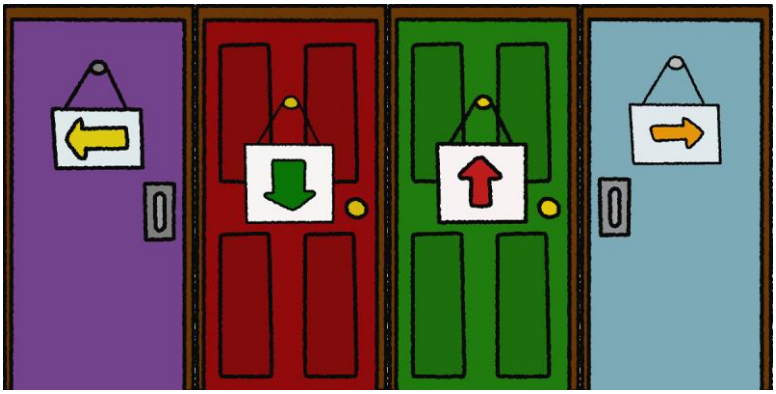


Figure 3.16: Doors' Different Doors

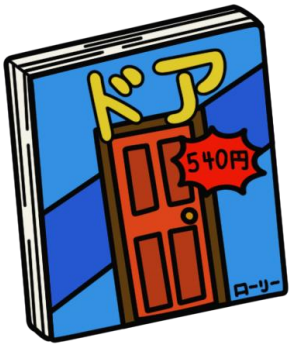



Figure 3.17: Doors Icon

Doors also needed an icon to represent it to players within the battle system and needed to be done in the overall style of *Konbini Konnection* to match with it. Figure 3.17 shows the icon created for Doors which is a magazine about doors. The magazine appears to open the wrong way but this design was intentional as magazines and books in Japan are read right to left.

### 3.4.2 Cats

The microgame Cats is set in a fancy house as the player takes the role of a house cat to hunt mice that have hidden themselves behind objects on a bookshelf. The art style of this microgame shown in Figure 3.18 uses a black outline similar to the overall art style, but uses rough edges on the outline and duller colors. The outline style was inspired by *Neko Atsume* (2014), a game that has a simple goal of collecting cats.

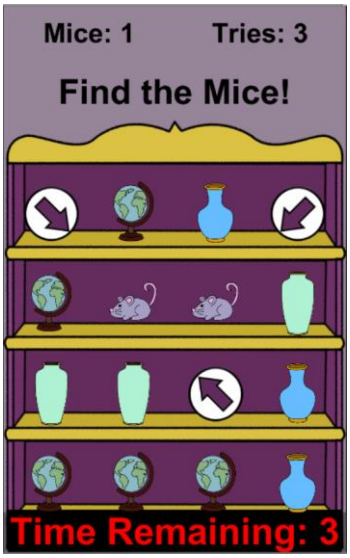



Figure 3.18: Cats Gameplay

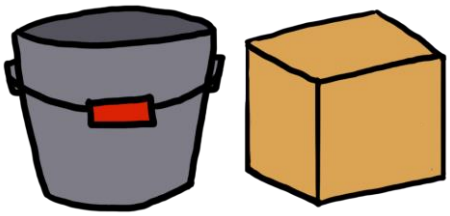


Figure 3.19: Storage Room Objects

The original setting for Cats was going to be a storage room with the mice hiding behind objects such as boxes and buckets. After creating objects for this setting (shown in Figure 3.19) it was decided that the setting would give Cats a dull feeling and that changing the setting would give Cats a more whimsical feeling. A concept sketch shown in Figure 3.20 was created to show off the setting of a fancy house. In this setting the player, taking the role of a cat, has to find the mice that are hiding behind objects on a bookshelf. The player is given a reason for the limited number of tries that regenerate over time in this setting as they must behave for their owner. Switching the setting from a storage room to a bookshelf in a fancy house allowed for the incorporation of more colors which helps give Cats a more whimsical feeling.

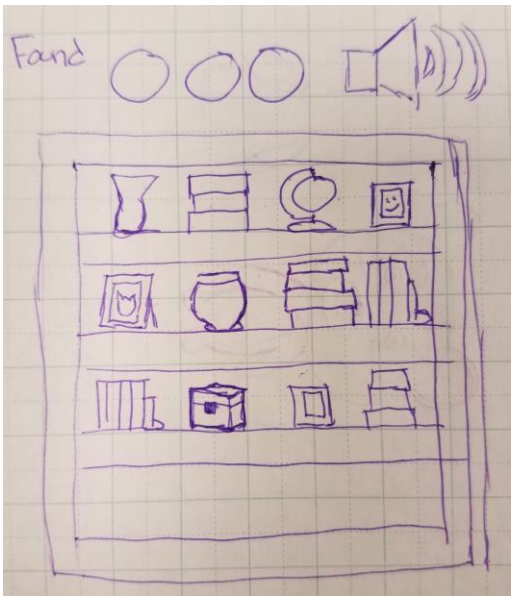



Figure 3.20: Concept Sketch of Cats


Figure 3.21: Cats Icon

Since Cats is a microgame, it has an icon to represent it during battles. Shown in Figure 3.21 is the icon created for Cats which is a bag of cheese chips with a picture of a mouse on it. The icon is done in the overall style of *Konbini Konnection* as the battle system which it appears in uses that style.

### 3.4.3 Dumplings

Dumplings is a minigame where players compete against each other by collecting dumplings and flowers in order to get the most points. In order to differentiate Dumplings as a minigame it has an art style that varies the most from the overall style to distinguish it from the microgames. The style shown in Figure 3.22 does not use a black outline like the overall style and has simple shading to give it a more natural feel.

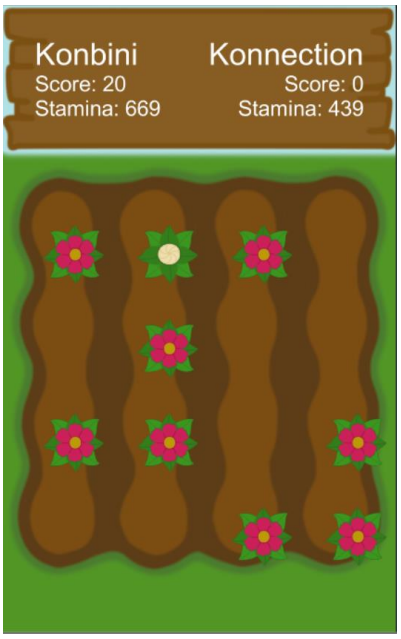


Figure 3.22: Dumplings Gameplay

The background for Dumplings is a garden plot of columns for the flowers and dumplings to grow. At the top is a wooden sign to display information about the garden to the players. Space was left at the bottom to add a key for players to guide them in their harvest. This was not implemented because of time constraints.

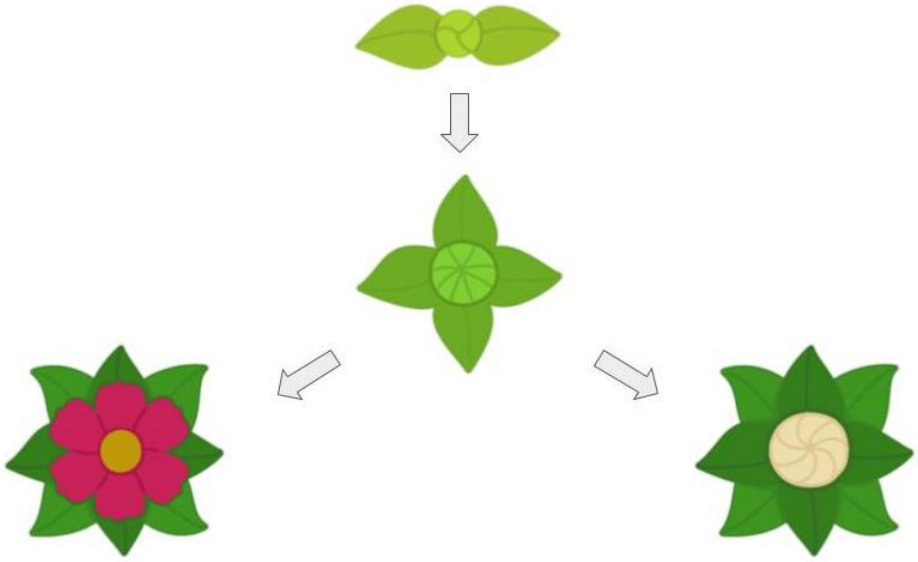


Figure 3.23: Plant Stages

Figure 3.23 shows the various stages of the plants, going from the sprout at the top to either a flower or dumpling shown at the bottom. The dumpling plant on the right shows an overhead view of a steamed dumpling emerging from inside leaves to create this kind of plant. Shown on the bottom left is the flower which is a 椿 or camellia, a flower that has a long tradition in China and Japan (Joly, 2017).

# 4. Technical Design

We developed *Konbini Konnection* using Unity 2017's 2D game engine for devices equipped with at least Android 4.1 'Jelly Bean.' In order to allow players to interface with each other, we employed concepts of password encryption, database structuring, location services, multi-threading, and socket programming. To manage data collection for testing, we defined analytic metrics through which we could understand how players played the game. Lastly, to manage gameplay, we utilized a variable timestep game loop in tandem with gestural inputs to provide a complete, tactile experience. This section discusses the various technical challenges that we ran into, as well as how we approached their solutions.

## 4.1 Login System and Password Encryption

Our game requires us to identify players so that we could access their information across multiple sessions in the event that they wanted to change the device that they were playing on. To do this, we determined the best course of action was to develop an Account System in which users could create their own username and password. Players could sign-in to *Konbini Konnection* with a new account or a pre-existing account, the results of which are shown in Figure 4.1. Using a new account to sign in would create basic information for the player, add him to the database, and bring him to the Character Creation screen. Using a pre-existing account would load in all of the user's data and bring him to the Storefront screen.

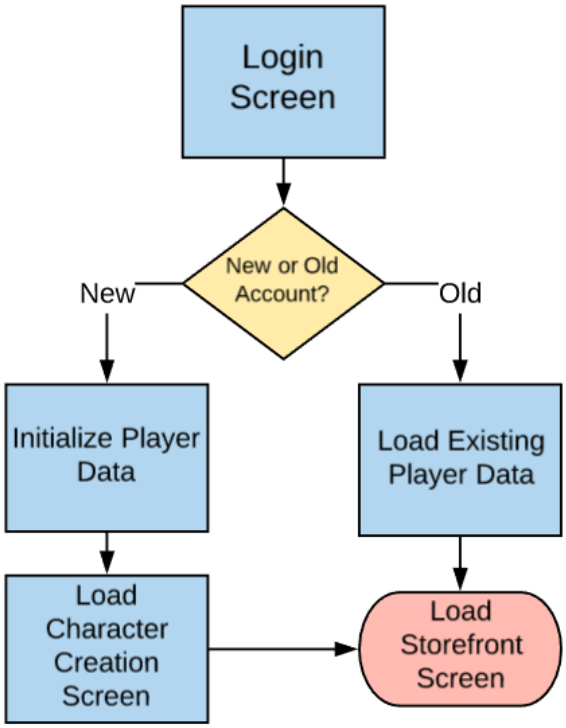



Figure 4.1: Login System flowchart

We wanted users to feel secure about using passwords that they would recognize and remember, which led us to employ PBKDF2 with SHA1 encryption. Other encryption methods, such as BCrypt and SCrypt are known to be more secure, however Unity's native support of PBKDF2 (Dustin-horne, 2013) made it a more desirable solution.

```
01.  function hashPassword(password, saltLength = 32, salt = null){
02.    //Input:  A password: password.
03.    //        An integer to determine the length of the bit array: saltLength.
04.    //        A byte array representative of the salt, if it exists: salt.
05.    //Output: A byte array containing the generated hash
06.
07.    byte [] hash;
08.
09.    if(salt == null)
10.      salt = generateSalt(saltLength);
11.
12.    for(i = 0; i < 10000; i++)
13.      hash = hashingAlgorithm(password + salt);
14.
15.    return hash;
16.  }
```

Code Listing 4.1: A basic password hashing function

Regardless of the encryption algorithm, the theoretical process of using it is typically the same. *Konbini Konnection*'s hashing function uses this same base structure, represented in Code Listing 4.1, in which the "hashingAlgorithm()" used is PBKDF2, outlined by OWASP (2015). It begins by generating a random array of bytes, called a "salt," which will be appended to the end of our password before we hash it. This allows for two passwords to be the same yet appear different based on the account that they are attached to. The system used to communicate with the database that holds this information may require salts containing certain characters to be filtered out. We then hash the password-salt combination a predefined number of times using our selected encryption algorithm and return the result.

This hashing system will always return the same hash when given the same salt. Therefore, if we wanted to query the database of a password, we would first need to obtain the salt from the database. We can then pass that salt into our hashing function instead of generating a new one and compare the result.

## 4.2 Databases

Some challenges of this project included figuring out a way to collect player data to allow for movement between devices and collection of game analytics. The solution we used for this problem was to create databases using MySQL to hold player information and analytic data. The following sections detail the platforms used to set up the databases and the creation of each.

### 4.2.1 PHP & MySQL

The original plan was to host the databases through WPI's MySQL server. This plan was changed due to slow speeds while working with it in Japan and the inability to host PHP files on WPI's network. The team decided to instead host both the databases and PHP files through a web service called 000webhost for the sake of speed and simplicity. However, 000webhost

added a challenge as they would update their interface frequently and put the website hosting our PHP files to sleep for an hour each day, making the game unusable at these times. Hosting PHP files was necessary because, when developing for android, Unity is not able to connect to a MySQL database directly to send queries, as MonoDevelop does not implement all of the .NET libraries necessary. Avoiding platform specific programming by encapsulating SQL queries in PHP files was easier as most devices can send GET requests. While one team member had worked with SQL databases before, neither had worked in PHP before. A tutorial detailing how to set up a scoreboard was used to understand the basics. Figure 4.2 shows how the interactions between Unity, PHP, and MySQL happen.

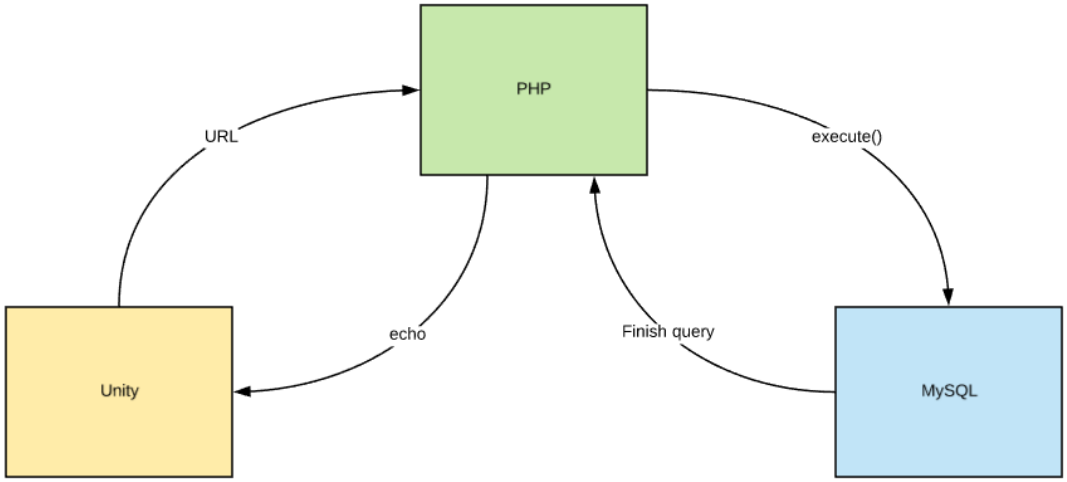



Figure 4.2: Unity-PHP-MySQL interactions

Unity starts these interactions by using the WWW class to send the parameters needed to retrieve or add to the databases to the appropriate PHP file through a URL to where the file is hosted. The URL is created by taking the base URL to the PHP file and combining it with the variables needed in the SQL queries by using the variable name used in the PHP file and setting it equal to the information to be added to the database as shown in the StartBattle function in Code Listing 4.2. The PHP file then takes the parameters and uses them in queries to the appropriate database when executed. We use the PHP Data Objects (PDO) extension as an interface for accessing databases in PHP. PDO was chosen over other extensions, such as MySQLi, as PDO currently supports 12 different drivers, providing a transparent process if the databases needed to be changed (Marjanovic, 2012).

```
01.  public void StartBattle(int sPlayer, int cPlayer)
02.      {
03.          StartCoroutine(_StartBattle(sPlayer, cPlayer));
04.      }
05.      private IEnumerator _StartBattle(int sPlayer, int cPlayer)
06.      {
07.          string startBattleURL="https://kkmqp.000webhostapp.com/startBattle.php?";
08.
09.          WWW startBattle=new WWW(startBattleURL+"splayer="+sPlayer+"&cplayer="+cPlayer);
10.
11.          yield return startBattle;
12.          KKNetworkManager.Instance.battleID=int.Parse(startBattle.text);
13.      }
```

Code Listing 4.2: StartBattle function

Every PHP file starts by defining the parameters needed to login to the appropriate database and then uses them along with the parameters to specify the database source with a constructor to create an instance of the PDO base class to establish a connection. This the connection is established within a try block to catch any exceptions and throw an error message with the exception back to Unity. Once the connection to the database is successfully established, a prepare statement is created using the MySQL query with placeholders of the information to be added. It is then executed using "execute($_GET)" which executes the query using the parameters defined in the URL to replace the placeholders as $_GET is an array of variables passed to the file based on the URL parameters. MySQL then runs the queries and if the query was selecting from a table, returns the selected data.

### 4.2.2 Player Information

The need for a way to keep track of a player's information in a source other than the phone being used comes from the possibility of players switching phones, especially during the testing phases. A solution for this problem was to keep player information within a player database. The original schema for this relational database is shown in figure PD1. It was designed to not only hold the typical account information of a username and password, but also other information important to game play such as avatar information and who is online. Two tables in this design were not implemented in the final design, as the features which they were meant to store the information for were not implemented in the game. The ProPoints table was meant to store the player's proficiency points for each microgame as described in section 2.8.1 Proficiency Points. This table would have had three columns for the player's username, the microgame name, and the amount of points the player has, with the username and game columns being primary keys so rows that have the same username do not get overwritten every time a player earns points for a different game. The PunchCard table was meant to store information related to the use of the player's active punch card as described in section 2.8.3. This table would have had a number of columns equal to the max number of slots possible for a punch card as two columns would have stored the player's username and the type of punch card, while the remaining slots would have contained flags to mark if the slot was free, nonexistent, or the type that was filling it. However, the removal of these features caused the need for a redesign of the player database schema.

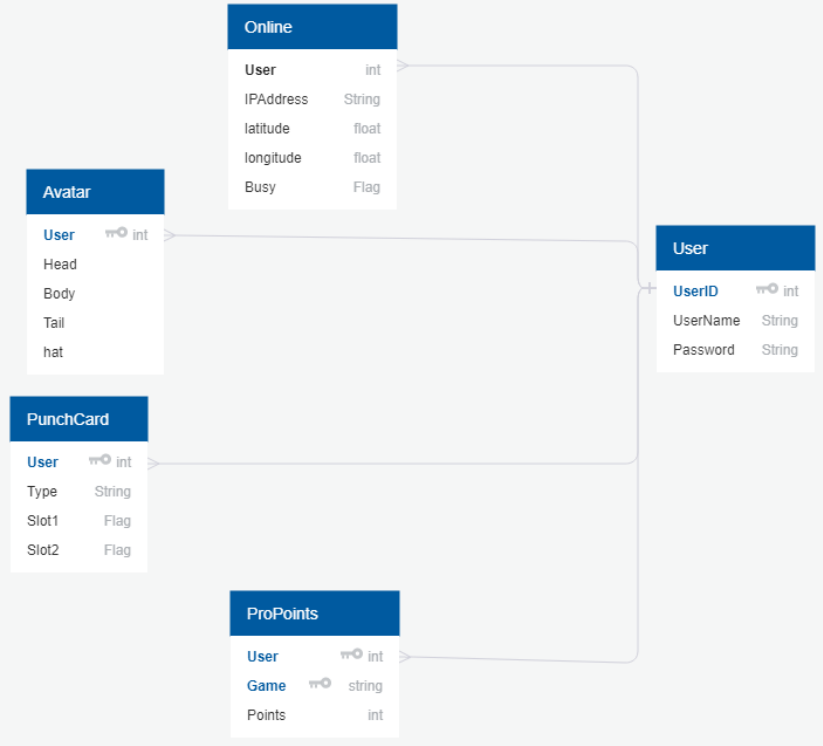



Figure 4.3: Original schema for Player database

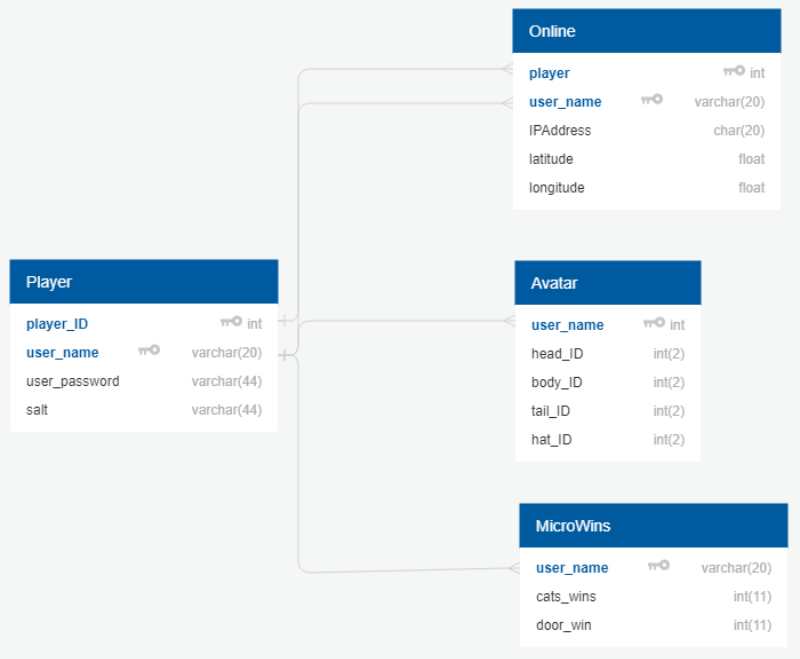



Figure 4.4: Player database schema

Figure 4.4 shows the relational database schema for the player database used in *Konbini Konnection*. The major change the player database schema went through was the removal of the PunchCard and ProPoints table. The MicroWins table replaced these as a way to keep track of how many times each player has won a specific microgame. Knowing the amount of microgames a player won allows for the possibility of a simple level system to be added as microgames could

be scaled up based on how many times previous a player won it before. The original purpose for it was to use in a simple unlockable item system that would have unlocked a new item in the character.

The Player database has a different risk than the analytic database as it takes in a user input which puts it at risk for SQL injection. SQL injection is the exploitation of improperly formatted queries which allow for input data to interfere with code. The method used to prevent SQL injection into the player database was the use of PDO prepared statements. PDO prepared statements parameterize queries by having variables bound to them.

### 4.2.3 Analytics

Since *Konbini Konnection* went through an iterative design process, a way to determine if it was meeting its experience goals was needed to support decision making. The game industry is diverse and different companies establish different processes for game analytics that depend on aspects such as features and target audience. While it required creating a specific process, it was decided to use game analytics through the collection of game telemetry to be derived into game metrics as a way to help support decision making. Game telemetry is game data that is obtained over a distance, while a game metric is an interpretable measure of something related to the game. The use of game telemetry is needed as data needs to be sent over a distance as the game is based on a mobile device, which is one of the popular applications of telemetry in games. Game metrics are more than measures of player behavior as it is broken down into three types: player, performance, and process. A core aspect of our game is the social context of playing against other players, pushing user-oriented analytics to be relied on in helping to measure the success in meeting the experience goals. This drove Player metrics to be the main focus as they are metrics related to people who interact with the game, such as total play time per player (El-Nasr, Drachen, & Canossa, 2013).

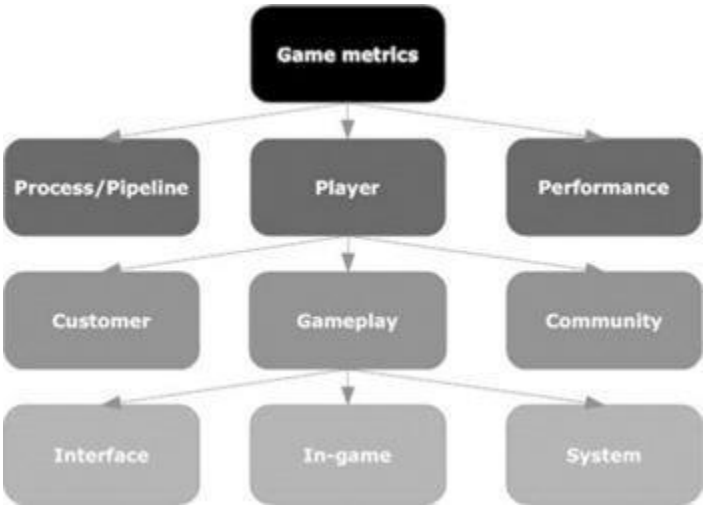


Figure 4.5: Hierarchy of Game Metrics (El-Nasr, Drachen, & Canossa, 2013)

Figure 4.5 shows a hierarchical diagram of game metrics, emphasizing player metrics as it breaks down into two more levels. The metrics focused on in the game analytics of *Konbini Konnection* are the center column of the diagram, as the goal was to collect data on in-game actions and behaviors of players. While the other metrics are important for improving the overall

game, time constraints made focusing on the aspect that provides data to measure the success of achieving the experience goals was more beneficial.

A tracking strategy is the transmission of a piece of information using a telemetry system (El-Nasr, Drachen, & Canossa, 2013). The game telemetry system was created using the process outlined in section 4.2.1. This tracking strategy is event based as when something is to be recorded, a call to the appropriate function within the AnalyticsManager class starts the process to send it to the analytic database. A relational database schema shown in Figure 4.6 was developed to create the analytic database which contains the data for various player metrics.

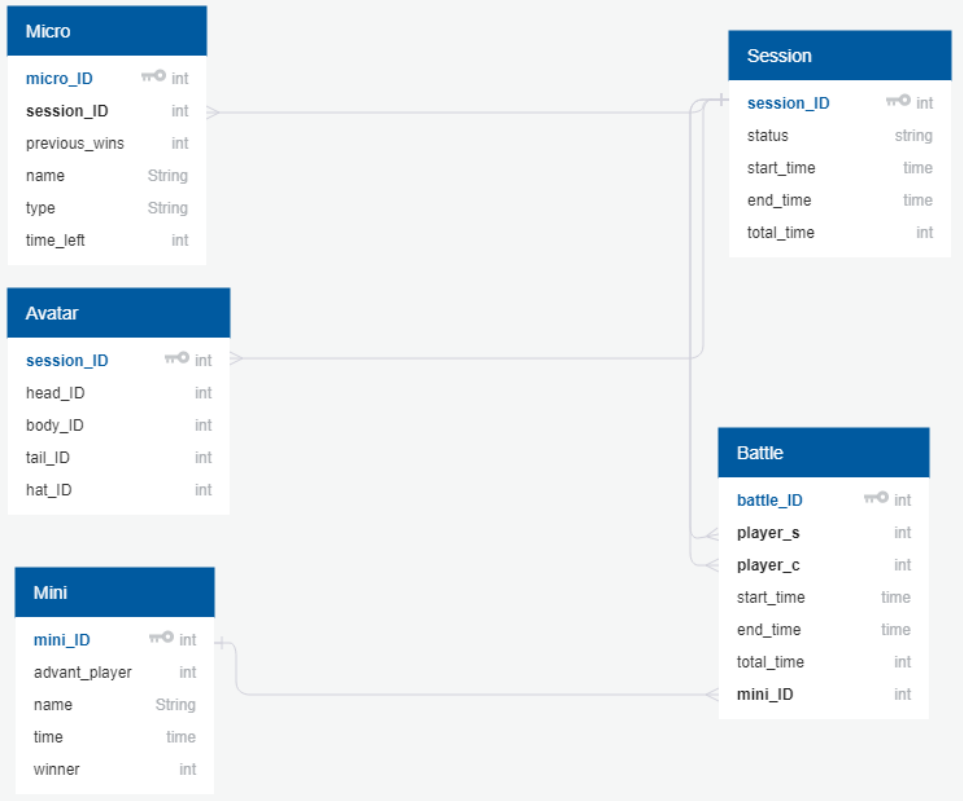



Figure 4.6: Analytic database schema

Current technology allows for detailed information on players to be collected, raising questions of privacy and ethics. This is a key issue when working with player telemetry as there is no largely agreed upon standard in game analytics, leaving ethics to remain a gray area (El-Nasr, Drachen, & Canossa, 2013). The use of separate databases with no crossover was done to ensure that the game analytics were not connected to any identifying information that would trace it back to a specific player. Within the analytics database, the data is anonymized by giving each player a session ID that is generated using auto increment within the session table. The use of this sacrificed better data accuracy to insure the privacy of players, as they only have that session ID from when they login to when they logout. This prevents the collection of changes over time such as changes players make to their avatar from their first login to the most recent.

One difficulty with the analytic system was the changing of formats between Unity and MySQL. Trying to get the datetime in Unity's format and put it into MySQL's datetime format was not working when having to put it through a PHP file. The solution used to get around this problem was to make use of MySQL's NOW() function to get the date and time need for the

analytics system. The use of NOW() also made it easier to get the difference between the start_time and end_time. Code Listing 4.3 shows how this is done using TIMESTAMPDIFF() inside of a query that is in a prepared statement in the endSession PHP file.

```
$sth = $dbh->prepare('UPDATE Session SET end_time= NOW()
                    WHERE session_ID= :ID;
                UPDATE Session SET status= :status
                    WHERE session_ID= :ID;
                INSERT INTO Avatar(session_ID, head_ID, body_ID, tail_ID, hat_ID)
                    VALUES (:ID, :head, :body, :tail, :hat);
                UPDATE Session SET total_time= TIMESTAMPDIFF(second, start_time, end_time)
                    WHERE session_ID= :ID;');
```

Code Listing 4.3: endSession prepared statement

## 4.3 Location Services

To develop the Location Services aspect of *Konbini Konnection*, we employed Unity's built-in Input class geared towards location. The player's current location is polled every 10 seconds that she is in the Storefront and sent to the Online Player database. When the location is sent, the database replies by handing the client application the usernames of all players within 50 meters of them in alphabetical order.

In order to create the function that polled the actual location of the player, we made use of the example code under Unity's LocationService.Start documentation (Unity, 2014). This code ensures that Location Services have been enabled by the user and that there has been no error in receiving a GPS signal. Then it simply sets the player location to a set of 3 coordinates, indicating Latitude, Longitude, and Altitude, based on her current GPS location.

The Latitude and Longitude are then passed to a PHP file, which calculates the Latitude and Longitude representative of 50 meters away in any of the four cardinal directions. For Latitude, this formula is represented by: Latitude (+/-) ((0.05 / R_EARTH) * (180/pi). For Longitude, this formula is represented by: Longitude (+/-) ((0.05 / R_EARTH) * ((180/pi) * cosine(Latitude * pi/180)). In both of these formulae, R_EARTH is equivalent to 6,378 kilometers, which is the approximate radius of the Earth. After calculating our ranges of Latitude and Longitude, we can then obtain all users in this range using a similar MySQL call to the one in Code Listing 4.4.

```
--MaxLatitude and MinLatitude represent Latitude range
--MaxLongitude and MinLongitude represent Longitude range
--PlayerName is the name of the player querying the database

SELECT *
FROM Online
WHERE Online.latitude <= MaxLatitude
AND Online.latitude >=MinLatitude
AND Online.longitude <= MaxLongitude
AND Online.longitude >= MinLongitude
AND Online.user_name != PlayerName
```

Code Listing 4.4: A sample MySQL call used for locating players

Once this result is obtained, we can feed the list of usernames back to the client using the "echo" keyword in PHP, in tandem with dividing characters. By looping through the usernames and using the call "echo '{username}#'," we can pass one long string of names and '#'s back to the client. The client can then use the '#'s to separate the string into multiple elements that can be used to generate the player objects that populate the Storefront.

## 4.4 Network Communication

In order for players to compete against each other, *Konbini Konnection* needed a system that would allow instances of the game to communicate with each other. We went through two different iterations of communication systems: Unity's Host-Client System and our "Konbinience" Server. At one point, we considered using Bluetooth connection to interface between devices; however, due to Unity's lack of exposure to the Android SDK, we were unable to access the desired functions.

The host-client system was our original choice for implementation, since it was integrated and fully supported by the Unity Engine. The problem arose that its intended use is between two computers and had many issues connecting mobile devices. Unable to get this system to work to fit our needs, we opted to build the "Konbinience" Server. This section will highlight the core design challenges involved with developing a functional server, including:

1. The specifications for design
2. The use of multi-threading and sockets
3. The design of our message object
4. The construction of networked game loop

### 4.4.1 Design Specifications

We needed the Konbinience server to continuously run on the device that was hosting it so that users could connect to it without having to worry about that connection being disrupted. This urged us to purchase a Raspberry Pi 3 as the computer that would house our server. The operating system Raspbian comes pre-installed with compilers for Java and Python. Of these two languages, we decided to use Java because of the object-oriented concepts and type-based interactions that it employs.

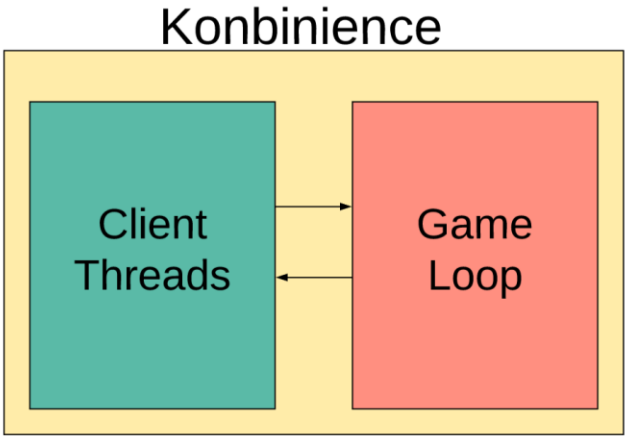



Figure 4.7: The Konbinience Server structure

Shown in Figure 4.7, the Konbinience Server is divided into two sections:

1. The “Threads” section, which handles all communication with the clients (Section 4.4.2),
2. The “Game Loop” that handles updating the state of the multiplayer games server-side (Section 4.4.4).

### 4.4.2 Multi-threaded Socket Programming

Multi-threaded socket programming, which makes up the “Client Threads” portion of Figure 4.7, involves three main types of objects: threads, sockets, and packets. When the Server starts up, it opens a “port” and waits for a client to connect to it. Once a client connects, a socket is created and the server and client then both wait for packets, which hold information that we want to transfer, and handle them as they arrive.

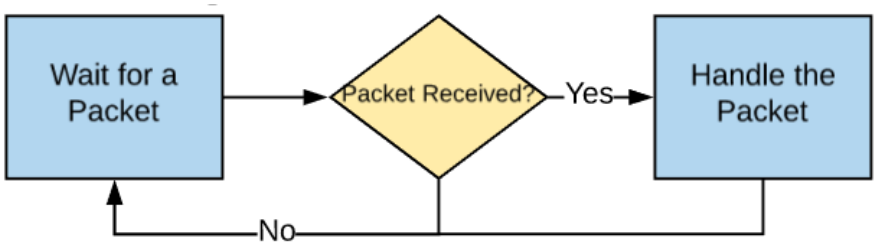



Figure 4.8: The flow of a socket thread

Figure 4.8 shows that a large portion of the time spent in a socket thread is waiting for packets. If this was run asynchronously, we would be stuck in this waiting phase and unable to update any other parts of the game state. To counteract this, we use threads, which Computer Hope defined as “[Portions of code] that are executed separately from the main program” (2017) Once our server receives a connection request, it accepts the user and assigns him a thread through which it can handle all incoming information from the client synchronously with the rest of the program.

The problem with threads, however, is that they are not natively supported by the Unity Engine. We could still use them, but we had to ensure that we did not utilize any of Unity's engine functionality from inside of these threads. This created a challenge of "How do we call Unity functions from our socket thread?" Inspired by MadDave (2012), we utilize a system that allows the socket thread to defer calls that would need to use Unity functions to the update loop.

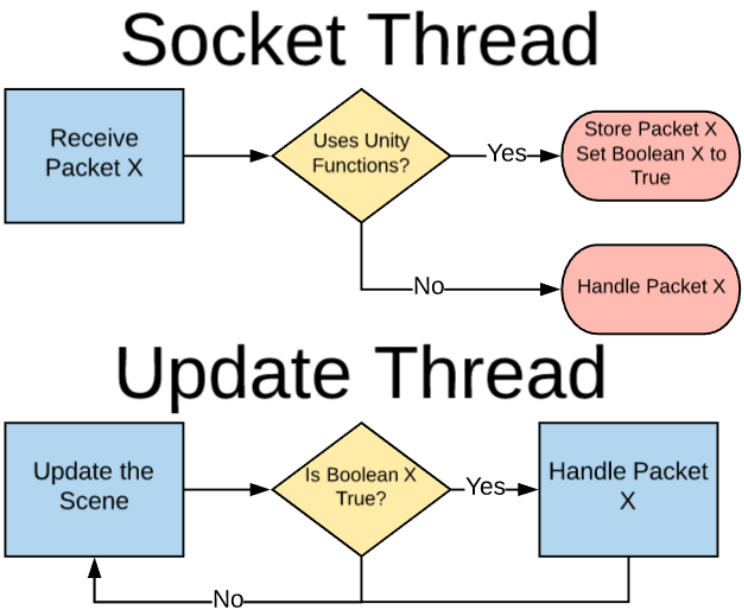



Figure 4.9: Multi-threading with Unity

Our solution, indicated by Figure 4.9, was to utilize booleans as the primary method of communication between threads. If a given packet needed to update certain aspects of the scene or game objects, we set a boolean indicative of that packet to true. This allows the external thread to inform the Game Engine that it would like to edit the scene with the information contained in the packet. The engine then takes that information and performs the necessary updates.

This multi-threading problem was extremely prevalent in the development of Dumplings. The game needed to update a lot of objects very quickly that it could not access from the socket thread. To get around this, every time the socket thread received an updated game state, it would convert a variable called "update" to true and store all of the deserialized "DumplingData" objects in an array, and all of the "PlayerData" into another. Then, during the next game loop, the update thread will cycle through all of the objects in the scene and replace their values with the deserialized values. The function ends with us resetting "update" to false.

Another example of this in Dumplings is the application of the feedback objects. When a plant is collected, the game will display either a blue circle or a red "X," depending on whether you collected it, or your opponent did, respectively. To do this, when a click is registered, the Server sends that object's index, as well as who clicked it, to the Client through the socket thread. In order to get that data to the update thread, we encapsulate it into a special object called a "FeedbackObject," push it into a queue, and set a boolean called "feedback" to true. A "FeedbackObject" has two fields representing the index and the name of the player who clicked the object. Then, in the update thread, each frame we check if "feedback" is true. If it is, we go

through the queue and, for each index, spawn a blue circle or a red “X” at that object’s location. Finally, we set “feedback” to false.

### 4.4.3 Messages

All of the packets sent to and from Konbinience utilize a structure that we refer to as a “Message.” These messages have three parts to them: the Header, the Divider, and the Body.

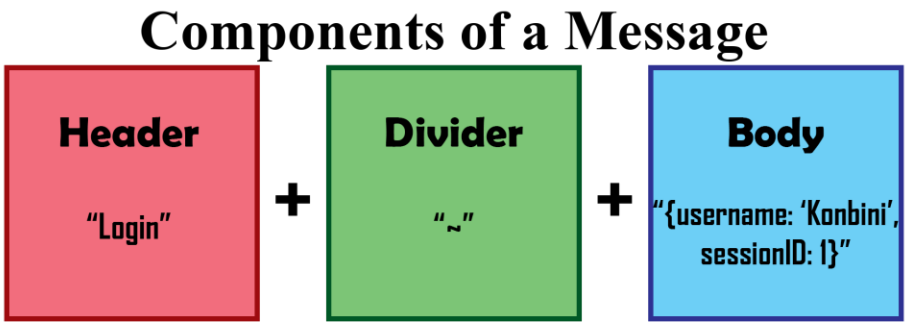



Figure 4.10: The construction of a Login Message

Figure 4.10 shows an example message for when the player “Konbini” logs in to *Konbini Konnection*. The Header acts as a notifier to let the recipient know what kind of message it should prepare to receive. Upon receiving a message, the recipient uses a “switch” statement to determine what function it should call on the data included in the body.

```
01.  function readMessage(Message){
02.    //Input: The message to be read: Message
03.
04.    Header = Message.split(DIVIDER)[0]
05.
06.    switch(Header){
07.      case “Login”:
08.        OnLoginMessageReceived(Message)
09.        break
10.
11.      case “Challenge”:
12.        OnChallengeMessageReceived(Message)
13.        break
14.
15.      ...
16.
17.      default:
18.        //Invalid Header
19.        break
20.    }
21.  }
```

Code Listing 4.5: An example of a message reading function

Code Listing 4.5 shows an example of how the Header is used in the process of reading a message using “Login” and “Challenge” message types as examples. The reader gets the Header and compares it to each type of Header that it could possibly receive. Upon finding the one that

matches, it will call the function that has been written to handle the kind of data that message sends.

The Divider is a predefined character that is never used in either the Header or the Body. It exists to allow the recipient to clearly define where the Header ends and the Body begins. *Konbini Konnection* uses the "~" character as its divider.

The Body of the message contains all of the information that needs to be communicated between the sender and the recipient. *Konbini Konnection* utilizes serializable message objects to represent the Body. A serializable object contains functionality that allows its state to be converted to a byte stream and copied (Oracle, 2012). There are several different notations for serialization, the most popular being JSON and XML. We decided to use JSON because it was easily supported by both Unity and Java.

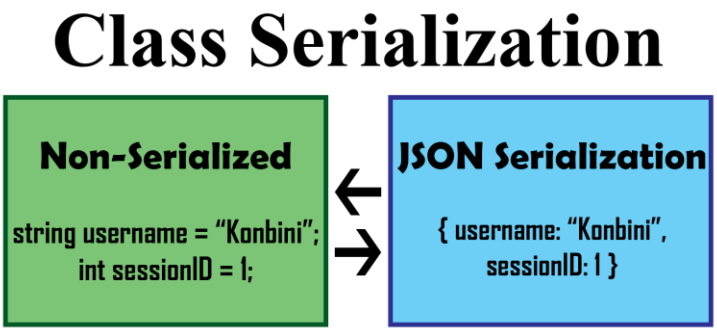



Figure 4.11: The difference between Serialized and Non-Serialized Objects

In order to serialize and deserialize an object, it's easiest to create classes in both the Messenger and the Recipient that have the same fields. This is due to the fact that when an object is serialized, the variable names are used as indices to the desired data, as shown in Figure 4.11. When the object is then deserialized, the parser looks for the variable with that field name and assigns the serialized data. For our serialization, we had to use two separate libraries communicate between the two separate languages:

1. Java Server: we used GSON 2.8.2, a library developed by Google
2. C# Client: we used Unity's built-in JsonUtility class.

Use of message classes and Serialization made networking between players incredibly simple, especially in the development of the Battle and the Dumplings Minigame. Both of these game modes required frequent updates of a game state containing many objects. By creating message classes that contained an array of those objects, we were able to send the whole array and deserialize it in the Client, in lieu of individually sending each object. Then, we were able to just loop through the deserialized array and pass the data to their corresponding objects in the Client Scene.

### 4.4.4 Game Loop

The Konbinience server uses a standard game loop that updates the state of all networked multiplayer games at a rate of 60 frames per second. To create this game loop with as little lag as possible, we employed use of a variable time step, inspired by Eli Delventhal (2012). This model checks the amount of time spent on previous loops and uses them to augment the wait time and keep the game at a constant frame rate. An example of a variable time step game loop is shown in Code Listing 4.6 below.

```
01.  function loopGame(fps){
02.  //Input: A predetermined Frame Rate: fps
03.
04.    timePerFrame = ONE_SECOND / fps
05.
06.    while(gameIsRunning){
07.      currentTime = getTime()
08.      updateLength = currentTime - lastTime
09.      lastTime = currentTime
10.
11.      delta = updateLength / timePerFrame
12.
13.      updateGame(delta)
14.
15.      wait(lastTime - getTime() + timePerFrame)
16.
17.    }
18.  }
```

Code Listing 4.6: A variable time step game loop

The loop in Code Listing 4.6 begins by getting the time and comparing it to how long it's been since the last time that it did so. This allows it to produce an accurate delta or change in time since the last loop. The delta is calculated as the number of frames that the last update took to complete. Knowing this is important for time-based events like physics calculations and countdowns. Finally, after all updates are completed, the loop waits for the remainder of the time that a frame should take to complete. This allows us to buffer out the updates if they complete too fast to keep the update rate constant.

In *Konbini Konnection*, the game loop hosted on the server is responsible for managing the updates of all Game Rooms that are being hosted. Every room holds the current state of a battle or minigame between two players. These rooms are stored in a list in the Game Manager and every frame and implement the IGame interface, informing each one that it should have an Update function that the game loop can call. These Update functions vary based on the needs of the game, but they contain two main similarities:

1. A Preparation Stage
2. 2) Utilization of an Event Queue.

The Preparation Stage is the state that the game is in while it is waiting for players to say that they are ready. In this stage the game will have been initialized, but the player will be unable to interact with any aspect of it aside from the "Ready" button. This stage ends once both players have confirmed that they are ready to play.

The Event Queue, shown in Figure 4.12, handles interactions that players make while the game is ongoing. We needed to know the order in which players clicked each object so that in

the event of a tie, we could know who clicked first. To do this, every time the Server receives a message indicating interaction from the client, such as a “Click,” the data of that interaction is logged in a special “Event” object and placed in the queue of the associated Game Room. In every frame update, the first event of each Game Room’s queue is updated and the game state is sent to the clients.

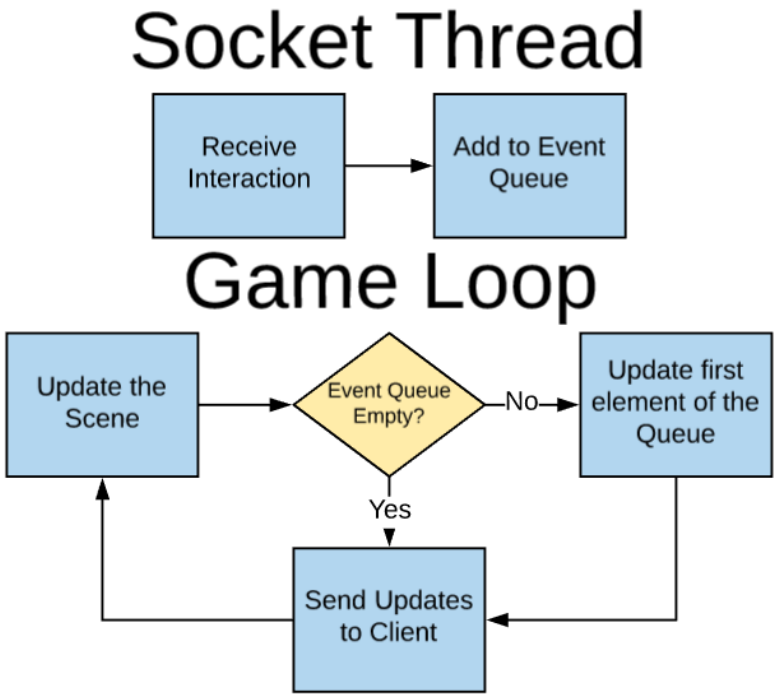



Figure 4.12: Implementation of the Event Queue

For example, Jake and Karen are playing Dumplings, and both of them tap the Flower at Plot 8 at almost the same time. Let us say that Jake pressed less than half a second earlier. Both send a message to the server saying that they clicked Plot 8, and a “DumplingsClickEvent” is created and added to the Event Queue in their instance of the Dumplings game on the Server. Since Jake pressed the plot first, his Event is added to the queue first. Then, the game reads the Events. It checks Jake’s click to see if Plot 8 has a collectable plant, which it does, and it removes the flower at that position and adds to his score. Next, it checks Karen’s click to see if Plot 8 has a collectable plant, which it does not because Jake collected it. Since there is no plant, Karen’s click has no effect on the current game state.

## 4.5 Modular Microgames

The Microgames are the building blocks of the *Konbini Konnection* experience. As a result, we needed a way of referring to them in the game context and switching them in and out on the fly. We resolved this by implementing an abstract class called a “Microgame.” The purpose of this class was to handle all of the startup and breakdown of each microgame. Each class that extends the class must create a definition for the pure virtual function “InitiateMicrogame(),” the purpose of which is to activate any UI necessary for the game. We utilized this class in tandem with the Prefab system Unity implements to make an object for each Microgame like the ones in Figure 4.13.

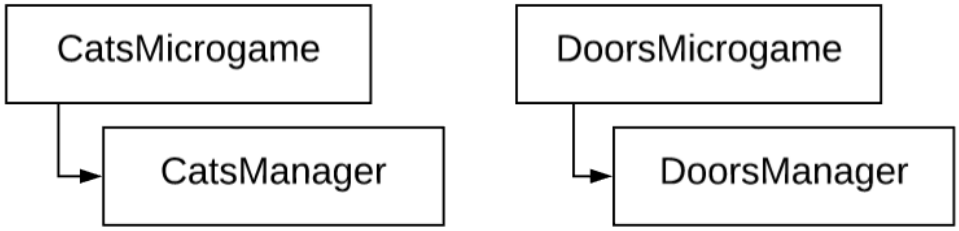



Figure 4.13: Hierarchical representations of the Microgame objects

Every Microgame has a manager class that handles all of its functionality in terms of instantiating game objects and handling player input. The Microgame class wraps around that manager class so that once the microgame is finished, it can destroy its parent object and everything will be cleaned up. In addition, any information that the game needs to transfer outside of the system, such as whether the player won or lost, is transferred. This flow of events is captured in Figure 4.14.

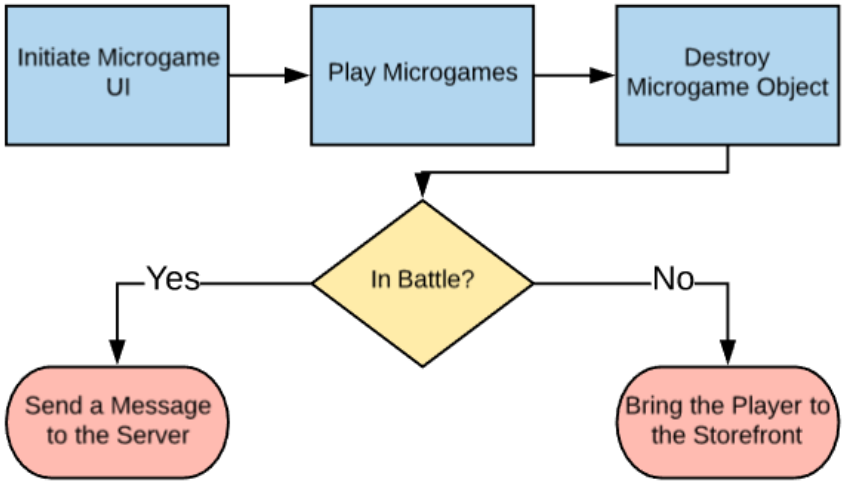



Figure 4.14: The Flowchart of the Microgame Class

## 4.6 Gestural Inputs

*Konbini Konnection* uses touch gestures as the main input method for gameplay. This input method is inherent to touch interfaces and provides users with stronger physical and emotional feedback during play. There are many different types of gestural inputs, discussed in Section 2.1 Hardware & Platform, but in the development of *Konbini Konnection*, we utilize two: "Taps" and "Swipes." The reading of gestural inputs was largely facilitated by the use of Unity's built-in Input class.

### 4.6.1 Taps

A "tap" is the simplest and most common form of gestural input. Players can facilitate a tap by placing their finger on a certain location on the screen and removing it from that same location. Then, whatever object the player tapped performs its function if it has one.

In *Konbini Konnection,* we primarily implement taps in the development of the Battle, Dumplings, and the Cats Microgame. All of these games feature objects that the player must interact with quickly, so they are well suited to taps. In Code Listing 4.7, we walk through the process of reading a tap using Unity's API.

```
01. if(Input.touchCount > 0){
02.
03.   Vector3 position = Camera.main.ScreenToWorldPoint(Input.GetTouch(0).position);
04.
05.   RaycastHit2D hit = Physics2D.Raycast(position, Vector3.back);
06.
07.   if(hit.collider != null){
08.     PerformAction();
09.   }
10. }
```

Code Listing 4.7: Reading a tap with Unity API

When a player touches the screen, each point on the screen that she touches is logged in a list called "Input.touchCount." The first thing Code Listing 4.7 does is ensure that that array has a Touch object in it. Next is to ensure that the player tapped a point on the screen that the player can interact with. To do this, we convert the screen coordinate where she tapped into a location in the game world. We then take that position and use it as a starting point for a Raycast. This Raycast will project as far back as it can and store the first object that it hits in the "hit" object defined on line 5. We then ensure that an object was hit on line 7, and then perform whatever action we would like to perform if it was.

For an application of this in-game, we turn to the Cats Microgame. In this Microgame, players are shown a 4x4 grid of objects on a bookshelf and told to locate the mice that are hiding among them. To search among these objects, the game requires players to tap on them. When a player taps on the Globe in position 9, we send a Raycast from where she tapped to the back of the play area. That Raycast hits the globe object, so we record the index of that object with regards to our "SearchLocations" array. Then, we perform our "SearchSpot" function, which uses the distance formula to calculate where the closest mouse is from that location.

### 4.6.2 Swipes

"Swipes" are slightly more complex than taps in terms of reading. A player can perform a swipe by placing their finger on the screen and, without removing it, sliding it to another location in a straight line.

In *Konbini Konnection*, we utilize swipes exclusively in the Doors Microgame. This Microgame features the player opening doors in different ways by swiping in different directions. Code Listing 4.8 represents our use of the Unity API to walk through reading a swipe gesture.

```
01. //startPos: The starting position of the drag
02. //swipeDirection: The vector between the start and end points
03. //directionChosen: Whether or not the drag is currently ongoing
04.
05. if(Input.touchCount > 0){
06.   Touch touch = Input.GetTouch(0);
07.
08.   switch (touch.phase){
09.     case TouchPhase.Began:
10.        startPos = touch.position;
11.        directionChosen = false;
12.        break;
13.
14.     case TouchPhase.Moved:
15.        swipeDirection = touch.position - startPos;
16.        break;
17.
18.     case TouchPhase.Ended:
19.        if(swipeDirection.magnitude > 1){
20.          directionChosen = true;
21.          break;
22.        }
23.     }
24. }
```

Code Listing 4.8: Reading a swipe input using Unity API

The function in Code Listing 4.8 begins by simply reading a "Tap." It then stores the information of that tap in a "Touch" object and performs a different action depending on the phase that the tap is currently in. In the "Began" phase, the player has just placed his finger on the screen, so we record the starting position of their swipe. Next, whenever the player moves his finger, we recalculate the vector that represents the line between the starting point and his current finger position. Finally, when the player removes his finger from the screen, we ensure he actually moved his finger and that he did not just perform a tap.

In the Doors Microgame, we utilize this exact method of reading swipes in order to determine how the player is trying to open the door. Once the swipe is complete and the "swipeDirection" vector is set, we perform a comparison against the absolute values of the x and y values of the vector. Then, we figure out if the larger change in direction was positive or negative to determine which of the four cardinal directions the player swiped in. For example, if the player swiped "up," this would mean that the change in y-direction between the start and end points of the swipe was larger than the change in x-direction. Then, when checking to see if the change in y-direction was positive or negative, we would find that it was positive.

# 5. Testing Process

After the initial design and implementation of *Konbini Konnection*, we went through an iterative design process to better meet the experience goals set. This iterative design process, also known as rapid prototyping, required a cycle of setting requirements, designing, implementing, testing, and evaluating (Interaction Design Foundation, 2018). Testing was necessary to provide data for evaluating for requirements needed for the next iteration. The following sections provide information on the challenges faced when testing, the methods used, and the results and changes it led to.

## 5.1 Challenges

When testing *Konbini Konnection*, we were faced with multiple challenges. One of these challenges was with the webserver we were using to host our PHP files and databases. The service we were using, 000webhost has all the websites it hosts take a nap for at least an hour a day, rendering it unusable during that time. This was a challenge with testing as the game cannot be played when the website is down as the PHP files are needed for basic functions such as logging in. As a result, testing sessions had to be planned around these naps.

## 5.2 Methods

A goal of testing was to provide data for evaluation, which meant that we needed to collect data. This was done by using in-game analytics, observation notes, and user input surveys. All tester were informed prior to the beginning of data collection that it was being collected and that they could remove themselves from the test at any time and that any data collected would be deleted. They were also informed that all data collected was anonymized and could not be traced back to them.

The first method used was in-game analytics to obtain accurate statistical data. This system measured various parts such as total session time. Details of this system can be seen in section 4.2.3.

Understanding the motivations behind player actions helped in understanding any confusion testers had when playing. This made collection data through testing observations an important resource to analyze to point towards improvements. Testing observations were collected through notes taken during every session and asking testers to talk through their thoughts.

User input surveys were the third data collection method used in order to gather feedback from testers. This allows for a non-verbal opportunity to express feeling about the game, which was beneficial for people who were not comfortable doing so out loud.

### 5.2.1 Testing Sessions

The use of an iterative design process required multiple testing sessions to be run after each version of the prototype was complete. All of the sessions were run on WPI's campus in different locations. Three different testing sessions were run with the following details:

1. Run on a Wednesday from 6pm to 8pm by both team members.
2. Run on a Saturday from 7:30pm to 11pm by Laurie Mazza.

3. Run on a Wednesday from 6pm to 10pm by both team members to start and finished by Laurie.

All sessions followed a similar testing procedure that consisted of three parts.

Part one was a short survey to gain a better understanding of the population being tested. This survey is located in Appendix A and asks the tester about things that would affect their ability to play such as how often they play mobile games.

Part two was having the tester play the game while both in-game analytics and observation data was collected. Testers were told how to create an account and that they would then be creating a character to start this part of the testing. After creating a character, testers were then instructed to go to soloplay mode and play each of the microgames until they won one of each. They were allowed to continue playing the microgames in soloplay and continue to the battle system when they felt ready. To get to the battle system, tester were told to return to the storefront and then pick the avatar of the team member guiding the testing. This sent a challenge request to the team member that would start the battle once accepted. Once the battle was complete, this portion of the testing sessions was finished.

Part three was a post-play survey that gave testers an opportunity to voice opinions and feedback that they might have felt uncomfortable doing out loud. Allowing testers ways to voice their opinions of *Konbini Konnection* was an important part in collecting feedback that provided helpful information when trying to locate issues with the various builds. The post-play survey shown in appendix N provided areas for feedback for each section of the game, allowing for both written and scale based feedback. All scale based feedback was done on a 1 to 6 rating with 1 being the lowest.

Minor changes were made to this process to better accommodate each session. The survey had sections removed for the third session to shorten the time it took to fill out to avoid scaring testers away. Changes focused on the collection of feedback on critical parts of the game. Testers were provided with information on how to play the game during the second and third sessions. A printed information packet was provided during session two, while the information was at various points within the game for session three.

### 5.2.2 Analysis

We analyzed all of the data collected during testing sessions to identify problem areas that were fixed for the next iteration. Data collected from observation notes and written by testers in the survey were used to point to problem areas through the frequency things were mentioned and how often it had a negative connotation. When something was frequently being mentioned with a negative connotation, it pointed to a problem area that most likely need fixes before the next iteration could go out. Data from the analytic system and scale based feedback from the survey was analyzed using the software *Tableu*, which allowed for simple statistical analysis. The survey data was overlapped with data from the analytic database when possible to allow for a more in-depth look between factors such as ranking of understanding of a microgame versus the amount of wins.

## 5.3 Results and Changes

The following sections talk about the results found during the testing sessions and the resulting changes made to the game.

### 5.3.1 Bugs

Throughout all the testing sessions various bugs were reported that affect game play. In order to improve the design of *Konbini Konnection*, these bugs were the first things that needed to be fixed for each iteration. All of the bugs in the order that they were found are mentioned in the following:

- Incorrectly logging out
- Still showing up when logged out
- Double tap to exit picks another microgame in battle system
- Lack of a result screen after battle
- Lack of clearly defined end condition in Dumplings
- Microgames appearing to be taken but still remaining playable in battle system
- Server setting sessionIDs to zero within the battle system

Incorrectly logging out caused the analytic system to only be able to collect data from microgames during the first testing session. Pulling from the wrong table was the cause of this bug as "SELECT LAST_INSERT_ID()" being used incorrectly prevented the correct sessionID from being pulled. Without the correct sessionID, the session cannot be ended thus preventing the recording of session times. This then prevented data from being crossed with survey data as the correct sessionID was not displayed to the testers.

Testers would sometimes still have their character show up in the storefront even when they had exited the game. The use of "OnApplicationPause()" was the cause of this bug as it would not register when the application was suspended by the testers, thus not calling the functions that led to the MySQL queries that would end the testing. The creation of logout button was used as a solution to this problem.

The bug of picking another microgame in the battle system when accidently double tapping was found during the first testing session. The system registering the second tap as the tester tapping the microgame node behind the microgame currently being exited was the cause of this.

The lack of a result screen to inform testers if they won or lost at the end of a battle was found during the first session. Providing players with their results using a screen that allows them to read it at their own pace was added to inform players of how they did.

The lack of a clearly defined end condition in Dumplings allowed for the possibility of the game to go on forever. This bug was present during the first and second sessions of testing. After it was discovered during the first session, a fix of cutting the dumpling replenishment value in half was tried. However, this did not fix the problem as it was discovered that the game would not end when one player ran out of stamina, even if that player had less points. For the third session, a fixed of checking the player who ran out of stamina and ending the game if they were the behind player was added. If that was not the case, the game ends once the surviving player beats the other player's score or runs out of stamina.

Microgames appearing to be taken but still remaining playable in battle system was found during the second testing session. The addition of art assets caused this bug as they assets could not be used the same way the place holders were. It was fixed by changing the way the system uses the icons.

Having the sessionIDs set to zero by the server during the battle system was discovered during the third testing session. Testers being logged into server as the account named Test and thus being given the sessionID for this account was the cause of sessionIDs being set to zero. The analytics system requires that the system remembers a player's sessionID correctly to allow

for queries to input information to the tables within the database correctly. The fix used for this was a check of the username logging into the server to prevent an account being set as Test.

### 5.3.2 Character Creation

Feedback on the character creation system came from the first two testing sessions. Based on this feedback, minor changes were made to the system. Testers in the first session gave the system an average level of understanding of 5 and an average level of enjoyment of the options of 4.667. From the first session, most of the comments about this section were about the desire for more options. Unfortunately, due to a lack of time, creating and implementing new options was not possible. One part of confusion was that the button to press to save a character, whether for the first time or editing, it said "Create Character." This caused confusion when editing a character as testers thought they were making a new one when returning to the closet. The text was set to change to say "Accept Changes!" when editing a character to end this confusion.

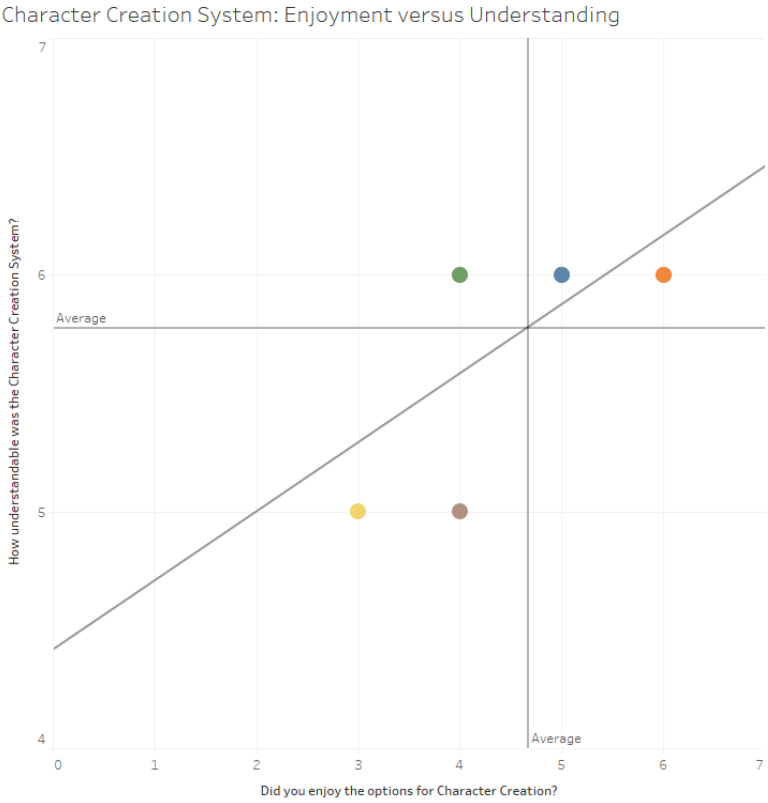



Figure 5.1: Character Creation System: Enjoyment versus Understanding

Testers in the second session understood the system well with an average level of understanding of 5.778 and the options were enjoyed with an average rating of 4.667. There was an upward between the enjoyment of the options and the understanding of the system shown in Figure 5.1 with a p-value of 0.05. This points to a probability of testers enjoying the system more because they understood it. Comments from testers pointed to some frustration with not being

able to go back when going through the options. Code was written to implement a back button for each option, but it was not implemented due to time.

### 5.3.3 Microgames

Throughout all of the testing sessions, feedback on each of microgames were collected. A total of 214 microgames were played during the three testing sessions of which 150 were won. The following section talks about the specific findings from all of the sessions about each of the microgames.

Doors

Doors was played a total of 100 times over the three testing sessions. It had the least amount of changes throughout the iterative design process, with minor changes being made to improve feedback and difficulty.

A shake was added when the door was swiped incorrectly to give players feedback. During the first session, doors was played 13 times and lost 5 of those times, giving it a 38% loss rate. Doors was understood by most as it had an average level of understanding of 5.667 and comments describing play as simply “swipe to open doors.” Testers generally liked playing doors as the average level of enjoyment was 5. The comments mentioned the lack of feedback when swiping incorrectly made it hard to know when they swiped wrong.

A major lack of difficulty with Doors was highlighted during the second session as not a single tester lost a game. The average level of understanding was 5.889 and the average helpfulness of the info page was 5.33. Testers had a neutral level of enjoyment with an average of 3.78. An increase in the number of doors needed to swipe through to beat Doors was increased from 10 to 15 in order to increase the difficulty.

The final testing session had a total of 60 plays of Doors with 10 plays resulting in a loss. This gives Doors a loss rate of 16.7%, which is an improvement from the previous session. The level of understanding was an average of 5.417, staying similar to the previous sessions. Enjoyment increased with an average of 4.5.

Cats

Cats was played a total of 114 times over the three testing sessions. It went through multiple changes to improve player understanding.

Cats had a 56% loss rate during the first session with 9 out of 16 plays resulting in a loss. Both the average level of understanding and the average level of enjoyment was 3.667, indicating that testers had neutral feeling toward the game. The comments mentioned that when playing Cats, testers only understood how parts of the microgame worked. A solution to this problem was to provide the player with text of how to play the game.

During the second session, Cats was played an average of 4.89 times per tester with the most being played 8 times by one tester. Testers won an average of 3 games of Cats. Cats had an average level of understanding of 4.111 with the average level of helpfulness of the info page

being a 4.667. The overall average rating of enjoyment for Cats was low with 2.89. Five of the testers lost at least one game of Cats, of which they lost an average of 3.4 games each. The reported average level of understanding of 5.8 from these testers points to a lack of understanding not being the reason they lost. They also had a neutral level of enjoyment of 3.6, meaning loosing was most likely not a driver in their enjoyment levels. The comments from the survey pointed to the use of the numbers in diagonal directions being an area that caused misunderstandings. Changing from numbers to arrows pointing in the direction of the closet mouse was done to fix this confusion. Comments also pointed to Cats giving players a disadvantage in the battle system because of the length it can be played for as the only lose condition was running out of tries. A ten second timer was added to match the max possible length of play to that of Doors.

Cats was played a total of 54 times during the third testing session with a loss rate of 42.6%. Testers had an improved understanding of how to play from previous sessions with an average level of understanding of 5.25. The average level of enjoyment was an improvement from the previous sessions with 4.667. Comments highlighted that testers had trouble finding information that was helpful such as the number of tries and when to wait. This information was changed to a darker color so it would stand out visually and be easier to find.

### 5.3.4 Battle System

During each testing session, the battle system was the last section testers explored. Multiple changes were made to this system based on the feedback provided by the tester.

Testers did not understand the battle system during the first session as the average understanding of how to get to a battle was a 3.333 with the average level of understanding of how the system worked being a 2.667. This left testers with a neutral average level of enjoyment of 3.333. A lack of understanding was also highlighted in comments from testers as did not know what each symbol meant, why it would become grayed out, and why it was all happening. This was made clearer for the next sessions as information was provided to them in a printed packet (Appendix B).

Within the battle system, testers did not understand the minigame Dumplings; the average level of understanding how to play was 2.333. However, this did not highly impact their level of enjoyment; the average level of enjoyment was a 4. Comments highlighted a lack of understanding as well as a few bugs. The lack of understanding was fixed for the next sections by providing information on how to play before Dumplings is played. Details on the bugs and how they were fixed are located above in section 5.3.1.

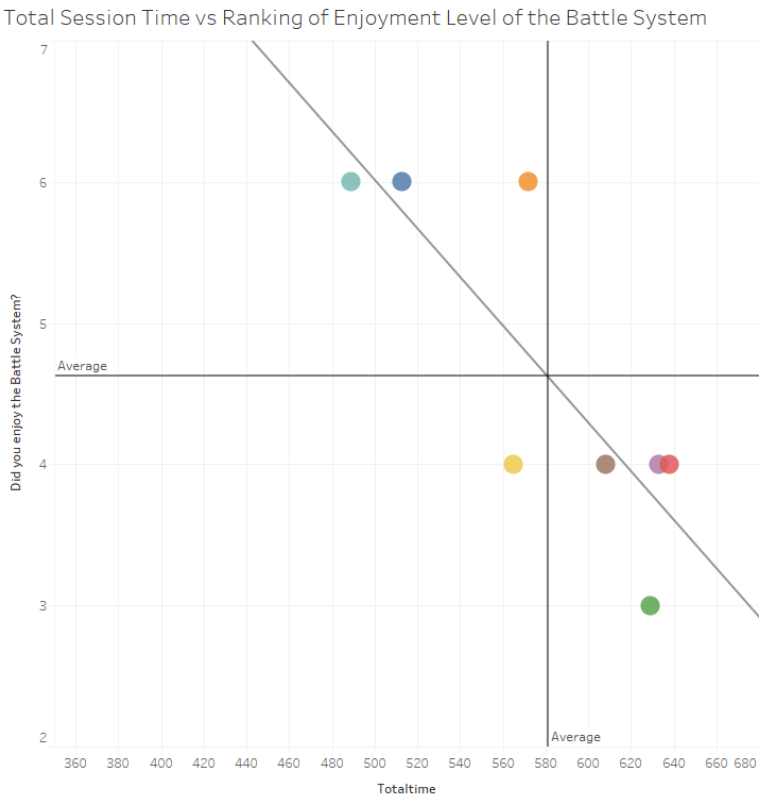



Figure 5.2: Total Session Time versus Level of Enjoyment of the Battle System

Battles took an average of 248 seconds or 4.13 minutes to complete during the second session. A bug described in section 5.3.1 might have caused the time to be longer. Testers understood how to get into a battle with an understanding level of 5.375 and had slight difficulties understanding how the system worked with an understanding level of 4.625. The information page provided them with some help as testers gave it an average helpfulness level of 4.625. Testers had an average level of enjoyment of 4.625 which had a downward trend with the overall time of the session as shown in Figure 5.2. This trend had a p-value of 0.013, which is in the range of statistical significance, meaning the r-squared value of 0.671 possibly points to the total session time accounting for 67.1% of the variation in the level of enjoyment of the battle system. One reason for this that was observed and commented on by testers was that playing Cats was a disadvantage in the battle system as it took too long to play. This along with a fix was mentioned above in section 5.3.3.

The last part of the battle system was the minigame Dumplings, which had an average level of enjoyment of 5.778. Testers found the information packet helpful, with an average helpfulness level 5.111 and understood how to play with an average level of understanding of 5.778. The person who went into Dumplings with the advantage ended up winning 50% of the time. Comments from the testers from both the survey and recorded during testing pointed to a lack of understanding to how the game was supposed to end. This was not helped by the bug described in section 5.3.1, which allowed Dumplings to go on forever. The fix for this is

described in section 5.3.1 and was paired with an updated to include the information within the game before it is played as well as clearly stating the end conditions within this information.

The need for visual feedback within the battle system was highlighted in the comments during the third session as testers found it difficult to figure out their standing before the minigame. Time for battles could not be collected during this session because of a bug within the server. Testers were able to figure out how to get to a battle with an average level of understanding of 4.417. They understood the system with an average level of understanding of 4.583 and enjoyed it at an average level of 5. The feedback needed in the system was implemented by having circles appear over microgames the player has won.

From the few times that were recorded for some battles during the third testing sessions, one was recorded as taking 834 seconds or 13.9 minutes. This was because dumplings did not have a limit on how long it could be played for, making it possible for it to go on forever between two people who are evenly matched. Stamina kept being collected which pushed the amount testers had to be greater than what was started with. This paired with the fact dumplings never stopped spawning, allowed for play to continue until one player got tired. A limit was placed on how much stamina a player can have to prevent long battles from happening. Despite the possibility of long battles, testers enjoyed dumplings, giving it an average of 5.25. It was well understood with an average of 5.25, and the information page shown in Figure 5.3 was helpful with an average of 5.25 as well.


Figure 5.3: Dumplings Information Page

### 5.3.5 Overall Game

*Konbini Konnection* had average ratings from each testing session that ranged from 3.625 to 5. The population of testers in the first session plays mobile games with an average frequency of 3. The most popular type of mobile game played in this population is strategy games with 66.7% reporting to play them. Since it was not possible to record times during this session, it is unknown what the average times were.

Tester in the second session reported the frequency at which they play mobile games to be an average of 2.78 and 56% of testers reported to play RPG mobile games. The average total time spent per test was 580.875 seconds and the average rating of overall enjoyment was 3.625. There was a downward trend between these, but fell short of being statistically significant with a p-value of 0.113.

The average total time spent per test during the third session was 385.25 seconds or 6.4 minutes. This population plays mobile game with an average frequency of 3.33 and 83.3% play mobile puzzle games.

## 5.4 Conclusions

Overall, testing each iterative design helped improve *Konbini Konnection* by pointing out bugs and revealing areas that were not enjoyable which allowed for changes to be made. The next major area for testing would be player to player interaction. Testing player to player interaction was planned to be done using the three different testing setups as follows:

- Tester to tester setup would have two testers play in the same area to see if they would naturally go into a battle.
- Multiple testers controlled setup would have had a varied number of testers set to play in a controlled environment to see how they would interact with one another.
- Multiple testers uncontrolled setup would release the game to anyone willing and able to test to allow testers to play in any location of their choosing.

All of these setups would allow both testers who have tested before and testers who have not would be included in this testing to measure how much experience contributes to success. Testers who have not played before would be told to play the microgames before trying the battle system as knowledge of the microgames is necessary for it.

Any changes made to the current iteration of *Konbini Konnection* that would add any features that were cut needs to be tested using the testing methods above. These methods proved to find bugs that could have possibly gone unnoticed in testing that focuses on player to player interaction.

# 6. Post-Mortem

With the development of *Konbini Konnection* complete, we assessed the project and the experience as a whole. In this section, we discuss that assessment in terms of learning, things that were successful, things that were not successful, potential improvements, and future development possibilities.

## 6.1 What Did We Learn?

Since *Konbini Konnection* was completed at Ritsumeikan University BKC in Japan, most of what we learned was about Japanese culture. Aspects of the culture that were most influential to our design included:

- The use of mascot characters to promote products and locations.
- The usefulness and popularity of Convenience Stores (Konbini)
- The many idioms that are used in everyday speech

Without these pieces of inspirational culture, *Konbini Konnection* may have wound up being a completely different game.

Another thing that we learned a lot about was providing feedback to players, motivating them, and assessing our methods. Early on in the development process, we did a lot of research on how we could make *Konbini Konnection* a continuously engaging experience that would keep players interested. We looked into psychology, design theory, cuteness, and more in an effort to understand how the human brain works so we could develop a system that works. That system may not have made it into the game, but the assessment system did. Laurie did a fantastic job producing a functional analytics system and providing analysis of that system. This aided the iteration upon our proof of concept and allowed us to drastically improve our game in a short period of time.

In terms of technology, we learned topics in Mobile Development, Unity, PHP, and Networking. For starters, neither of us had ever worked on a Mobile game before, so there was a lot to learn on that front in terms of building and loading the game onto the device. Laurie had never programmed in Unity or PHP before and was able to employ a system that allowed the game to interface with the databases that were hosting our data. Lastly, Erik was able to produce a functioning game server from scratch that was able to allow players to communicate with each other in game.

## 6.2 What Went Right?

There were several positive outcomes that came out of the development process of *Konbini Konnection*. First, we were excited that we were able to produce a scoped-down version of the game early on, and for the most part it was realized. Our initial proof of concept featured:

- Location Services
- Account Creation
- Networking Services
- Character Creation
- The Battle System
- 3 Microgames
- 2 Minigames
- Solo Mode

- Punch Cards
- Unlockables

In retrospect, this was a very large scope considering the initial time frame that this project was meant to encompass. That being said, we are happy with the fact that our current proof of concept features most of these features, with the exception of Punch Cards, Unlockables, one Microgame, and one Minigame. In addition, Laurie was able to produce most of the art assets that were planned to be produced for the proof of concept.

We were also able to produce a functioning solution to Networking player-to-player, as well as player-to-database. Our lack of prior knowledge in PHP and socket programming made both of these tasks daunting, but in the long run, we were able to get it done. They were used, iterated upon, improved, and finally successful. Without building these tools, we would not have realized our goal of a positive, social experience between players.

## 6.3 What Went Wrong?

There were two major things that went wrong during the development of *Konbini Konnection*. First, we could not get our 2D-Rigging system to integrate with Unity. Second, after our time in Japan was over, our Networking system was not functioning properly, so we were forced to ask for an extension to our development time.

We wanted to implement a 2D-Rigging system to help provide some kind of movement to the custom characters. In order to do this, Laurie tried several different 2D rigging programs and tried to generate animations that could be used with interchangeable parts in Unity. The main program that she tried was called DragonBones, which was open source and showed the ability to produce Unity files. Sadly, upon trying to implement the system into Unity, it would not integrate due to updates with Unity 2017.

Another method tried when attempting to implement was the use of Unity's Animator to create pin-joint style animations. This system allowed for the creation of a 2D rig with arm animations, but did not allow for the dynamic creation of characters as swapping parts caused errors with the system. The need for every character body to have seperate arms and the error with swapping parts caused this idea to be tabled in the interest of time.

One of the reasons that we chose Unity at first was the fact that it utilized its own built-in Networking API. This made the process of creating a networked game far less daunting as there were tutorials and documentation files we could look at to make it happen. Unfortunately, after following the tutorials and making the game functional between computers, we were unable to produce the result with Mobile Devices. During the extended development period, the use of Unity's Networking API was scrapped entirely and replaced with our own Konbinience server hosted on a Raspberry Pi 3.

## 6.4 Future Developments

Future developments for *Konbini Konnection* would likely start with the completion of our desired proof of concept, or even the rest of the intended game. That would include the implementation of Punch Cards, Unlockables, and more Micro and Minigames. The major reason for this is that, in testing, players reported that they were bored with the amount of content that we provided. In addition, the implementation of our intended feedback system is definitely something that we would like to test and see if it would have worked as expected.

There are also several different artistic aspects that Laurie would like to expand upon in continued development of the game. The first is the 2D-Rigging system. Since this was a desired aspect early on in the design of our Character Creation system, it was unfortunate that it had to get put off. With future development time, it would be nice to figure out what was wrong and fix it to make the characters less static.

In addition, playtesters were interested in having more character creation options, especially in terms of accessories. Players seemed to enjoy the ability to aesthetically design their own character. Including more options would make the cast of creatable characters far more diverse and give us the ability to hide some of them behind unlockables. Another interesting experiment on this front would be allowing players to choose their own colorations for their mascot character, as it would largely open up the animals that we could choose parts from.

We also would like to start showing the game off at more public events, such as Showfest, PAX East, and MassDigi. Spreading the game and getting feedback would be a fantastic way of staging our game to, at some point, be released on the Google Play Store.

# References


Alastor. (2016, July 9). Fruit Ninja VR Early Access on Steam. [Digital image]. Retrieved from https://mmoexaminer.com/fruit-ninja-vr-early-access-steam/

Alfornso, A. (2006, August 9). The History of Wario Ware. Retrieved from http://www.ign.com/articles/2006/08/09/the-history-of-wario-ware

Bartle, R. A. (1996) Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs. Retrieved from http://mud.co.uk/richard/hcds.htm

Bathgate, M. (2003). *The fox's craft in japanese religion and culture : shapeshifters, transformations, and duplicities*. Retrieved from https://ebookcentral.proquest.com

Beardsley Zoo. (2018). Red Panda [Digital image]. Retrieved from http://www.beardsleyzoo.org/project/red-panda/

Bertoli, B. (2016, August 28). Ko Takeuchi Is The Master Of Nerd Culture Doodles. Retrieved February 24, 2018, from https://kotaku.com/ko-takeuchi-is-the-master-of-nerd-culture-doodles-1785573509

Brassor, P. (2008, August 08). The obsession over those dumbed down cute mascots. Retrieved February 22, 2018, from https://www.japantimes.co.jp/news/2008/08/03/national/media-national/the-obsession-over-those-dumbed-down-cute-mascots/#.Wo9WoYPwaUl

Carnivore. (2018). In Encyclopædia Britannica. Retrieved from http://academic.eb.com/levels/collegiate/article/carnivore/105981

Convenience Stores. (2017, February 07). Retrieved February 22, 2018, from https://www.japan-guide.com/e/e2071.html

Delventhal, Eli. (2012) Game loops! [Online forum comment] Message posted to http://www.java-gaming.org/index.php?topic=24220.0

Dustin-horne. (2013, April 04). C#: Encryption-Decryption Class that works good in a project. [Online forum comment] Message posted to https://forum.unity.com/threads/c-encryption-decryption-class-that-works-good-in-a-project.394688/

El-Nasr, M. S., Drachen, A., & Canossa, A. (2013). *Game analytics: maximizing the value of player data*. [Books24x7 version] Available from http://common.books24x7.com.ezproxy.wpi.edu/toc.aspx?bookid=76617.

Foster, M. D. (2008). *Pandemonium and parade : japanese monsters and the culture of yokai*. Retrieved from https://ebookcentral.proquest.com

Fruit Ninja. [Computer software]. (2010). Brisbane: Halfbrick Studios.

Gayler, M. (2016, July 19). I've made more friends through Pokemon Go this week than in a year of London living. Huck Magazine. Retrieved February 28, 2018 from http://www.huckmagazine.com/art-and-culture/met-people-pokemon-go-weekend-year/

Goodyear, S. (2014, September 18). Millennials Love Transit Most, Boomers Still Stuck on Cars. Retrieved from https://www.citylab.com/transportation/2014/09/new-study-millennials-love-transit-most-boomers-still-stuck-on-cars/380380/

Gordenker, A. (2008, July 14). Tanuki genitals. Retrieved February 22, 2018, from https://www.japantimes.co.jp/news/2008/07/15/news/tanuki-genitals/#.Wo-LeIPwaUm

Hall, J. (2006, April 06). [Folk art Tanuki]. Retrieved from http://www.eugenewoodbury.com/foxwolf/kitsune.htm

Harmon, J. (2016, March 31). Nintendo first Miitomo ad is aimed at wacky Miillennials. Retrieved from http://www.egmnow.com/articles/news/nintendo-first-miitomo-ad-is-aimed-at-wacky-miillennials/

Interaction Design Foundation. (2018, February 14). Design iteration brings powerful results. So, do it again designer! Retrieved March 02, 2018, from https://www.interaction-design.org/literature/article/design-iteration-brings-powerful-results-so-do-it-again-designer

Joly, O. (2017, February 24). Camellia's history. Retrieved from http://www.camellias.pics/histoire-gb.php

Kageyama, Y. (2006, June 14). Cuteness a Hot-Selling Commodity in Japan. Retrieved February 24, 2018, from http://www.washingtonpost.com/wp-dyn/content/article/2006/06/14/AR2006061401122.html

Kerr, H. (2016, November 23). What is kawaii – and why did the world fall for the 'cult of cute'? Retrieved February 24, 2018, from https://theconversation.com/what-is-kawaii-and-why-did-the-world-fall-for-the-cult-of-cute-67187

Lavendei2. (2015, May 09). [Kumamon Daiso Japan]. Retrieved from https://twitter.com/lavendei2/status/597273076946968577

Lorenz, K. (1970). *Studies in animal and human behaviour.* Cambridge, Mass: Harvard University Press.

MadDave (2012, August 22) How do I invoke functions on the main thread? [Online forum comment] Message posted to https://answers.unity.com/questions/305882/how-do-i-invoke-functions-on-the-main-thread.html

Marjanovic, D. (2012, February 21). PDO vs. MySQLi: Which Should You Use? Retrieved February 28, 2018, from https://code.tutsplus.com/tutorials/pdo-vs-mysqli-which-should-you-use--net-24059

McKirdy, E. (2014, May 12). Japanese cuteness overload could result in mascot cull. Retrieved February 22, 2018, from https://www.cnn.com/2014/05/12/world/asia/osaka-mascot-cull/

Neitram. (2007, March 3). Kopfproportionen [Digital image]. Retrieved from https://commons.wikimedia.org/wiki/File:Kopfproportionen.svg

Neko Atsume. [Computer Software]. (2014). Japan: Hit Point Co. Ltd.

Nittono, H., Fukushima, M., Yano, A., & Moriya, H. (2012). The Power of Kawaii: Viewing Cute Images Promotes a Careful Behavior and Narrows Attentional Focus. PLoS ONE, 7(9). doi:10.1371/journal.pone.0046362

Oracle. (2012). Serializable Objects. Retrieved from https://docs.oracle.com/javase/tutorial/jndi/objects/serial.html

OWASP. (2015, October 14). Using Rfc2898DeriveBytes for PBKDF2. Retrieved from https://www.owasp.org/index.php/Using_Rfc2898DeriveBytes_for_PBKDF2

Pokemon GO. [Computer software]. (2016). San Francisco: Niantic.

Red panda. (2018). In Encyclopædia Britannica. Retrieved from http://academic.eb.com/levels/collegiate/article/red-panda/58235

Ripley, W., & Henry, E. S. (2014, June 10). How a hyperactive, dancing, talking pear became a Japanese obsession. Retrieved February 22, 2018, from https://edition.cnn.com/2014/06/10/world/asia/japanese-mascot-pear/

Robertson, A. (2013, December 11). Parent Trap: StreetPass Makes Social Networks Social Again. Nintendo Life. Retrieved February 28, 2018, from http://www.nintendolife.com/news/2013/12/parent_trap_streetpass_makes_social_networks_social_again

Sanders, K. (2017, February 13). To Use Or Not To Use: Touch Gesture Controls For Mobile Interfaces. Smashing Magazine. Retrieved February 27, 2018, from https://www.smashingmagazine.com/2017/02/touch-gesture-controls-mobile-interfaces/
Schumacher, M. (1998). Tanuki in Japanese Artwork. Retrieved February 22, 2018, from http://www.onmarkproductions.com/html/tanuki.shtml
Shearin, M. (2011, October 03). Triumph of kawaii. Retrieved February 24, 2018, from http://www.wm.edu/research/ideation/ideation-stories-for-borrowing/2011/triumph-of-kawaii5221.php
Smith, K. (2016, May 25). Don't Be In The Dark About How Light Affects Color. Retrieved February 22, 2018, from http://www.sensationalcolor.com/understanding-color/theory/dont-be-in-the-dark-about-how-light-affects-color-5154#.Wpi5MmrwaUl
Sonders, M. (2016, December 07). Pokémon GO demographics: The evolving player mix of a smash-hit game. Retrieved from https://medium.com/@sm_app_intel/pok%C3%A9mon-go-demographics-the-evolving-player-mix-of-a-smash-hit-game-b9099d5527b7
Streetpass Mii Plaza. [Computer software]. (2011). Japan: Nintendo.
Symeonidis, S. (2016, April 29). The use of tactile feedback for mobile devices. In Academia.edu. Retrieved February 27, 2018, from https://www.academia.edu/6356819/The_use_of_tactile_feedback_for_mobile_devices
Thread. (2017). In *Computer Hope*. Retrieved from https://www.computerhope.com/jargon/t/thread.htm
Toto, S. (2012). Gacha: Explaining Japan's Top Money-Making Social Game Mechanism [Social Games]. Retrieved from https://www.serkantoto.com/2012/02/21/gacha-social-games/
Unity. (2014). LocationService.Start. Retrieved from https://docs.unity3d.com/ScriptReference/LocationService.Start.html
WarioWare, Inc.: Mega Microgame$! [Computer software]. (2003). Japan: Nintendo.
West, M. I. (2009). *The Japanification of children's popular culture: from godzilla to miyazaki*. Lanham, MD: Scarecrow Press.
Zhang, J. (2015, March 29). Hokkaido Red Fox [Digital image]. Retrieved from https://mymodernmet.com/popshiretoko360-hokkaido-cute-animals/

# Appendix A: Playtest Survey

## Playtest Survey

* Required

### Pre-test Survey

1. **How often do you use your Mobile Phone daily?** *

*Mark only one oval.*

- ◯ An hour or less
- ◯ 1 - 3 hours
- ◯ 3 - 6 hours
- ◯ 6 - 9 hours
- ◯ 9+ hours

2. **How often do you play Mobile Games?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Never | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | All the time |

3. **What kind(s) of Mobile Games do you play?** *

*Check all that apply.*

- ☐ Card
- ☐ Clicker/Idle
- ☐ Geocaching
- ☐ Puzzle
- ☐ RPG
- ☐ Strategy
- ☐ Twitch
- ☐ I don't play mobile games
- ☐ Other: _____

### That's all for the Pre-survey!

We will now guide you through play of Konbini Konnection!

### Playtest Survey

4. **Please enter your session ID** * _____

## Character Creation System!

This section will contain only questions for feedback on the Character Creation System.

5. **How understandable was the Character Creation System?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| I didn't understand it at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Completely understandable. |

6. **What was unclear if anything?** *

7. **Did you enjoy the options for Character Creation?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Not at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | I loved them! |

8. **What did you particularly like or dislike?** *

9. **Do you have any suggestions for improvements to the current system?**

# Cats

This section will contain only questions for feedback on the 'Cats' Microgame.

10. **About how many times did you play Cats?** *

*Mark only one oval.*

| Just Once | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 10 or More |
|---|---|---|---|---|---|---|---|---|---|---|---|

11. **Did you understand how to play?** *

*Mark only one oval.*

| I didn't understand it at all. | 1 | 2 | 3 | 4 | 5 | 6 | Completely understandable. |
|---|---|---|---|---|---|---|---|

12. **What was unclear if anything?** *

13. **Did you enjoy playing Cats?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Not at all. | | | | | | | Absolutely! |

14. **What did you particularly like or dislike?** *

15. **Explain Cats to somebody who has never played it.**

16. **Do you have any suggestions for improvements to the current Microgame?**

## Doors

This section will contain only questions for feedback on the 'Doors' Microgame.

17. **About how many times did you play Doors?** *
*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Just Once | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | 10 or More |

18. **Did you understand how to play?** *
*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| I didn't understand it at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Completely understandable. |

19. **What was unclear if anything?** *

20. **Did you enjoy playing Doors?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Not at all. | ⬭ | ⬭ | ⬭ | ⬭ | ⬭ | ⬭ | Absolutely! |

21. **What did you particularly like or dislike?** *

22. **Explain Doors to somebody who has never played it.** *

23. **Do you have any suggestions for improvements to the current Microgame?**

## Battle

This section will contain only questions for feedback on the Multiplayer Battle.

24. **Did you understand how to get into a battle?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| I didn't understand at all. | ⬭ | ⬭ | ⬭ | ⬭ | ⬭ | ⬭ | Completely understandable. |

25. **What was unclear, if anything?**

26. **Did you understand how the Battle System worked?** *
*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| I didn't understand it at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Completely understandable. |

27. **What was unclear, if anything?** *

28. **Did you enjoy the Battle System?** *
*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Not at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Absolutely! |

29. **What did you particularly like or dislike?** *

30. **Do you have any suggestions for improvements to the current system?**

## Dumplings

This section will contain only questions for feedback on the 'Dumplings' Minigame.

31. **Did you understand how to play Dumplings?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| I didn't understand it at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Completely understandable. |

32. **What was unclear if anything?** *

33. **Did you enjoy playing Dumplings?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Not at all. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Absolutely! |

34. **What did you particularly like or dislike?** *

35. **Explain Dumplings to somebody who has never played it.** *

36. **Do you have any suggestions for improvements to the current Minigame?**

## Konbini Konnection

These last few questions are focused on the experience you had as a whole!

37. **Did you enjoy your experience playing Konbini Konnection?** *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| Not at all | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | It was a lot of fun! |

38. **Would you recommend Konbini Konnection to a friend? Why or Why not?**

39. **Is there anything else you'd like to say about the experience as a whole?**

40. **Are there any bugs that occurred during your session that you would like to report?**

## Thank you for playing Konbini Konnection!

Your feedback is highly appreciated! If at any time you would like to disassociate yourself with the Konbini Konnection MQP team, please send an e-mail to KKMQP@wpi.edu containing your session ID, and we will destroy all playtest data associated with you.

---

Powered by
Google Forms

# Appendix B: Session 2 Info Packet

# Konbini Konnection

Information Packet

## Welcome to Your Konbini!

Congratulations! You are now the proud owner of a Konbini (コンビニ), or Convenience Store! As the owner and sole employee of your store, you'll need to carry out all the tasks to keep your store running. You can start stocking the shelves by playing the games found below in Solo Play!

## It's time to play Cats!

You are a cat and your owner's study has become infested with mice! They seem to be hiding behind these (very breakable) objects on the shelves! Tap the objects to knock them over and search. Each object will tell you how far the nearest mouse is from its location. But be careful! If you make too much noise breaking things, the mice will all run away and you'll be the one in trouble! If you start getting too noisy, wait for a bit before you try again. Find all three mice to win!

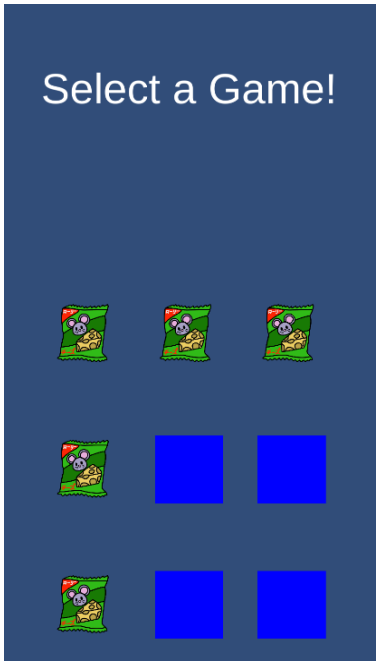
## It's time to play Doors!

You've been working your way through this hallway of doors for a while, but it looks like there's an end in sight! Make your way through each door by swiping it open! Be careful, though! If you try to open the door the wrong way, you'll be pulled back to the last one! Open all of the doors before time runs out to win!

## It's time to Battle!

Now that you've gotten used to stocking shelves, it's time to mention the other main task involved in running a Konbini! Helping customers! Good customer service helps you keep people coming to your store! However, since there are a lot of Konbinis in this part of town, sometimes other owners will come in and try to steal your customers away!


Tap an icon to challenge them to see who can help the customer check out the fastest!




When the battle begins, a grid of products will be displayed, pick one of these products to play the microgame. If the product is faded in color then it has already been taken. Win more microgames than your opponent to gain an advantage in the final challenge.

## It’s time to play Dumplings!

It’s harvest season at Hana-Dango Farm and today’s the day that you and your friend are going to pick flowers!  The farm is known for its beautiful flowers as well as its exotic Dumpling plants, and you’ve been given permission to pick until you drop! Your friend has challenged you to see who can pick more flowers, so get out there and show them how it’s done! Tap the plants as they reach full bloom to pick them!  Picking Dumpling plants will replenish your energy, while picking Flowers will raise your score! Collect more flowers than the other player to win the challenge *and* the Battle!

Dumpling plant

Flower

# Appendix C: *Dash Conductor*

As part of my work at Ritsumeikan University, I took part in a secondary project called *Dash Conductor*. The team consisted of Yang Changeun, Ishii Ryota, and Takano Yoshina who were students in the Intelligent Computer Entertainment Lab at the time. The goal of *Dash Conductor* was to create a game that took advantage of using Amazon dash buttons. On this project I served as the game designer as well as project manager.

## Concept

Player takes the role of a conductor and must send commands to each section to keep them playing. They send these commands by pressing the dash button that corresponds to that section.

## Gameplay

The player will be given 5 or more amazon dash buttons as their controller. When the game is started, the player will see different instruments in front of them that correspond to each dash button they were given. Each instrument has a bar above it that is going down at a random rate as shown in figure DASH. When the bar is in the red, that instrument is about to stop playing. It is the player's job to keep each instrument playing  by sending them a command using the corresponding dash button when the bar is in the red. Each time the player successfully sends a command to refill an instrument's bar, they receive points. If they fail to send the section a command, they receive no points and that section stops playing for the rest of the song. Only one command can be sent at a time and a command cannot be sent if a bar is green as the section is still executing the previous one. The goal of this game is to keep the orchestra playing until the end of the song or until they lose.

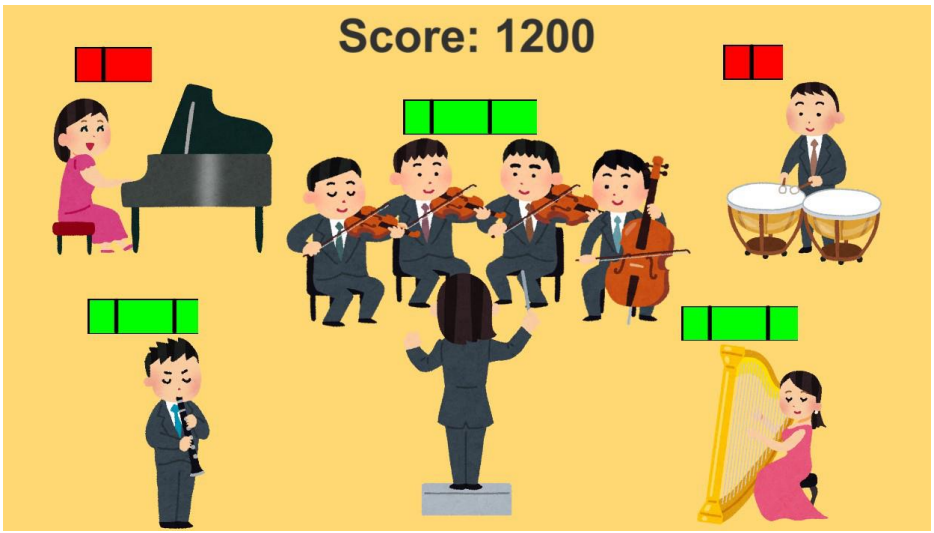



Figure DASH: Screenshot of *Dash Conductor* gameplay

## Reasoning

This game takes advantage of the dash button in two different ways. The first is that it requires players to switch between dash buttons as they try to keep each section playing. A starting number of 5 dash buttons was decided on because it was observed that an average person would not comfortably be able to hold 5 dash buttons in their hands and still be able to press them. The second way this game takes advantage of the dash button is that it uses the signal problems and reset time of the dash button as mechanics for the game. The game was built around the known delay between when the dash button is pressed and when the signal is received. The minimum time between when a bar turns red and when it is empty takes this delay into account along with the average reaction time, allowing the player enough time to press it and refill the bar. The time between when the dash button is pushed and when it can be pushed again is also taken into account when designing this game as the minimum time between when a command is received and the bar turning red is greater than the reset time.