November 2013

Railroad Runner

Alexander Jungnom Kim
Worcester Polytechnic Institute

Haoyang Zhang
Worcester Polytechnic Institute

Jennifer Renee Baulier
Worcester Polytechnic Institute

Merrielle Diane Ondreicka
Worcester Polytechnic Institute

Runzi Gao
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation
Railroad Runner
A Major Qualifying Project
submitted to the Faculty of the
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
on November 8, 2013

Submitted By: Jennifer Baulier, Alexander Kim, Merrielle Ondreicka, Runzi Gao (Jack), Haoyang Zhang (Lance)

Advised By: Professor Robert Lindeman
Abstract

Many people in the world regularly visit the same places. Trains are a common means of transportation for these people, especially in Japan where nearly everyone rides trains for their daily commutes. However, these people may never get to know or even meet each other despite their proximity. In response to this, a project called Railroad Runner was designed to allow users riding the same train to participate in multiplayer games together while remaining anonymous should they so choose. Users connect to a local server using web-enabled devices such as smartphones and participate in various activities together.

The primary purpose of Railroad Runner is to allow users to pass the time on train rides in a casual, social fashion. The primary means for this interaction is a collection of four minigames, which is supplemented by a fifth, single player game as well as social features such as a chatroom and viewable userscores from other players. Users are organized into groups based on the train routes for their chosen commutes, ensuring that they can play games with others who are physically close by. These groups are joined by scanning QR codes tied to certain train lines, but previously visited groups can also be rejoined by selecting from a list. By allowing users to interact with others nearby, Railroad Runner encourages its users to socialize with other passengers on their train.
Table of Contents

Abstract ................................................................................................................................. ii
Table of Contents .................................................................................................................. iii
List of Figures ........................................................................................................................ vi
Chapter 1: User Stories .......................................................................................................... 1
Chapter 2: Project Overview ................................................................................................. 3
  2.1. Project Goals .................................................................................................................. 3
Chapter 3: Overarching Structure ........................................................................................ 5
Chapter 4: Game Descriptions .............................................................................................. 10
  4.1. Treasure Trap .................................................................................................................. 10
    4.1.1. Treasure Trap Networking ...................................................................................... 12
  4.2. Creature Creator .............................................................................................................. 12
    4.2.1. Creature Creator Networking ............................................................................... 13
  4.3. Bomb Pass Panic ............................................................................................................ 14
    4.3.1. Bomb Pass Panic Networking ............................................................................... 14
  4.4. Elemancer ...................................................................................................................... 15
    4.4.1. Elemancer Networking ......................................................................................... 17
  4.5. My Train ......................................................................................................................... 17
    4.5.1. Gameplay .................................................................................................................. 18
Chapter 5: Planning and Background .................................................................................... 21
  5.1. Decision Process ............................................................................................................ 21
  5.2. Designing Each Game .................................................................................................... 21
  5.3. Context and Audience ................................................................................................... 21
Chapter 6: Game Design and Prototyping ............................................................................ 23
  6.1. References ..................................................................................................................... 23
  6.2. Treasure Trap .................................................................................................................. 26
  6.3. Creature Creator ............................................................................................................ 27
  6.4. Bomb Pass Panic ............................................................................................................ 27
  6.5. Elemancer ...................................................................................................................... 29
  6.6. My Train ......................................................................................................................... 29
  6.7. Summary ........................................................................................................................ 32
Chapter 10: Post-Mortem.................................................................................................................. 51
  10.1. Goals Completed...................................................................................................................... 51
  10.2. Ownership............................................................................................................................... 51
    10.2.1. Merrielle Ondreicka ........................................................................................................ 51
    10.2.2. Jennifer Baulier ................................................................................................................ 52
    10.2.3. Alex Kim ........................................................................................................................... 53
    10.2.4. Haoyang Zhang (Lance) ................................................................................................ 53
    10.2.5. Runzi Gao (Jack) .............................................................................................................. 53
  10.3. What Went Well ....................................................................................................................... 54
  10.4. Problems Encountered ........................................................................................................... 55
  10.5. What Was Cut ......................................................................................................................... 55
10.6. Future Iterations .............................................................................................................. 57
Chapter 11: Conclusion ........................................................................................................... 60
Appendix A: Meeting Minutes
Appendix B: Database Structure
Appendix C: Goal Creation
Appendix D: UI Mockups
Appendix E: Initial Game Ideas
Appendix F: Scheduling
Appendix G: My Train Prototype
Appendix H: My Train Final Events
Appendix I: Art Asset List
List of Figures

Figure 1: Train passengers playing with their smartphones .................................................. 3
Figure 2: The Railroad Runner logo ................................................................................. 5
Figure 3: Railroad Runner's Log In Screen ....................................................................... 6
Figure 4: Railroad Runner's Train Room Screen ............................................................... 6
Figure 5: Railroad Runner's Main Menu ............................................................................ 7
Figure 6: Railroad Runner's User Profile Screen ............................................................... 8
Figure 7: Railroad Runner's Host New Game screen .......................................................... 9
Figure 8: A traditional game of Amidakuji ...................................................................... 10
Figure 9: Railroad Runner's version of Amidakuji - Treasure Trap ................................. 11
Figure 10: Creature Creator from the first player’s perspective .......................................... 13
Figure 11: Creature Creator’s results screen .................................................................... 13
Figure 12: Bomb Pass Panic ............................................................................................. 14
Figure 13: Elemancer Multiplayer Mode ......................................................................... 16
Figure 14: Elemancer Single Player Mode ....................................................................... 17
Figure 15: Railroad Runner's My Train screen ................................................................. 19
Figure 16: My Train's Store Screen .................................................................................. 20
Figure 17: Processing prototype for Elemancer under the working title "Wizard" .......... 29
Figure 18: The My Train Prototype Board ....................................................................... 30
Figure 19: Playtesting My Train with volunteers from Takemura Lab ............................... 32
Figure 20: Network UML Diagram .................................................................................. 38
Figure 21: Initial sketch of the Bomb Pass Panic character .............................................. 45
Figure 22: Bomb Pass Panic's background graphic ............................................................ 45
Figure 23: Sheet for Bomb Pass Panic's explosion animation ........................................... 46
Figure 24: Sheet containing Creature Creator's various options ...................................... 47
Figure 25: Treasure Trap's "boot," which awards zero points .......................................... 47
Figure 26: Treasure Trap's characters .............................................................................. 48
Figure 27: My Train's background loop .......................................................................... 48
Figure 28: Shinkansen Model n700 Train in Japan ............................................................. 49
Figure 29: My Train's Shinkansen-inspired vehicle ............................................................ 49
Figure 30: Elemancer's spell hierarchy chart .................................................................... 50
Figure 31: Rough UI mockup
Figure 32: Rough UI mockup ctd.
Chapter 1: User Stories

Junya is an average salaryman living in Japan. He needs to take the train to commute to work every day, and he takes the same train at the same time. Often Junya sees the same people, most of whom are passing the time by playing games on their smartphones. They all seem oddly familiar with each other. Junya wonders what it would be like to say hello to them. However, since talking on the trains is generally frowned upon, Junya sits quietly by himself.

One day, Junya sees a new advertisement on the train. It is a large poster with only a QR code. Curious, he uses his phone to scan it. It opens up an application called Railroad Runner. It is free, so he downloads it and creates a user account. He sees a main menu with a list of minigames; these minigames are played with other people on the same train! All of the people Junya has seen on their smartphones are getting to know each other by playing this game.

Junya sees a list of games that are starting soon. He clicks on one labeled Treasure Trap. As the game loads and connects everyone, it shows a splash screen with instructions for how to play. The game starts and goes rapid-fire - everyone only has ten seconds to act. Junya gets lucky and hits the jackpot score. The game gives him 70 Gold. The game ends, and he returns to the main menu.

Mori is a girl who has been playing Railroad Runner for a while. Every week, she tries to get the highest score on My Train. She is slowly getting better and better. She carefully invests her gold throughout the day for maximum efficiency and distance. If Mori buys too much speed, her train will break down by the end of the day. If she buys too little speed, it will not go very far and her score will be low. She buys cargo to earn a steady income. So far, she has the highest score of any user she has seen - except one user named Jeff266. His total overall score, or userscore, is nearly double hers. She has played minigames and chatted in-game with him before, and he almost always wins. She is determined to keep playing until one day she can beat his userscore.

Hayato is not a gamer. He is just a guy looking to pass time on the train. He gets on the Hankyu line, and the only other person in the car is a cute girl on her smartphone. Hayato only has a couple stops before he has to switch trains, but he wants to say hello. He figures that since Railroad Runner is popular, she might be playing it. He goes into the game’s chat room and sees
her there. They talk for a bit and play Elemancer together, he gets into the game and enjoys it, and when he finishes he is near his stop. Before getting off the train, he sends the girl his phone number.
Chapter 2: Project Overview

The team came up with this project idea from riding the trains in Japan. In Japan, railways are a major means of passenger transportation. Many Japanese people play with their smartphones during their train rides, as shown in Figure 1. However, very few of them ever interact with others around them. Because of this, the team thought it would be a great idea to build an application that allowed people on trains to interact anonymously with each other using their smartphones. The initial brainstorming can be found in Appendix C.

2.1. Project Goals

With the hope of creating a new and interesting way for people to interact while traveling by train, the team decided upon six detailed goals:

1. Develop an android application that allows passengers on each train to play games and interact with each other.

2. Create 3-5 multiplayer minigames.

3. Design a persistent game in the application that ties all the minigames together. Players can earn gold by playing minigames, and they can use their gold to make progress on the persistent game.
4. Implement a user account system that stores basic user information including high scores in games and a record of with whom they have played the most games.

5. Allow players to join a Train Room by scanning QR code on their train.

6. Provide the users the options of hosting a game, joining an existing game, or joining the chat room once the user enters a Train Room.

The team chose these six goals for very specific reasons. The Android operating system was chosen as the development platform because it is currently the most popular mobile platform\(^1\). Compared to iOS, Android is much more open to independent developers as well as being free to use.

The time that passengers spend on a given train ride can vary from as short as 10 minutes to as long as several hours. The majority of passengers do not have enough time on the train to play a long game, so the minigames included in the application were designed to be relatively quick to play.

The persistent game serves as a motivation for players to play more minigames. Points earned from playing minigames can be used to make progress in the persistent game, and players will be rewarded for making the most progress on a weekly or monthly basis.

The user account system was meant to be as simple as possible so it does not draw too much of the player’s attention from playing the game. As one of the ultimate goals of this project is to encourage people to socialize, the system can store information such as how many times the user has played with other users.

Chapter 3: Overarching Structure

Railroad Runner has a complex internal structure, but it is organized in a way to make it intuitive for users. The application icon is shown in Figure 2. The first time the user opens the application, they are greeted with a screen asking them to make a user account. They are presented with a form, shown in Figure 3, that contains fields for a username and a password. This information is then sent to the database and the user is logged in. Next, they are asked to join a “Train Room”. A Train Room groups users together based on their location. Users in a Train Room can only interact with other users in the same room. A Train Room is entered by scanning a QR code or picking from a list of rooms that the user has been in before. This is shown in Figure 4. For comparison, the original mockups of these interfaces can be found in Appendix D.

Figure 2: The Railroad Runner logo
After joining a room, the user is sent to the main menu. The main menu, shown in Figure 5, has several paths the user can choose. They can play the persistent game, called My Train, they can host or join minigames, they can look at their profile and edit their information, they can view other users’ information or they can chat with other users.
If the user chooses to look at their profile, they will see a page with their username, userscore, and buttons to change Train Rooms, change their password, log out, or delete their account. The profile screen is shown in Figure 6. If the user elects to change Train Rooms, they are brought back to the screen where they can scan a QR code or enter a Train Room manually. If the user wants to change their password, they are brought to a page that asks them to enter their old password once, and the new password twice. Choosing to log out will automatically log the user out of the application. The next time the user starts the application they will have to enter a username and password. Finally, if the user chooses to delete their account, they are first required to enter their password and confirm that this is really what they wish to do.
If the user wishes to play a minigame, they can either host a game or join a game that another user has created. To host a minigame, the user first clicks "Start New Multiplayer Game". This opens a screen, shown in Figure 7, that allows the user to select which game they want to host. Next, the user must state whether they are ready to play or if they would like to cancel. A player who is currently ready for one game cannot join or host any other minigames. Once the user clicks "Ready", the button that says "Start New Multiplayer Game" switches to show which game the user is hosting. Once the minimum number of players have joined the game and stated they are ready, a button appears that allows the host to start the game. If the host does not press this button, the game will start automatically in 20 seconds.
Figure 7: Railroad Runner’s Host New Game screen
Chapter 4: Game Descriptions

The railroad runner application contained four mini games and one persistent game. The mini games were all designed to accommodate multiple players, and one of them also included a single player mode. All minigames are meant to take a relatively short amount of time to play. The persistent game is played alone by the user and lasts through the course of a week. Minigames include Treasure Trap, Creature Creator, Bomb Pass Panic, and Elemancer.

4.1. Treasure Trap

Treasure Trap is inspired by a game called Amidakuji in Japanese, or Ghost Leg in English. An example of this game is shown in Figure 8. Amidakuji is a lottery-style game designed to match objects in one set to objects in a second set at random. It is a popular game in Japan used to distribute things among people. Players can win this game fairly and with an element of impartial randomization. Amidakuji was both fun to play and simple to program, which were deciding factors for including it in the application.

![Figure 8: A traditional game of Amidakuji](http://images.apple.com/jp/downloads/dashboard/games/images/amidakuji_main2006122711191202.jpg)

---

Playing Amidakuji in real life is easy. First, players write their names on the top of a board and draw straight vertical lines below the names. Second, the names are hidden so that people do not know which player is on which line. Third, each player adds “legs” on the board. Legs are straight horizontal lines connecting two adjacent vertical lines. A leg cannot touch any other leg. Finally, a path is traced from top of the vertical lines to the bottom, moving horizontally to follow every leg encountered along the way. Once players arrive at the bottom of the vertical lines they will see which element they have been paired with. Because of the nature of Amidakuji, no two players will finish on the same line.

Treasure Trap is different from Amidakuji is several ways. It has a scoring system so that players can compete with each other. Coins were added on the lines to provide bonuses for players who pass by, and treasure chests were added at the endpoints where a larger bonus is collected. These modifications are shown in Figure 9. The player with the most points at the end wins the game. Another difference is that every player is able to draw lines on the board. When players draw their lines on the board, the system sends the line information to the server and shares the status with everyone connected. This feature allows all players to see the lines as they are being drawn, as well as making the game more interactive and competitive.

Figure 9: Railroad Runner's version of Amidakuji - Treasure Trap
Treasure Trap’s gameplay is rather similar to the original Amidakuji game. Here is the flow for the game from beginning.

1. When the player starts the game, the application connects to the server. The player gets a player ID that assigns their position.
2. The player has 10 seconds to draw lines on the board. The player can draw up to five lines.
3. After 10 seconds, all players follow the paths down the lines at the same time. When the player touches a coin, points are added to their score based on the coin value.
4. At the end of the line, the player will get some point bonus depending on their final location.
5. When everyone reaches the end, everyone’s scores are shown on the results screen. The player with the most points wins the game.

4.1.1. Treasure Trap Networking

The network structure for Treasure Trap is straightforward. First, the client sends a login request to the server so that the server knows that the player has joined the game. Then the server sends a package back and requests for acknowledgement packages that keep the connection alive. When the game interface appears, the server sends all users the position of the randomly generated coins and chests, and tells the clients to start the timer at the same time. Then the player can draw lines on the screen. When the user draws a line, their client tells the server. Then the server broadcasts this new line to everybody in this instance of the game. At this point, every player should see the same game state. When the game is finished, the server will create a new set of coins and chests for the next game.

4.2. Creature Creator

Creature Creator is inspired by casual artistic games often played by children. In such games, players work together and create unique characters. The creatures are created by combining body parts drawn by different people without looking at what the others are drawing. The results are usually quite entertaining.
Railroad Runner’s Creature Creator is a three-player game in which users work together to create humanoid creatures. Each player is assigned the head, body or legs, and then they can select their favorite option in their assigned section. This selection process is shown in Figure 10. Additionally, each player can only see their own part of the body; the other two sections are hidden until the end of the game. Once all three players have made their decisions, their selections are combined to create the final creature. An example of a finished creature is seen in Figure 11. Since Creature Creator does not have a losing condition, participating players always earn a small amount of gold toward My Train.

![Figure 10: Creature Creator from the first player's perspective](image1)

![Figure 11: Creature Creator's results screen](image2)

4.2.1. Creature Creator Networking

When each player joins the Creature Creator game, an instance of the Kryonet client is created. Then the client initializes a client listener, and sends a Log Info packet when connecting to the Creature Creator game server. If the server listener hears the Log Info packet, it first puts the requesting client to its clients list, and then sends back a message with a part ID to the game client which assigns the player one of the three parts (head, body, or leg).
During gameplay, whenever a player switches their body part, their game client sends a message to the server so that the server always knows each player’s current selection. At the end of the game, the server broadcasts the current state to all players so that everybody sees the resulting creature.

4.3. Bomb Pass Panic

Bomb Pass Panic is a minigame intended for four to six players, and it is based on a children’s game called “hot potato.” Instead of a potato, players in this game pass around a bomb by swiping on their device’s touch screen. The angle of the swipe is used to determine which player receives the bomb in each pass. Each round of gameplay will end after one of two conditions: the time limit for the round is reached or one player holds the bomb for too long without passing. A numerical timer is not displayed onscreen, but the bomb begins to flash when only five seconds remain in the round. When the bomb explodes, the last player holding it is eliminated and a new round begins, and the game continues until only one player remains. Bomb Pass Panic is shown in Figure 12.

![Figure 10: Bomb Pass Panic](image)

4.3.1. Bomb Pass Panic Networking

In Bomb Pass Panic players are assigned player numbers based on the order they reached the loading screen. All players are labeled with their own numbers, and the starting player in the
first round is always player one. In each subsequent round, the starting player is the remaining player with the lowest player number.

In the client code, players have an index that is used for their sprite, position, and other info that is stored in various arrays. When a player passes the bomb, both their player index and the index of the player being passed to are sent to the server. The connections of all players in a game are stored in the server in a HashMap with a GameID as the key. A GameID is a unique value that indicates an individual game session. The GameID is sent to the server along with an integer that indicates that this is a pass action. The game checks whether the player attempting to pass is the player currently holding the bomb, and the player passed to is then set as the current player by the server. A packet with the info on the player passed to, along with the integer indicating the pass action is then sent on to the client side listeners of every player connected to that GameID. The client’s listener will interpret this and send the player number to a function in the client code that will translate this to the in game index and handle the pass as needed.

A similar procedure to what is done with passing is used to tell the game when a player did not pass on time. In this case, the only info needed to be sent is the GameID, the player that is out, and an integer indicating that they ran out of time.

### 4.4. Elemancer

Elemancer is a memory game based around the four elements that often appear in fantasy and mythology: fire, water, wind and earth. The primary game screen displays a two by three grid of stars, and players cast elements by tapping four of these six stars to create patterns. Each element has its own corresponding pattern, and a color-coded animation plays when a player enters a valid pattern.

Elemancer has two modes of play: single player and multiplayer. In multiplayer mode, players take turns attacking and blocking. This is shown in Figure 13. To start, the attacking player is told to cast one spell of any element. The blocking player is briefly shown this spell and must memorize it. Then the player counters with the correct element. Fire beats earth, earth beats wind, wind beats water, and water beats fire. If the player casts the incorrect element or takes
over thirty seconds, they lose a life. After attempting to block, if the player has lives remaining, he or she will become the attacking player.

![Figure 11: Elemancer Multiplayer Mode](image)

Every six rounds, the number of spells the attacker must cast goes up by one. The blocking player must counter the spells in the same order they appeared. For example, if the opponent casts fire, fire, earth then the blocking player must cast water, water, fire. Play will continue until either player is out of lives.

Single player is similar to multiplayer except the player never attacks. An AI opponent will attack the player and they will always block. Since the player will always be able to play until he or she loses, unlike multiplayer that may stop when the other player loses, single player is a survival game. The player still only has two lives, and the number of spells they need to remember still increases by one every six rounds. The user interface for single player Elemancer can be seen in Figure 14.
4.4.1. Elemancer Networking

The Elemancer server keeps a HashMap of player connections with GameIDs and keys. A GameID is a unique value that indicates an individual game session, which is only important in the multiplayer version of Elemancer. Once all players connect, each player will be told whether there are one or two players, and if there are two players, whether they are the starting player. In game, integers are used to indicate specific intended actions and GameIDs are always included in the information a packet contains.

In multiplayer, when an attacking player casts spells they need to send a packet indicating that they are casting a spell and which spell was sent. This will be sent to the other player in the game with the same GameID. If they try to cast something that is not a spell or take more than thirty seconds, they will send a packet indicating failure. If the blocking player misses or takes too long blocking, their client will send a packet saying that they missed.

4.5. My Train

My Train is a single player persistent game that runs alongside all of the other minigames within the application. Each user is given a virtual train that runs by itself until it breaks down. Players influence their trains by using the gold won from the other games to purchase various resources for the train to keep it running and increase its performance. My Train is not an inherently competitive game, though users are able to view the progress and scores of other
players for comparison. Additionally, although My Train helps motivate players to participate in the minigames, it has also been designed in such a way that players can ignore it entirely should they so choose. Its purpose is to supplement the minigames rather than distract from them, so any given player is never required to participate in it.

4.5.1. Gameplay

When a user first enters the application, they are given a virtual train. This train continuously moves on its own unless it has broken down, usually requiring no direct input from the player. Thus, My Train’s game scene is more of a stats screen, as shown in Figure15. Rather than being in direct control of the train, each player is in control of their train’s resources. These resources are:

1. Health: the train’s physical state. When a train’s health reaches zero, it is no longer able to move until its player repairs it.
2. Speed Points: determines how far the train will move in a given period. The more speed points a train has, the farther it will go. However, having more speed points also causes the train’s health to deteriorate more quickly.
3. Cargo: a booster resource. Carrying more cargo multiplies a train’s final score, and it causes the train to earn gold over time.
Players can buy these three resources in any quantity they can afford. These resources are purchased through a dedicated store screen (shown in Figure 16) using gold, which can be obtained in several ways. The primary method for earning gold is to play the other minigames within the application, each of which has its own reward depending on the difficulty of the game and how long it takes to play. Additionally, carrying cargo causes a train to accumulate some gold over time. Another feature in the game is an event system. Every so often, each train has a chance of encountering a random event. Events can have a variety of effects, some positive and some negative, and some events will even have both. The full list of events included in My Train can be found in Appendix H.
Finally, My Train globally ends and resets on a weekly basis. When the game ends, each player is given a score based on how far their train moved as well as how much cargo they were carrying. This is added to an overall player score that can be viewed by other users, and rewards are available for achieving certain weekly milestones. The next day, My Train resets back to its initial state with each player being given a new train for a fresh start.
Chapter 5: Planning and Background

A long period of planning preceded any of the actual coding that went into making Railroad Runner. The team had quite a few ideas so it was necessary to ensure that the core of the project had a clear sense of purpose. There were also restrictions to keep in mind, one of the biggest being time.

5.1. Decision Process

The initial decision-making process in Railroad Runner’s development revolved around making the application varied and interesting, but still capable of being built in only a few months. The decision to build the application for Android as opposed to iOS was made based primarily on this criterion. The Unity engine is capable of building applications for both operating systems, but most of what Unity offers simply was not necessary. Ultimately, it was decided that the application would be built for Android using Eclipse, ADT, and a set of game-friendly libraries called AndEngine.

5.2. Designing Each Game

Since the games within the Railroad Runner application were all so different, each one was designed with its own unique criteria. However, some common elements were featured in prototyping and testing each game. Primarily, one of the most important factors in designing each game was determining whether it would be accessible to both Japanese and American players. Some games were based on Japanese concepts and others were based on American ideas, but in the extreme cases, the games were altered to be more recognizable by an international audience. The initial design for the games can be found in Appendix E.

5.3. Context and Audience

While this application is available to anyone with an Android phone, the primary audience consists of commuters, particularly younger commuters who are more likely to make use of their smartphones to pass the time during train rides. Daily train travel is an important part of life for many people in Japan. Most commuters can spend anywhere from thirty minutes to
several hours riding trains every day for various purposes such as work, school or leisure. While these train rides often involve many people being forced together into a small space, the daily commute remains a very private experience for most. Though there are some exceptions, most train commuters tend to be very quiet and keep to themselves. However, this is not a problem to be solved; it is merely a requirement that this application has to accommodate.
Chapter 6: Game Design and Prototyping

When creating a game, it is very important to decide on the exact mechanics that will go into the completed version. In order to finalize these details it is often necessary to conduct user testing or ask for feedback. Such testing occurs before any major work on creating the games takes place.

6.1. References

Several games provided inspiration for various parts of Railroad Runner. Mario Party is a very well known example of a game that provides a large suite of minigames for its players. In Mario Party, the minigames give the player currency that he/she can use in a competitive board game that lasts a certain number of turns. This acted as evidence that a series of small games supporting an overarching game can be enjoyable to a large market of people. Unlike Mario Party though, the team decided not to force all players to participate in the overarching game. If so desired, players in the application could play solely minigames and chat with other users.

One of the minigames, Bomb Pass Panic, is a variation of both the American game hot potato and Hot Bob-omb, a game found in Mario Party 2. The team was unsure how many people in Japan would be familiar with the concept of hot potato. It was decided that using a bomb would make the goals of the game more clear. Hot Bob-omb may be a source of the inspiration for using a bomb instead of a ball or actual potato. The idea of characters passing a bomb between each other trying not to be the one stuck with it is also often seen in cartoons, which was another source of inspiration for this idea.

Two other video games known for their suites of minigames are WarioWare and Bishi Bashi. Many of the minigames in these titles are extremely simple to play and focus on silly imagery to make them memorable. The success of these two games demonstrated that minigames do not need to be overly complicated to remain enjoyable. Similar bizarre and silly ideas were found in quite a few of the team’s early concepts. Creature creator is less of a game and more of a fun activity to do with friends. Similarly to the games in WarioWare and Bishi Bashi, the thing that ultimately stands out about Creature Creator is the silly imagery. Giving players the ability to go back and view creatures that they made emphasizes this.
Our overarching game drew inspiration from several sources. The two main sources of inspiration were the computer games Faster Than Light and Oregon Trail. Another source of inspiration was the game Puzzle Pirate. The games in Puzzle Pirate were each meant to simulate a different action that would be done on a ship, and how players were doing in one game affected how hard it was for people playing the other games. The team considered implementing this, but thought redesigning the games would be ineffective because the overarching game was supposed to be optional.

The early concept for My Train was much more closely inspired by Faster Than Light. In Faster Than Light, the main mechanics involve controlling the crew of a spaceship and assigning different crewmembers to take on various jobs as needed. The player then travels around facing obstacles and ultimately tries to reach a final location. In early iterations, the minigames were going to reward the user with crewmembers and other resources. Crewmembers could be assigned to different roles on the train. The random events occurring that occur in My Train were a loose parallel to the obstacles in Faster Than Light.

Some team members were afraid that users would not want to allocate crewmembers on a short train ride. They were also worried that it would be complicated for casual users, and divert too much attention away from the minigames. Therefore, the team instead developed a system where the minigames provide money that can be used to buy resources. Events come into effect, and then players have to choose how to deal with the results based on what they have.

At this point, the event system in My Train held a closer resemblance to the event system in The Oregon Trail. In The Oregon Trail, the player’s route will randomly affect their supplies. The player has very little say in the outcome of these events, just like in My Train. The events system in My Train also holds resemblance to random event cards as commonly seen in board games.

When the design for My Train’s game mechanics had been finished, the game was still intended to be multiplayer. However, there were major concerns with the multiplayer aspects of the game. The team feared that any given player would not have an effect on the game, and that the winning train would simply be determined by whatever room had the most players. The solution to this problem was inspired directly by the Streetpass games on the Nintendo 3DS. In these games, every time a player meets another person in game they get closer to completing
some personal goal. However, the player can choose how exactly the person they met will help them. For example, in one Streetpass game the player collects puzzle pieces in order to complete various puzzles, and starts with one piece at random. Every time they meet someone, they get to choose a puzzle piece that the player they met had collected and then add it to their own collection. The idea of using interactions with other people to help the player reach their goal solved the concerns the team had involving the overarching game. As a result, it was decided that the train game should reset every week instead of daily so players could have time to work towards that goal. The train game was turned into a single player game, but the currency was still gathered through multiplayer minigames. In that way, the player could have a goal that they worked towards individually, but they would still need to interact with others by playing minigames if they wished to do well.
6.2. Treasure Trap

Treasure Trap was one of the first games to be prototyped due to its relative simplicity. Similarly, many gameplay details were unclear during most of the brainstorming stage. For example, a decision had to be made as to whether the player icons should be at the top or the bottom of the lines, whether the players should move down into goals, and whether players should collect falling coins in their own specific end areas. Would the line placement phase be turn-based or real-time? Moreover, would the lines placed by other players be visible during the placement phase, or would they only appear afterward? Despite these early inconsistencies, heavy discussion during the design period led to the elimination of all but the final variation of the game.

In the final version of Treasure Trap, each player starts at the top of a vertical line. When the game begins, players have ten seconds to place up to five horizontal lines between the vertical lines. All players place their lines at the same time, and all lines are immediately visible to the other players. Coins are randomly placed along the vertical lines, and any players that pass over them gain ten extra points per coin. Once the line drawing phase ends, all player-icons begin to move down the vertical lines. Anytime an icon reaches a horizontal line, they must travel across it to the connected vertical line and then continue downward again. Finally, when the players reach the bottom of the vertical lines, they each receive a bonus depending on which line they finished on. Three of the lines end in treasure chests that give ten, twenty or thirty points and the one remaining line ends with a boot that gives zero points.

The initial prototype of Treasure Trap was first played by the team in front of the selected group of testers to give them a basic understanding of the game's mechanics. Afterward, the four volunteer testers took over gameplay. The playtesters appeared to be consistently engaged in the game, and they all displayed a very competitive attitude. One of the first changes the team decided to make was the placement of the end goals. Originally, goal placement was going to be entirely random, but after a few rounds of playtesting, it was decided that the highest value goals should always be placed on one of the center lines so that no one player has to traverse all the way across the board to reach them. Once playtesting concluded, one of the players expressed the concern that while the game was hectic and fun to play the first time, it was difficult to feel a
sense of improvement on repeat playthroughs. As a result, the number of lines each player can place was reduced to a maximum of five, encouraging a greater tactical focus on line placement.

6.3. Creature Creator

Creature Creator’s mechanics were simple enough that no playtesting was needed. The players simply need to select their favorite image for their assigned section of the creature. Ultimately it was decided that Creature Creator was more of a fun social activity than a game, and as such, there are no winning or losing conditions.

6.4. Bomb Pass Panic

Bomb Pass Panic was the first minigame to be implemented, and as such, it had one of the earliest playtesting sessions. The initial prototype was built in Processing, a programming language focused on graphics, allowing the team to test the single player functionality of the game. Even before reaching the multiplayer stage, it became apparent that there was a variety of options for several gameplay mechanics. Before moving forward, the team decided it would be best to continue playtesting through a physical metaphor for the game rather than the Processing demo.

The two primary gameplay elements examined in playtesting were adding a catch mechanic, and determining whether the game would last until there was only one winner, or if it would stop after one loser. With the catch mechanic in place, players would have to perform an action such as tapping their screen to receive the bomb when it was thrown to them as opposed to catching it automatically. This introduced the possibility of players being able to drop the bomb, thus making it more likely for them to eliminate themselves and preventing spam gameplay. To simulate this mechanic with the physical prototype, testers played several rounds of traditional hot potato with varying rules. Players rolling a ball to each other in a sitting position simulated the lack of a catch mechanic, since each player would receive the ball with no catching action required. Conversely, players tossing a ball to each other in a standing position simulated the inclusion of the catch mechanic, where the actual catch paralleled the action required to receive the bomb.
Game duration was tested physically in the same manner it would have been tested in a programmed prototype. Testers played several rounds of the game, some of which stopped after the first elimination and some of which continued until only one player remained. To test this mechanic, two rounds of single elimination were played, one with each of the catch mechanic options. The team asked the testers which version they preferred, and then the testers played a single winner game with their chosen mechanic.

Based on the team's observations, the testers appeared to be having fun throughout each stage of gameplay. A maximum of three seconds between passes proved to be the optimal time limit, as was thirty seconds total for each round. After the first two rounds of play, testers agreed that including a catch mechanic added difficulty to the game but also made it more enjoyable. They also pointed out that the lack of a catch mechanic would enable players to simply swipe on their screens without paying attention. Luckily, the testers enjoyed both versions of gameplay. The catch mechanic was ultimately cut out for a few reasons, primarily because other more important features had to be implemented first, and latency and lag made the catching difficult to synchronize across devices.

The final playtest lasted for multiple rounds, removing the losing player each time and beginning a new round until only one player remained. Afterward, the testers expressed a slight preference for the single elimination games due to their shorter duration. However, this may have been because the longest round was played last, by which point the testers may have already felt as if they were playing for too long at once. Additionally, in the actual game, losing players are kicked from the game and do not have to wait while the other users finish. In the prototype the other testers would have to idly stand on the side. Despite the testers' preference, the team was worried about a particular case in which the starting player does nothing, causing the entire game to end after three seconds and without giving the other players a chance to do anything. Because of this, the team decided that the games should continue until only one winner remains for the sake of a better overall user experience. Finally, despite initial misgivings, the longer games still proved to be enjoyable during the final demonstration of the application.
6.5. Elemancer

The initial prototype for Elemancer was built in Processing, similarly to Bomb Pass Panic. In the interest of time, this prototype only displayed the features of Elemancer's single player mode. The interface for this prototype is shown in Figure 17. After a short round of initial playtesting, the team agreed that the mechanics of the game were solid and required no substantial changes. No further prototyping was necessary, and any future changes would be small balancing details rather than large alterations to gameplay. However, while Elemancer's core gameplay was unchanged, playtesting did lead to a series of changes to the game's user interface to fit with a more uniform art style and to be more accessible to the players.

![Figure 15: Processing prototype for Elemancer under the working title "Wizard"](image)

6.6. My Train

The initial testing of My Train called for the most complicated prototype built by the team. Creating a fully functional prototype in Processing or another language would have taken too much time, and as a result, My Train was adapted from a single player game into a multiplayer board game for testing. The game board can be seen in Figure 18, and the written game instructions are located in Appendix G. While the board game was multiplayer, there were no directly competitive elements of gameplay. The multiplayer component of the board game
simply allowed multiple testers to play at the same time, and it was meant to simulate multiple players interacting with their own trains concurrently.

![Figure 16: The My Train Prototype Board](image)

To accommodate the nature of a board game, all of My Train's time-based elements had to be converted to a turn-based system. The test games lasted for ten turns, each turn roughly representing one hour of real time. A turn consisted of four main activities: rolling for an event card, playing a minigame, buying resources, and moving. At the beginning of their turn, each player had a 50% chance of getting an event, which would either have an instant effect or last for a number of turns. Events could affect resources owned, distance traveled, prices in the shop, or a variety of other elements. Some events also allowed players to select from several different outcomes, and others were determined by random chance.

After the event phase, the player could play one minigame. This rule was enforced in the interest of time, and the team acknowledged before testing that most players in the real application would be able to play more than one minigame per hour. The player could select one minigame from the list of planned games, and then the player would then roll a virtual die of varying denominations, determining win or loss by random chance. This rule was implemented to allow faster play, but the chance of winning was decided based on the number of players
required for each minigame as well as how likely it was for the average player to win or lose. Gold rewards were determined on the same principle, with higher rewards going to longer games or games the average player was less likely to win. Additionally, the minigame phase of the prototype featured another mechanic that was later cut: challenge mode. Players could choose to participate in a challenge mode for their selected minigame as long as they had enough gold to make the required wager. After wagering, if a player won their game, they would get their wager back and their winnings would be doubled. If they lost, however, they would lose their wager and receive no reward.

In the next phase, the player could spend their gold to buy resources from the My Train store. Provided they had enough gold, players could restore their train's health up to 100, buy up to a total of 20 speed points, or buy as much cargo as they could afford. After the buying phase, the player calculated their new resource totals and proceeded to the movement phase. For the sake of the board game prototype, each speed point moved the player's train one space on the board and removed one health. If the train had less health than its number of speed points, it would stop after running out of health. Additionally, if a player began their turn with no health, their train would not be able to move at all that turn, even if more health was purchased. Finally, at the end of the tenth turn, each player's score was determined by an equation that used distance traveled and cargo held as its variables.

The board game prototype of My Train was played twice, once with the development team and once with volunteer testers. The first playthrough was intended to eliminate any glaring issues before an open test, and as such only members of the team played it. During the second playthrough, testers responded well to the game. The second playthrough is shown in Figure 19. Most changes made after testing were minor alterations for balance, but two significant improvements were made.
First, the testers did not think events happened frequently enough. Even though events could sometimes be negative, the testers thought they added an interesting element of surprise. This also led the team to realize that users with the actual application may only play for short periods, and as such would experience events even less frequently than in the board game prototype. As a result, events were made more likely to occur, but not so common that they would become tedious.

Finally, the testers realized that cargo was largely useless until score calculation at the very end of the game, and as such, players might not want to buy any throughout the majority of gameplay. One of the testers suggested that cargo could provide a small amount of gold over time, making it useful as a source of income before the end of the game. While the other minigames are still the primary means of earning gold, the team agreed that this change would be beneficial to overall user experience.

6.7. Summary

User testing helped the team determine that Railroad Runner’s minigames would appeal to a Japanese audience. The method of testing each minigame was different, but several
reoccurring elements appeared in the feedback from multiple games. The most common of these elements was motivation; playtesters nearly always reminded the team to think about why users would want to play each game, and more importantly, why they would want to continue using the application over extended periods. Sometimes it was unclear to the team what this motivation would be. As a result of this feedback, player purchases were made more significant in My Train, so earning gold could serve as additional motivation for all the application’s minigames.
Chapter 7: Game Engines

The team had to decide whether to use an existing game engine or build one from scratch. The first step in approaching this problem was creating a comprehensive list of features needed in the final product. The team decided that requirements included networking, sprite loading and manipulation, support for animation, scene switching, accelerometer detection, touch detection, swipe detection, and physics. After detailing the production timeline it was also determined that there was not enough time to design, program, and thoroughly test a new game engine. Instead, the team searched for a preexisting engine that could handle the features listed above and add in any additional features that were needed. The focus should be on game building and other aspects of the project.

The team started by taking a quick look at a long list of engines to see what features were available. The results showed that many of them were out of date, incomplete, or no longer supported. Through investigation, the team found three engines that seemed promising. These were AndEngine, LibGDX, and Monogame. Each of these engines appeared to be well supported, have active forums, and provide example games that were made with them. Despite them being the best candidates available, none of them had support for networking. Fortunately, one of the team members found an outside library called Kryonet that could be used in conjunction with a game engine. The three remaining programmers were asked to take a closer look at one of those three engines: AndEngine, LibGDX, and Monogame.

7.1. Monogame

The team conducted research on the Monogame game engine, an open source implementation of Microsoft's XNA 4 framework. Monogame uses C# to implement games and is multiplatform, supporting iOS, Android, Windows Phone 8, Windows (including the Windows App store), Mac OS X, Linux, PlayStation Mobile, and OUYA.

The team liked several things about Monogame. Several noteworthy games had been made with it. These games prove that Monogame is a complete and functional game engine.

3 http://www.monogame.net/
Monogame has tutorials and an active online forum. It supports Android development. Finally, Monogame is free and open source, meaning that the team could use it without paying for a license.

However, after further research, Monogame was found to be incompatible with the project. Setup required installing many outdated programs, including Visual Basic 2010 (the most recent version is 2012). It was impossible to get some of these to work using Windows 8. The difficult setup seemed like it would waste valuable time, and was a large blow to Monogame's viability.

More importantly, in order to port games made in Monogame to Android, it was necessary to install the Xamarin platform. While Monogame is free and open source, Xamarin requires a business license. The team could not afford this, and so Monogame was abandoned permanently.

7.2. LibGDX

The team also researched LibGDX, which is an open source and cross platform game engine. Its Java API allows developers to develop a game once and deploy it to Windows, Linux, Mac OS X, Android, and iOS. LibGDX is well supported with a full set of tutorials and an online community, and it has useful features including graphics, audio, input handling, math, physics, file I/O, and storage. All these features made LibGDX a very strong candidate for the team. However, its cross platform features were not required for the project, and it could have led the team to spend extra time on features not needed for the project.

7.3. AndEngine

Ultimately, the team decided to use AndEngine, a Java based Android game engine intended for work with two-dimensional graphics. It is also free to use. AndEngine proved to be very easy and quick to get working in eclipse alongside ADT. It supports very easy sprite creation and manipulation without the need of any vector math on the programmer’s side. It was also quick and simple to add physics features such as gravity, collisions, and velocity vector-based movement. Since AndEngine is an engine made only for Android games, many examples and tutorials found were relevant to the team’s needs. It was possible for us to build a demo
quickly that displayed each of the desired features without losing a lot of time getting used to the engine, and as such, we decided that it would be reasonable to continue working with AndEngine.
Chapter 8: System Architecture

8.1. Networking

The networking structure is one of the core components of Railroad Runner's code base. The team needed to find programming tools to help send information between devices.

8.1.1. Kryonet

For networking in Railroad Runner, the team used an open source API called Kryonet, a library that is focused on efficient networking. It is easy to use and intended for development in an Android environment. Since the team members did not have a lot of experience with network programming, Kryonet appeared to be the best option to implement networking support in a timely fashion. Another bonus was that the official website included a video tutorial showing how to make a simple network program.

Before the team chose Kryonet, research was conducted on several other options. AndEngine had multiplayer support that was shown to work well. However, the code for this example was extremely complicated and without documentation, so the team did not know how to use it. Therefore, the team decided that figuring out AndEngine’s native multiplayer support was not a viable option.

8.1.2. Networking Structure

There are three main components for the networking structure. These components are the protocols, the listeners and the packets. A UML diagram of these components can be seen in Figure 20. This diagram shows both the networking components and how they interact with the database and the game scene.
Each protocol acts as either a server or a client. The server and client programs are the backbone of the whole networking structure. They contain a main function that starts a server or client automatically upon instantiation. Each server listens to a specific port and waits for clients to connect. While the server and client can send and receive messages, they are not programmed to interpret these messages. Instead, a listener class associated with a specific client or server interprets the messages.

The listener reacts to three basic events: connecting, disconnecting and receiving. The connecting and disconnecting events only occur when the client or server connect and disconnect with each other. Receiving occurs whenever the server or client receives a packet. After the main components receive an event, they call functions in the listeners according to the type of the event. This is the key component to handling all real-time requests and actions.

The package holds information and is sent between the server and the client. Kryonet has strict rules about the size of the object to be sent, so one of the team members created small packet classes that contain only public variables.

A prototype to demonstrate Kryonet’s networking capabilities was written in three steps. First, communication between the server and client was tested locally to ensure basic functionality. For this step, the team used the IP address localhost. Localhost connects the
computer’s server with its open ports. If the network worked using localhost, it would also work on a real server in a separate location. Therefore, the team decided to test using localhost first.

Second, more clients were added to test whether the server could accept multiple connections. Kryonet has built-in thread management. After testing, it was found that Kryonet’s thread management was functional, but not as good as the process management service provided by UNIX. As a result, the team built multiple server programs that were bound to different ports. This reduced chance of crashes or stack overflows on the server.

Finally, the server was tested on physical devices to make sure everything worked in a real life situation. These devices included the virtual server and an android device. At first, the team believed that since the networking worked in the console, if it was put in the android device it would work without being modified. However, the team ran into some problems. Android requires the user to grant permissions for services that pose a security risk. For instance, there are permissions required to use Wi-Fi, 3G, the camera, and the network. The team discovered that without permission to use networking, the application would crash.

In addition to the permission issue, the latest version of android also has a strict rule for threading. Once network permissions were added, the application started crashing with a NetworkOnMainThreadException error. The system threw this exception when attempting to run network code on the main thread - the android activity itself. For example, if a connect function is added directly in the activity class, an exception will be thrown. The solution for this issue is the Android class called AsyncTask.

An AsyncTask works similarly to a thread, in that it separates processing from the main thread. Any code can be put into this class and then run using execute(). Running network code in an AsyncTask removes the NetworkOnMainThreadException. However, using an AsyncTask is slightly complicated. First, if multiple network operations are needed, different AsyncTask must be created for those non-repeating actions. These individual AsyncTasks make the code less readable because they create more classes. Second, since an AsyncTask works like a separate thread, it can cause concurrency errors. For example, if an AsyncTask is used to initialize a variable, and then the main function contains more code after the AsyncTask is executed, it can cause a problem. If this code needs to use the variable, the AsyncTask may not
finish initializing the variable before the main function attempts to access it, and the code below will return a null pointer exception.

During the project, the team used a virtual server set up by Professor Kiyoshi Kiyokawa at Osaka University. This server has many common components such as MySQL server, SSH, file management, security and Webmin, an administrative tool. There were some networking problems caused by using a real system. The program was listening to the port, but the client could not connect to it. At first the team assumed the API had errors, but finally it was discovered that the server’s firewall was blocking incoming traffic. Thanks to Webmin’s easy-to-understand GUI, the team successfully changed the firewall settings and fixed the error. The list of helpful tools Webmin provides made it easy to modify the virtual machine’s settings. Because the computer running this virtual server is located in Osaka University, network lag was not a problem. Otherwise, lag would have caused severe problems with the application.

There was still one more problem with virtual server. First, the IP address was not static - this means it would change upon server shutdown. The team had to find the new IP address by hand, because it would change randomly. When the IP address changed, the team had to change all the client connection functions, because it required the IP address hard-coded as a parameter. Because the team never knew when the IP address would change, it had no control over when some fatal error would occur.

8.1.3. Kryonet Defects

Although Kryonet has had many revisions, it still has many major defects. For example, in the latest version (2.18), the system will automatically disconnect the client or the server soon after they finished connecting with each other. For this reason, the team used version 2.12 instead of the latest build. Kryonet also cannot send strings on android devices running Android 3.x. When trying to send a String object from 3.x devices, the client always throws an exception for accessing a non-existing index. However, when the team tested sending strings on 2.x and 4.x devices, this error did not come up. There were reports of this problem on Google Group, and the solution was modifying the API. The team decided it would be easier to send arrays of chars instead of strings.
In conclusion, although the team had difficulties with Kryonet, it was still helpful for networking. It saved the team a lot of time. Rather than developing a new network structure, which would be error-prone as the team has little experience coding networks, the team could use this library instead. Since the server and the client share a similar pattern and structure, Kryonet is easy to understand and use.

8.2. Database

The application often needs to store data for future use and for sharing between users, and the user needs to receive their personal data upon login. It is also impractical for each user to store the data for all other users on their personal device. To solve these problems, the team created a database, made using MySQL, the most popular and best-supported database management system. The data was stored on the team’s virtual machine at the Osaka University campus.

8.2.1. Database Structure

The database is used for storing permanent data. This data includes user information, game scores, and records of the number of times each user has played with other users. The corresponding tables are Users, Scores, and Played_World. There is also a table of GIDs (GameIDs), which pairs each game name (String) with an ID (Integer) and is mostly for extensibility and readability when performing manual lookups. The exact database structure can be found in Appendix B.

The users table has one row for each individual user. The primary key is a UID, which is a 4-digit integer. When a row is added to the Users table, a UID is assigned, incremented automatically, and then sent to the client. The Users table also stores a username, password, user score, and current train score for each user.

The scores table has one row per game played. The UID, GID, and the score itself are also stored. Lookups are primarily done by UID, but since there are several scores for each user and each game, the trivial field SID was created for use as the primary key.

The Played With table has one row for each pair of users that has played at least one game together. This primary key is called the UID_HASH. The UID_HASH was designed for
extremely fast lookup for any user pair combination. It is made by ordering the two UIDs from least to greatest and then concatenating them. Since each UID is exactly four characters, there are no duplicates.

8.2.2. Database Networking

The Android client does not send queries directly to the database. Instead, a server and client system was made using Kryonet. There are three main components for networking in the client and the server. These are the client and server classes, the protocol, and the listener. The client and server handle sending messages, the listener handles receiving messages, and once messages are received, the listener passes them to the protocol class to be interpreted.

DBProtocolC is a class that is never instantiated. It contains a static DBClient class that is created as the DBProtocolC class is loaded. Therefore, any time the application is running, it is connected to the database server. Upon instantiation, the DBClient creates a DBListenerC. This structure is mirrored in the server with DBProtocolS, DBServer, and DBListenerS.

When any code in the application requires data from the database, it calls static functions within DBProtocolC. All parameters are strings, which are then concatenated with a function identifier and converted into an integer for efficiency. The only functions that do not send integers involve usernames and passwords. This integer is then sent to the DBServer, which passes it to the DBListenerS, which converts all incoming packets to strings and then hands them to DBProtocolS. DBProtocolS then interprets these strings and sends queries to the database.

DBProtocolS sends database queries using static functions in a class called MySQLAccess. MySQLAccess is another class that is never instantiated. It uses the JDBC (Java Database Connectivity) interface to send and retrieve queries to the MySQL database. First, a connection function must be called to establish a connection between the program and the database. Next, queries, insertions, and deletions are run using a PreparedStatement, which prevents security risks such as SQL injection. Finally, the connection is closed.

8.2.3. Storage in Android

In addition to the data stored on the database, it was necessary to have persistent storage in Android. This storage includes game data for My Train and the user’s personal information
such as their username and UID. Normally any variables in an Android application are deleted when the process is killed. In order to get around this, the team used an Android feature called SharedPreferences. SharedPreferences takes a filename string and stores variables as a key-value pair in a persistent XML file. These values can be loaded or stored at any point during application runtime. For security reasons, SharedPreferences are cleared when the user selects the Log Out option within the application.

8.3. Joining Games

Joining games appears simple to the user, but it was actually quite complicated to program. When the user joins a Train Room and enters the Main Menu of the application, they are connected to the GameJoinServer. The application sends the server the user’s UID and the Train Room he wishes to join. The server maps the UID to the user’s data. This data includes a connection value, Train Room, and later the game the user is joining. Each Train Room is also mapped to a list of users, and each GameID is mapped to its game data. The application then requests a list of users in the room and a list of games. Each game’s data is turned into a list of integers and all the games are then sent as a list of lists.

In the application, the user can take several different actions. If the user has not already joined a game, they can start a new game or join another game. All new games are visible to any user in the same Train Room. If the user has joined a game, they can leave that game. If the user is hosting a game and a sufficient number of players are ready, they can start that particular game. The user can also change Train Rooms whenever they are not currently playing a game. Finally, the user can quit the application at any time, disconnecting from the server.

Each time the server updates a game, it sends the updated game to all users in the relevant Train Room. If a user joins a game and the minimum number of players has been met, the server will start a timer for 20 seconds. When this timer runs out, the server will send a message to all players to begin the game immediately. During this period, no user can leave the game and the game’s host can send the server a message to start the game early. Once a game has been started, the server removes it from the list of games and notifies all users of its removal. If a user chooses to leave and is the only player remaining, the server will remove the game. When the user changes Train Rooms, the server updates the user’s information and removes them from any
games they were previously in. When the user disconnects from the server, the server removes the user from any game they have joined and removes the user’s data. If the user commits any action that would cause an error, such as attempting to join a nonexistent game, the server will send back an error code.
Chapter 9: Art

Railroad Runner primarily features a cartoon-inspired, sketch-like art style. Because this application is played on mobile devices, it had to implement an art style that would be able to retain its detail on small or low-resolution displays. For this reason, the majority of the art in each game falls into one of two categories.

Important, interactive game objects boast Railroad Runner’s cartoon-like style. This style is demonstrated in Figure 21. They frequently employ bright colors and bold outlines that help them be more visible to the player and indicate that they are important to gameplay. Conversely, the backgrounds, shown in Figure 22, featured in the minigames use a more faded, painterly style. This helps the backgrounds remain interesting but also keeps them from becoming too intrusive and distracting the player from actual gameplay.

![Figure 19: Initial sketch of the Bomb Pass Panic character](image19.png)

![Figure 20: Bomb Pass Panic's background graphic](image20.png)
9.1. Individual Games

Each game within the suite adheres to the application’s overall style. However, since each game is different, each one makes some slight variations to that style. Additionally, some games rely more heavily on program-generated art rather than custom-made art, although original art is featured in every game. A complete list of all of Railroad Runner's original art can be found in Appendix I.

9.1.1. Bomb Pass Panic

Bomb Pass Panic was the first minigame to receive its final art, and as such, it is the first demonstration of this application’s art style. The primary game objects, including the players, bomb and explosion, were all created in the application’s main sketch-like style. Meanwhile, the background is more painterly and subdued, and this has the effect of keeping the player focused on the characters and the bomb. The animation sheet for the explosion can be seen in Figure 23 below.

![Figure 21: Sheet for Bomb Pass Panic's explosion animation](image)

9.1.2. Creature Creator

Due to the nature of its gameplay, original art is especially important to Creature Creator. The entire game environment consists of a large, segmented character that can be manipulated by three players. These character segments follow a more detailed form of the sketch-like style seen in the other minigames, though the style is not so different that it might alienate players. Ultimately, each character segment was designed to be recognizable to both Japanese and American audiences while also avoiding copyrighted likenesses. An example of Creature Creator’s art style can be seen in Figure 24. This image shows all the options for legs for a creature.
9.1.3. Treasure Trap

Due to the highly variable nature of the game, much of Treasure Trap’s art is generated by the application. However, this game still features original graphics for static elements such as the coins, players and goals. One of these goals, a boot, is shown in Figure 25. The boot was chosen to represent the least valuable goal. An old boot is instantly recognizable as having comparatively less value than the other goals, which are treasure chests. The Treasure Trap coin resembles the gold coin used as money across all the games, but it was specifically redesigned so as not to confuse players. Use of this new gold coin implies collecting one would not give players a direct monetary reward; rather, the coin earns points within the game.

Conversely, to create a sense of unity between the games within Railroad Runner, the player icons were designed to resemble those from Bomb Pass Panic. The various player heads, shown in Figure 26, directly share color schemes with four of the Bomb Pass Panic characters, though they were all given different expressions to keep them clearly distinct.
9.1.4. My Train

My Train displays a style of art direction similar to Bomb Pass Panic. The primary game objects are drawn in a bold-outlined, sketch-like style while the backgrounds are more subtle. However, in My Train, certain aspects of gameplay are also incorporated into the graphics. The main game screen shows the player’s train moving horizontally with a scrolling background. Additionally, the background trees and background clouds scroll at different, scaled speeds to create a more three-dimensional environment. This background can be seen in Figure 27.

Additionally, given the cultural significance of trains in both Japan and The United States, the primary assets were designed to be accessible to both cultures. The train itself takes cues from several models of Shinkansen, but it is also not so exotic that American players will find it strange. The Shinkansen can be seen in Figure 28, above our interpretation in Figure 29. The store screen is also directly inspired by the vending machines found aboard 700-series Shinkansen trains, but it features an entirely symbol-based interface to remain familiar to everyone.
Figure 26: Shinkansen Model n700 Train in Japan

Figure 27: My Train’s Shinkansen-inspired vehicle

9.1.5. Elemancer

Unlike the other games in Railroad Runner, the background and primary assets blend together to create a unified interface and environment for the players. In terms of gameplay, the only primary elements are the six points the players tap to cast spells. At the same time, important information is conveyed through both the surrounding game elements and also the background itself. These background elements were designed to be readily visible to new players, but because Elemancer is ultimately a game of memorization and skill, these guides can also easily blend into the background once players have learned all they need to know. Figure 30 is shown in Elemancer’s background. It is used to show players which elements are used to counter other elements, as well as the constellations used to cast these elements. Important areas are highlighted using white outlines, making them easily visible to new players,
Figure 28: Elemancer's spell hierarchy chart
Chapter 10: Post-Mortem

With the application complete, it is important to learn from the project experience. Things to reflect on include what went well, what problems were encountered, what was cut, and what could be done in future iterations.

10.1. Goals Completed

The team successfully achieved all the planned project goals. The main application was developed to allow users to create their account and log in. With the QR code reader users can scan a QR code and join a Train Room. Then they can join a game or host a new game. A chat room was also implemented, allowing users to chat with other users inside the same Train Room. Four multiplayer minigames are included inside the main app: Treasure Trap, Creature Creator, Bomb Pass Panic, and Elemancer, each of which can be played in about 1-3 minutes. Finally, the team designed and implemented the persistent My Train game that ties all the minigames together, serving as motivation to play more minigames.

10.2. Ownership

Though everyone worked as a team to ensure that Railroad Runner fit together smoothly, the team divided work and each person focused on different areas of the application.

10.2.1. Merrielle Ondreicka

I acted as the team’s project manager, which served as my primary role for the duration of the project. As project manager, I created a timeline for software development and writing the paper. For this, I used a Gantt chart made in a program called Gantt Project. The resulting timeline can be found in Appendix F. In addition to creating an overarching timeline for the project, I scheduled group meetings, took minutes at all the meetings (occasionally with assistance from Alex), created to-do lists, and mediated the assignment of work. The meeting minutes can be found in Appendix A.

In order to organize everything, I needed a clear vision for the project and its goals. The team had extensive meetings in order to plan product features, and from these plans, I created
mockups of the application UI (excluding the multiplayer games). These mockups made it so that each team member knew exactly what was needed for each area of the application.

Every day during the development cycle, I checked in with each team member to ask how their work was going, what they needed to be done for the day, and whether they were having trouble with anything. Each Wednesday we had a meeting with Professor Lindeman so that he could check up on our work. One thing I am particularly proud of is the creation of a regular Friday meeting. Every Friday at 2pm, the team would stop work, review progress for the week, and plan work for the next week. After this, we would create a playable prototype for a new minigame and play it with the lab members. There was often food offered as a reward. The meetings served as a way to wrap up the week and gave the project a feeling of regularity and steady progress. They were fun while still remaining productive.

In addition to working as project manager, I also developed the database and all related networking and functionality. I worked extensively on the prototype of My Train, creating both the game instructions and the board. Lastly, I worked on the main menu as well as joining and hosting new games. This included both the GUI and the networking for joining unique instances of each game, as well as separating games by Train Room.

10.2.2. Jennifer Baulier

My focus was on minigame logic and game design. I designed Bomb Pass Panic and Elemancer and also built the Processing based prototype for Elemancer. I built a processing based prototype for Bomb Pass Panic, but we ended switching to a physical prototype instead. Before the team decided to use AndEngine I wrote a small program to test AndEngine’s functionality as proof-of-concept that it would work for our project. I helped my group design and run all game prototypes. I was also responsible for the initial application setup. This included building a scene manager and creating placeholders for any screens planned for the application. I also built a basic menu used for testing the minigames until the final layout was set up.

I fully built the mechanics for the Bomb Pass Panic minigame and added in the networking. I also put in the art and animation. I fully built the mechanics for Elemancer single player and multiplayer, and handled the multiplayer networking. In the My Train game I was responsible for building the vending machine.
10.2.3. Alex Kim

My primary job was creating all of Railroad Runner’s art assets. I began by creating an asset list for each section of the application, and then creating the assets in the order they were being implemented by the rest of the team. Toward the end of development, after most of the art was finished, I started taking orders from the rest of the team whenever they needed me to fix an existing asset or make something new to account for sudden changes or unforeseen issues. While I was not tasked with building any of the games within the application, I made sure to give as much input as possible during the conceptualizing and testing stages.

10.2.4. Haoyang Zhang (Lance)

My job was developing and providing support for the network components, the Treasure Trap minigame, and some GUI pages. At first, I went through a Kryonet tutorial and built an prototype client-server program. Then I tested this prototype using a real server and an android device. During this process, I encountered issues with the firewall, Android threading, and also version-specific problems. Finally, I implemented Treasure Trap using my networking component. After I finished making the game, I made the GUI and programmed the functionality for the Train Room login page, the join room page, the QR Code scanner, the profile page, the change password page, and the credit page.

10.2.5. Runzi Gao (Jack)

I worked on the design, as well as various technical component of the project. Based on my personal train ride experience in Japan, I first came up with the idea of some sort of social game that could be played on trains, buses, and other public places. This was one of the ideas the group really liked in our brainstorming stage, with the other being a workout application. In order to make this idea more feasible and practical, I made a few changes to it before it was chosen as our MQP.

One of the technical components I worked on was the implementation of Creature Creator game, which was first built as a single player game and then converted to a multiplayer game with networking components added in. I built the stats screen of the My Train game, which
shows an animated Shinkansen train as well as the player’s train stats updated automatically. And I also implemented the Chat Room feature in the Railroad Runner application at the last.

10.3. What Went Well

The team did quite a few things well. Several of the programming choices the team made sped up development. The team chose to develop for Android over iOS. Android proved to be relatively easy to program for, and since its apps were created using Java, the team did not have to learn a new programming language. The team also chose to use an existing game engine, AndEngine, instead of creating one from scratch. AndEngine took mere days to learn, whereas developing a game engine would have taken weeks and the team would have spent even more time ironing out bugs. The team also chose to use several other libraries to support the project. Kryonet was used for networking, Zxing was used to read barcodes, and JDBC was used to connect the server to the MySQL database. These libraries allowed the team to speed up development and focus on content rather than lower level mechanics.

The team members also worked well together. Work was divided evenly between everyone. When deadlines were set for work, they were often met. If a one person struggled with an assignment, other team members were willing to pick up any queued work. One aspect that worked surprisingly well was the time scheduling. Everyone worked in the lab at different times. Several team members arrived early in the day and worked a standard schedule. Other team members, however, arrived later but often stayed all night. This setup allowed people to work at times that were most comfortable for them. Everyone continued to make steady progress with their work, so having different schedules never proved to be a problem. All team members still attended scheduled meetings, however.

A final thing that benefitted the team was that the project was incredibly scalable. Since the application consisted of a core with several minigames added on, it was possible to add and remove minigames based on the amount of progress the team had made. Even if some of the minigames had to be cut, the project would still stand as a complete product. This allowed the team to adjust its workload dynamically.
10.4. Problems Encountered

Several things either did not work well or could have been better for the team. First, the team spent too much time planning and designing the project. It was not until Professor Lindeman arrived in Japan that the team really made any decisions. After that, a project manager was assigned and work progressed much more quickly. If the team had been more organized in the early stages of the project, much more could have been accomplished.

Another problem was that the team worked on creating the minigames before working on the application’s core. Initially this seemed harmless, but once the core was finished all the games had to be restructured to work with it. Additionally, all the games were initially made single player, as this was simpler, and then modified to support multiplayer, and then modified again to handle several instances of each game at once. If the core functionality had been made first, these repeated modifications would have been unnecessary. Even though the multiplayer games were completed first, some of the games were not coded with multiplayer and extensibility in mind. Some of them were very difficult to modify from their original states.

Next, the networking aspect of the application could have been better organized. Each game has its own server, client, and listeners. There are several packages and classes containing redundant code, especially the networking packets. A more organized way to handle this would be to create interfaces or abstract classes for the clients, servers, and listeners and to organize all packets into one package.

10.5. What Was Cut

The first and most noticeable cut to Railroad Runner's content was a good deal of the minigames. Before the development phase began, each team member envisioned at least two minigames, but only four of them remain in the final application. Ultimately, it was decided that no single minigame was essential to the overall application, and most of the conceptualized minigames were cut in the interest of time to allow the team to focus on improving the application's overall user experience.

Sound was another important omitted feature, but its removal was a simple decision that was made early on. Making noise onboard trains is considered rude and disruptive, especially in
Japan. Because of this, the team quickly realized that sound would either have to be available only to users with headsets or left out entirely. After some short deliberation, it was decided that no sound would be the best option. Including sound could have potentially improved the user experience for players with headsets, but it was also deemed to give an unfair advantage to players without them.

While chat was implemented in the final version of Railroad Runner, it was originally planned to be much more expansive. The team considered the possibility of a simplified emoticon chat that would remove the issues of language barriers and profanity. Additionally, this emoticon chat was also intended to be available within the minigames, allowing players to communicate directly with each other. In-game chat was cut due to time constraints, and it was decided that verbal communication would allow for a better user experience despite the risk of profanity.

My Train exists as a single player game, but it was initially proposed as another multiplayer game, though still larger in scale than the minigames. At first, all players in one Train Room would interact with one shared virtual train rather than each having their own, and these virtual trains would directly compete with those representing other routes or areas. In this version of My Train, the game was much more focused on micromanagement and long-term strategy. While this would have been fun for some players, it was eventually determined to be too complicated for casual players. Additionally, the competitive nature of the multiplayer version of My Train gave a clear advantage to anyone taking longer train rides, since someone would always be present to maintain the train as opposed to shorter or less frequently used routes. This led to My Train being modified into its final, single player incarnation, but there was one other planned feature to be cut. Single player My Train was originally going to feature a cargo trading mechanic which would allow players to help each other’s’ virtual trains. While no one on the team was opposed to this feature, it was eventually cut due to time constraints and to make way for the chat room, which was agreed to be a more useful social feature than trading.

Several peripheral features were also planned to help users with their train rides. Originally, train routes would be monitored by GPS, and users would be notified when their destination was approaching. However, enforcing certain locations to allow use of the
application proved to be too complicated, and many train routes intermittently drop out of GPS coverage. Without GPS, there was also no accurate way to inform users of their destinations.

10.6. Future Iterations

If future groups were going to continue this project there are quite features that could be added onto the existing application. One possibility is the addition of more social features related to the My Train persistent game. At its core, My Train is an individual game such that the players can work towards a goal over time. However, features such as an item trading system would give players the chance to help each other in-game. A player could offer another player a set of items and request others in return. The offered player would then get a notification saying they have an offer and can either accept it, reject it, or modify the terms of the trade. If accepted, they will immediately receive their half of the items and give away the items they are sending. The player who offered the trade could get a notice and the items would then be swapped on their side too.

Another possibility would be to implement a gifting mechanic instead of implementing a full trading system. In this system, some number of times per day or week, a player can donate cargo to another player. When that player goes to the train game menu, they will be notified and receive the gifted cargo. Cargo received in this way could count as “foreign cargo” and would be worth more when tallying the score. To encourage players to do this, the player sending the gift could get one foreign cargo for every cargo they send. In addition, players should be told who gave them to the gift to encourage further interaction between those players.

In addition, to encourage further interaction it may be a good idea to put in a mail system such that players can send messages to specific players that can be received the next time they log into the application. These could only show up when the player first logs in, or a full inbox mail system could be implemented. This would encourage communication between people who have played together a lot, or helped each other through trading. Similarly, a way to invite specific players to play minigames may be useful once players start getting to know each other. At the very least, a way for a player to notify a given user that he/she is starting a game may be a good addition to the application. On top of this, if a friend list system is implemented the option to start a game that only friends can join may be a good feature.
Another major feature that could be added is a way to ensure that players are actually on the same route as each other using GPS. Currently the application allows players to play on any route they have scanned the QR code for. This would be a useful feature because it would ensure that people would be playing with users in the same area.

On top of this it would be a great addition, albeit a challenging one, to build a system to automatically put players into a room for the route they are currently traveling on without them ever having to scan a QR code or input the route themselves. This ensures that they are playing with people who they are traveling with and removes the need to put QR codes on the trains. If such a system exists, it should be able to move them into a new room automatically when they switch routes, and inform them that their Train Room has been changed. The team feels that this alone would be a very complex and interesting technical challenge for a future group to undertake.

If either GPS or a system to determine train routes were implemented, a feature could be added so that the player could get information about their trip. A screen showing what stops are on their route and at what time they will arrive would be a very useful feature. Furthermore, if a player was able to input their destination, the application could warn them when it is approaching and even advise them against starting minigames that the player would not have time to complete.

People often enjoy things that are customized for them, so certain customization features could be interesting additions to Railroad Runner. Players could unlock different styles of train as they complete goals in My Train. For example, if the user’s high score is above a certain value they can unlock a new train to use. Their user information page could list what train they have set so other users can see. It may be interesting to come up with other items that can be unlocked through various achievements. To give the players a greater sense of achievement, a leaderboard could be implemented for the room showing the top player for each minigame as well as the top My Train score and the top user score.

Another interesting way of giving players custom items could be based on the real-world locations they have visited. If the player is on a train that stops at a certain well-known spot, he/she could receive a train decorated in a way related to that location or something some other specialized item. Another variation of this is to have certain minigames that could only be played
while on certain train lines. This could be done to promote a given train line or to create variety. This will encourage people to continue playing because they would want to see the new minigame for their current route.

The addition of more minigames is a good way to extend the game. There are infinite possibilities for minigames that could be created that would fit in and enhance the existing game library. The larger Railroad Runner’s minigame library is, the more it would encourage users to come back and keep playing. It may also be a good idea to have more games that could work with single player so if someone were the only player on the train they would not be bored.

If players want to play something as quickly as possible but do not have a preference as to which game, having a quick play option would be a good addition. Once there are any four people looking for a game, using quickplay could pull them into a four-player game at random. Another possibility for quick play is that as soon as any game is about to start, but can hold one more player then that player will be added into the game session.
Chapter 11: Conclusion

The Railroad Runner team successfully created an Android application that acts as a client for various live multiplayer minigames. These minigames then contribute gold to a persistent, single player game to act as further motivation to continue using the application. Near the end of development, the team presented this application to professors and graduate students in Professor Haruo Takemura’s lab at Osaka University in Japan. The presentation was very well received, especially the demo during which the members of the audience were able to play the minigames with each other. Afterward, the team received very positive personal feedback from the Osaka University professors.

The decision to keep the minigames very short made them playable and easy to demonstrate. As a result, the presentation audience and volunteer playtesters were all able to directly interact with the Railroad Runner application rather than simply listening to a description of it. Despite being anonymous, the playtesters were all very engaged and competitive with each other. This reaction validated the project’s primary concept: players had fun and socialized while still remaining anonymous.
Appendix A: Meeting Minutes

2013_07_12 Meeting Minutes - Job Query and General Android Tutorials

Due Tuesday:
- go through Android tutorials (something that you DO step by step)
  - build a basic android application
- Get GanttProject up and running
  - get synchronized document
  - put in all the dates that you know (leaving japan, A term start, paper due)
    - testing start date
- Make and share calendars
- Finish ranking games, compute average rank

Next Meeting w/ Professor Lindeman: Wednesday at 2
Bring chocolate bars

Decisions:
- timeline
- platform - Java?
  - Find other dev environments for Android
- minigames
- server location
- Version Control
  - Set up FusionForge
  - SVN vs GIT

Tasks:
- Game prototypes: build in a day at most
- Test with people in our lab!
- go through tutorials (something that you DO step by step)
- Decide whether scrum is useful

Jobs:
- Project Manager
  - Jack, Merrielle
  - Gantt Project
  - Help people figure out what they need to do
- Android - Everyone
- Get android dev environment
- go through tutorials (something that you DO step by step)
- Look up rules for developing for Android (back button, etc)
- Research Android App Life Cycle

- Networking
  - ?Merrielle, Lance
  - Query structure - strings, objects?
  - What data and how much and how often to send
  - Encrypt passwords
  - Protocol - structure of messages
    - Protocol Layer
  - How does the client find the server?

- Server Location
- Databases
- Art - Alex
- Canvas Drawing
  - Drawing on the canvas and moving sprites around
- Logic Control
  - ?Jen, Merrielle
- Architecture
  - ?Jen, Lance
  - UML
- UI Design
- Game design
  - ?Jack, Jen
  - not Merrielle :P
  - What’s the goal of each game
  - How do I make it so it has appropriate and increasing difficulty levels?
  - Types of objects
  - Basically telling the artist what we need and how they interact
    - in detail!

2013_07_16 Meeting Minutes - quick tutorial checkup

- Tutorials
  - Jen – started screen support
  - Alex – started activity life cycle
  - Lance – set up, not started
  - Jack – none
  - Merrielle – none
• Minigames - choose
• Gantt Project - complete
• Scrum / Meeting plan - research - Merrielle
• SVN vs GIT - research - Jack
• Calendars

Notes: Green = Complete, Red = In Progress, Yellow = Waiting

Ideas:
• Send emoticons within games?
• Ticker for next station - based on GPS?
  ○ issue with Subway
  ○ Possibly get schedule from online

2013_07_23 Meeting Minutes - Present Apps and Start Picking Jobs

Agenda:
• Present apps from tutorials

Suggestions from Merrielle:
• More planning
  ○ Put more details on Gantt Project
  ○ Schedule deadlines for game prototypes
  ○ Scheduled small team meetings?

  Note: When and how much time you spend in the lab doesn’t really matter as long as you’re getting your work done. However, coming into lab at weird hours or not at all makes it harder for us to work together, keep everyone on track, and ask each other for help. I would like to suggest that everyone be here at LEAST from 10:00 - 4:00 every day unless sick or attending a class. That only leaves 5 hours of work with a 1 hour lunch break. Get here earlier or leave later, whatever works for you.

  I’m thinking we should schedule a daily 15 minute meeting at 10:00 in the meeting room to catch up on what everyone’s done and what they’re doing for the day and whether they’re stuck on something. Super quick, 3 min each.

• SVN vs. Git
  ○ fusion forge?
  ○ Get that set up!
• Break down project into smaller tasks/jobs (networking, etc)
  ○ Maybe come up with better ideas of exactly how each task will work
    ■ use a flowchart, userstory, whatever works
I’m mostly thinking about the main interface here / connecting to games

From last time:

Jobs:
- Project Manager - Merrielle
  - Gantt Project
  - Help people figure out what they need to do
  - What is the experience of the player at this point
  - Make sure all separate parts fit together properly
- Android - Everyone
  - Get android dev environment
  - go through tutorials (something that you DO step by step)
  - Look up rules for developing for Android (back button, etc)
  - Research Android App Life Cycle
- Networking - Lance
  - Start quick!!!!!!!!!!
  - Query structure - strings, objects?
  - What data and how much and how often to send
  - Encrypt passwords
  - Protocol - structure of messages
    - Protocol Layer
  - How does the client find the server?
- Databases
  - ?Merrielle, Jack
  - Figure out when to tell the user to get off the train
- Art - Alex
  - Mummy will be difficult - smooth animation
- Canvas Drawing
  - ?Jen, Lance
  - All games in OpenGL to start
  - Drawing on the canvas and moving sprites around
- Artificial Intelligence - Jack
  - Different for each game
  - Needs more discussion
  - Grizzly bear will be hard
- Logic Control
  - ?Jen, Merrielle
- Architecture
  - ?Jen, Lance
  - UML
• UI Design
  ○ ?Merrielle, Jack

• Game design
  ○ ?Jack, Jen
  ○ What’s the goal of each game
  ○ How do I make it so it has appropriate and increasing difficulty levels?
  ○ Types of objects
  ○ Basically telling the artist what we need and how they interact
    ■ in detail!

• Documentation
  ○ design docs
  ○ planning for the paper

• Testing

  Put bitmap in open gl
  recognize touches and which polygon it is touching

  prototypes:
  reproduce users, environment

  add in head/body/feet game

2013_07_31 Meeting Minutes - General Planning

• How everyone’s doing
• Requirements (Maybe later)
  ○ Divide everything into smaller pieces
  ○ make user stories
  ○ test as you go
• Review Goals
  ○ Check that we’re following them
• Make the project more cohesive
• Tweak and throw out ideas
• Dependencies (How to handle them)
  ○ Critical Path
• Schedule game prototypes and testing
  ○ test subjects gone for summer?
• Architecture (Jen)
  ○ Game engine components

Server
• Virtual machine at WPI
  ○ just SSH in
  ○ Latency?
• Commands vs. Events
  ○ command: touch here
  ○ events: move here
• Sceneref
• Framerate - how fast I draw stuff
• Update rate
• ex: send velocity instead of position
  ○ dead reckoning: keep moving unless you hear otherwise
• Inconsistency b/w what everyone believes
• Increase tolerance for latency
• Games Database
• Users Database
• Exactly how much data we need to send
  ○ and how often
• Jen - Make a communication dataflow diagram for each game

Audio
• media players for long sounds
• separate things for effects

Alex - test vertical vs. horizontal layout for frame
  turn asset list into excel spreadsheet - one line per file!

Decisions:
who do you play with?
  many users - same train
  some users - same destination
  few users - anyone on a train

how do we tell which train someone’s on? Assume the train has bought into it.
  gps - problem with subway
  qr code - on each train
  time of day - taken from phone
  user input - state which train you’re on
  user history - remember which trains the user takes

emoticon chat - in game frame

2013_08_02 Meeting Minutes - Amida Testing
Today:
1. Amida planning / which prototypes
2. Amida testing
3. Game engine - specifications
   what do we need?
4. Data Flow Diagrams

Monday:
1. how we're going to prototype
2. Networking
3. Dependencies

Homework for Tuesday:
1. break down each project into small chunks
   a. decide how long each chunk will take
   b. state whether your chunk is dependent on another job
Merrielle: put everything together

Tuesday:
1. Talk about how long everything will take, agree on things
2. More game design / prototype decisions

(Week after) Wednesday:
Meeting with Professor Lindeman at 10am

Meeting times:
monday - sunday night for Professor Lindeman
tuesday doesn't work - Professor Lindeman is teaching
wednesday - trying it
thursday - ???
friday isn't great - japanese class

2013_08_05 Meeting Minutes - Amida Results and Game Engine

Schedule:
• amida
  o test results
  o what do we do about them?
• talk about game engine
  o Firefly example
  o what needs to get done? - asset managers etc
how long will it take?

Amida results:
- not predictable
- no motivation to play many times
  - the persistent game will help with that
- people want to get better at it
- cosmetic upgrades - like zynga
- higher motivation - like WoW

Amida Changes:
- keep as is mechanically
- draw 5 lines in 10 seconds

Game Engine discussion:
http://mobilegameengines.com/android/2d_game_engines
choose C++, Java, C#, Lua (maybe)

- GraphicsManager
- InputManager
- ResourceManager
- WorldManager
- networking

Game process -
- Startup
- central loop - world manager
  - update()
  - draw event
  - step event
  - list of registered events
  - objects take in ID for an event and respond accordingly

Homework due Tuesday:
- look at details of 2 game engines
  - networking
  - graphics management
    - bitmaps
  - Collision handling
  - touch, accel., swipe
  - Tutorial: http://www.edu4java.com/androidgame.html
Jack: Mage, MonoGame
Jen: Candroid
Merrielle: Unity3D (can also do 2D), libgdx
Lance: AndEngine

- break down each project into small chunks
  - decide how long each chunk will take
  - state whether your chunk is dependent on another job

Jen - try to draw game engine diagram

### 2013_08_06 Meeting Minutes - Looking at Game engines

Previous Homework:
https://docs.google.com/document/d/1gOQOURyYTnEgCZeMA6CT7_HdxFovn0VCLgIsP67PHK0/edit

Agenda:
- **Alex** - art
- **Discuss findings on game engines**
  - pick one, or none
- **Jen** - show game engine diagram
- **Discuss breakdown of different jobs**
  - Talk about how long everything will take
  - Merrielle - update Gantt Project after
- **Dependencies**
  - Which parts of the project need others?
- **Extra**: Start designing more prototypes
- **Lance**: work on networking demo

Notes:
everything: kryonet as networking support

Andengine:
support forums
tutorials
no official javadocs
games have been made with it
graphics, physics, touch support, sound support
easy to get started

LibGDX:
faster than andengine
more like a framework than an engine
support forums
wiki
fully javadoc’d
tutorials
games have been made with it
supports graphics, physics, touch, accel, sound, all inputs

Monogame:
C#
cross platform

Homework:
• Due Fri 23 - create a simple game with one of the three game engines
  ○ Jen - AndEngine
  ○ Jack - LibGDX
  ○ Merrielle - Monogame
  ○ Lance - Kryonet demo, set up physical server
    ■ connecting to the server
    ■ server is sending hello world
    ■ talk to Kiyokawa sensei and set up VM here
• Monday: if we decide on an engine, spend one day learning it

2013_08_21 Meeting Minutes - chose AndEngine

Meeting with Prof. Lindeman at 10am in the lab

Notes from today:
• Choosing AndEngine - have Jen share what she learned
• Get SVN up and running
• Start working on DB
• Get physical server
  — in progress - waiting on Kiyokawa
• Merrielle should make an agenda for every meeting
• Merrielle should always make sure the Gantt Project reflects reality
• Make a folder on Drive for art, and subfolders for each game
• Discuss the order we’ll be working on the games
  ○ Creature Creator
  ○ Hot Potato
  ○ Amida
  ○ Train
  ○ Wizard
○ Grizzly Bear
○ Mummy

To Do:
● Set up SVN in Eclipse
● Learn Andengine - Due friday
  ○ sprite manips
  ○ switching screen
  ○ animation
  ○ physics
  ○ tapping
  ○ accel
  ○ swiping
  ○ sound
● Wait for Kiyokawa so we can set up server
● Write paragraph on individual game engine
  ○ which engine
  ○ describe
  ○ what we liked about it
  ○ if it didn’t work, why?

2013_08_23 Meeting Minutes - AndEngine and HP demo

Agenda:
● AndEngine Demos
● Check if paragraphs are done
● Design Hot Potato prototype
● Test Hot Potato prototype
● Review what we got out of testing
● Quick rundown of next week
  ○ Ask Kiyokawa about weird server error?

Notes:
● paragraph now due Monday
● Finish AndEngine for Monday
● HP: leave button - press if getting off the train,
  ○ cannot play another game for 2 mins

AndEngine:
● Jen: everything complete, incl: accelerometer, touch, swiping, gravity
● Merrielle: physics / gravity on sprites
• Jack: went through jen’s notes
• Lance: looked at tutorials online, they were confusing

Hot Potato:
• get everyone in a circle
  ○ how many: 6ish people
• start timer (not visible to people)
  ○ how long: 30 seconds
• potato changes hands
  ○ via rolling
  ○ via tossing (drop mechanic in game)
• hold timeout: 3 seconds
• as time runs out notify players
• when: 4 seconds left
• game ends
  ○ one loser: high score = survival streak
  ○ one winner: loser leaves, everyone else continues playing
  ○ loser messes with game: not possible in demo

Rules:
1. Pass the volleyball back and forth
2. If you’re holding the ball when the timer runs out,
   you lose!

Run:
• 1 game rolling - one loser
• 1 game tossing - one loser
• pick between them
• 1 game, one winner

Results:
• Tossing is better
• Players didn’t like monotony
• Shorten the timer as people lose, or make it completely random 5-30 seconds
• people generally liked having one loser instead of one winner, because there was no wait time
  ○ however, this doesn’t deal with an issue:
  ○ if the first player never passes the ball and loses after 3 seconds, it’s boring
  ○ it also doesn’t include some of the loser mechanics we’re thinking of adding
    ■ tapping the ball to change its form
    ■ allowing the player to either leave or watch the game after losing
2013_08_26 Meeting Minutes

Agenda:

- paragraph: everyone done
- split into teams of 2
  - hot potato: Jen and Lance
  - creature creator: Jack and Merrielle
- Jen's AndEngine lesson

2013_08_29 Meeting Minutes

Agenda:

- Demonstrate work done
- Future schedule
- More game planning (Jen knows what I’m talking about)

Minutes:

- lance:
  - demo with mobile app connecting to server
  - amida
- merrielle:
  - connect db to server
  - creature creator
  - schedule more paper writing
  - more google calendar
- alex:
  - take up paper
  - getting out art as fast as possible
- jen:
  - keep programming hot potato
- jack:
  - keep working on creature creator

2013_08_30 Meeting Minutes - initial Alphas

Agenda:

- Progress
- Hot Potato questions
- Train Prototype
  - Read Alex’s docs!
Notes:
- hot potato is going really well
- creature creator, left/right buttons, switching images
- art - catching up over the weekend
- amida - gui done
- merrielle - figure out slopes for HP
- lance - teach everyone kryonet Monday

hot potato:
- angle issues
- get the most basic game done first
  - no tapping at first
  - yes swiping
  - one loser at first
- empty spaces if < 6 players

TRAIN:
- user cumulative score
  - from train winning game
  - from minigames
- train winning:
  - crew members - not just extra points?
- Prices for
  - health
  - cargo
  - speed
- reward money for
  - gameplay: each game
  - based on chance of winning game
  - and effort to play
- How are we simulating games?
- Event cards
- challenge other players
  - if you have enough money to wager
  - win double points
  - or lose points (half of what you’d normally win)
- Wager price
- Each player plays one minigame
- 24 turns
- how much movement per power / health
  - health - only affects @ 0
- power - ???

- Max power / health
  - health - start at 50, max at 100, reduced by speed each score
  - 20, for each extra point move one extra square per turn

- Each side of the board is 10 spaces
  - keep a counter for laps

Distance - 50%
Cargo - 50%

<table>
<thead>
<tr>
<th>Resource</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>cargo - no max</td>
<td>$10 / cargo</td>
</tr>
<tr>
<td>health - 0 - 100</td>
<td>$5 / health point</td>
</tr>
<tr>
<td>speed - 1 - 20</td>
<td>$25 / speed point</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Game</th>
<th>Reward</th>
<th>Challenge Wager</th>
<th>Chance of Winning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amida</td>
<td>big - 40 med - 20 lose - 0</td>
<td>10</td>
<td>1/4 big win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/2 med win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/4 lose</td>
</tr>
<tr>
<td>Creature</td>
<td>10</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>Grizzly</td>
<td>big - 60 med - 20 lose - 0</td>
<td>15</td>
<td>1/4 big win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/4 med win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/2 lose</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>big - 50 med - 10 lose - 0</td>
<td>10</td>
<td>1/6 big win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4/6 med win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/6 lose</td>
</tr>
<tr>
<td>Mummy TP</td>
<td>big - 30 med - 15 lose - 0</td>
<td>15</td>
<td>1/3 big win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/3 med win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/3 lose</td>
</tr>
<tr>
<td>Wizard</td>
<td>win - 40 lose - 0 (5 in app)</td>
<td>15</td>
<td>1/2 win</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1/2 lose</td>
</tr>
</tbody>
</table>

Materials:
- cardboard
- colored construction paper
- glue
- index cards
  - requirement to be a first world country
- markers / sharpie
- pens
- paper for
  - instructions
  - money - just keep a notepad
- player trains
  - bottlecaps
  - pokemon figures
- cargo figures
- dice - random
- markers for health / speed for each player
- counters for laps - keep a notepad

2013_09_03 Meeting Minutes - Train Notes

Agenda:
- Networking
- Train Instructions / Design
- Random Events

Notes:
- Networking:
- Lance - make Andengine server demo by Friday

Issues with Train Board Game:
- works better as a standalone
  - Jack doesn’t think it will motivate players to play minigames
- location and time based
- board game designed to ignore location otherwise it's not balanced
- which train wins depends on who has the most players
  - winning isn’t everything
    - people will still enjoy the minigames
  - ex: guildwars 2 world PVP
  - 3DS: you have an individual goal, not competition
    - make the persistent game more individual?
    - not resetting every day / individual prizes
- Balance the trains - keep the numbers even
combine with location
- balance the train based on location
  - ex: 1 train for Toyonaka, ~10 for Umeda, etc

Make it single player?
- Minigames are the multiplayer interaction
- Players can strategize by themselves
  - Don’t have to coordinate with players you can’t communicate with
- Removes negative reinforcement
  - Other players can still do better than you, but you don’t “lose” to them
- Reset after 1 week?
- Almost all client-side. Only sync score with server.
- Playing repeatedly with someone gets you a bonus multiplier to your score
- Jen’s idea: trade cargo for cargo
  - foreign cargo is worth more
  - can trade once per person
- Jack - let’s make instructions for trading cargo
- Events -
  - first event after the first minigame
  - 3/8 chance each turn
  - or random period of time to activate event
  - have a dismiss button before players see the event

Playtesting:
- We should playtest for 1 hour
- first play ourselves
- chop the prototype

2013_09_04 Meeting Minutes

Agenda:
- Individual work
- Train game prototype
- new Train game modifications

Notes:
- Merrielle - plan multiplayer stage for each game
- Jen - plan for this week
- look up scrolling images vs. animated sprites
- fix offline google drive
- by next week (Wednesday) -
- CC - have artwork and multiplayer done
  - swipe to switch image, create winning screen w/ db placeholder
- Jen and Lance - discuss Amida afterward
- Lance -
  - Amida - have the squares follow the lines down
  - Networking comes first
- If you have nothing to do - start writing!!
- Show other players the stats on your current game
  - all time high score
  - cumulative score
- Merrielle - change instructions to be more matter-of-fact
- bigger reward for testing our board game
- settings screen
  - but it’s a lower priority
- Usual meetings last an hour
- Later - google earthquake

2013_09_06 Meeting Minutes - Wizard Prototype and networking demo

Agenda:
- review work
  - networking demo
- wizard
  - changes?
  - testing
- train design
- user screen
- paper writing??
  - have a paper writing day: MONDAY
  - 4 pages each per week for the paper
    - if 100 pages
  - Merrielle’s doing the art section of the paper
    - duh

Notes:
- Jen -
  - HP art added
  - looking good!
- Alex -
  - started train art
    - concept art
o HP
  ■ all done and on drive :D
  ■ updated bomb flash
o Merrielle - look into setting up Jira for bugs
o Jack -

Train:
o rename health to: HP / Condition / Tread
o cargo gives $1 per turn

2013_09_09 Meeting Minutes - Paper and Train Work

Agenda
o Paper
  o what needs to get written
    ■ who does what
  o pacing
  o schedule
o Train work
  o stats screen
  o store
  o events
  o view other users
o Server program on SVN?

Notes
o First draft by the time we leave Japan
o decision process
o prototyping
o application features - describe the program
o goals
o architecture / backend
  o networking
  o other individual jobs (each write about your own job and research)
 o 2-4 pages per week -> 50 - 100 pages by the time we get back

o Train work - done by next wednesday
  o stats screen
    ■ jack
  o store - vending machine
2013_09_11 Meeting Minutes

Agenda:
• review work done
• paper questions

Afterwards:
• Assign paper sections
• Talk about networking

Notes
• Alex
  ○ Rearrange creature creator images
  ○ Maximum allowed dimensions: 2048 px
  ○ Make intermediate frames
  ○ Email Britt again
  ○ Art section of report
    ■ Individual sections for each game
  ○ Amida Art
    ■ Background
    ■ Players
    ■ End points
    ■ Coins
• Merrielle
  ○ put protocol for android -> server for db in gantt
  ○ get db done by friday
  ○ working on CC multiplayer by friday
• Lance
  ○ keep working on multiplayer for Friday
• Jack
  ○ fix ratio issue for images
  ○ working on adding images / going to start multiplayer
  ○ bugs with orientation?
• Jen
- keep working on vending machine
- possibly make players stay in the same spot when one person loses
  - not a huge priority
- talk with Lance about networking
- try running games on different devices
- Paper
  - process is not as important
    - maybe in an appendix
    - schedule in appendix
  - concept art in report, not appendix
  - final design vs. original design
    - original is appendix
  - have a narrative of how our game works
- separate program for handling all the users and putting them into rooms
- server - each game has its own program
  - extension of OO

2013_09_13 Meeting Minutes – multiplayer

Agenda:
- demo work done
- grizzly prototype - how?
- grizzly testing

Notes:
- Alex - scroll clouds and trees separately?
- Merrielle - done all protocol
  - last step is sending and receiving strings to/from the server
- Jen - keep current remove player method
  - current issues: crashes when trying to create a client - Async task issue
  - later: fix bomb throw issue
- Jack - solved some networking issues with Firewall and MainClass error
  - current issues: fatal exception with Async task
    - same as Jen
    - we’ll be talking to Lance
  - next: create protocol
- Lance
  - network structure for Amida
  - finished putting in multiplayer code
    - hasn’t been tested because of hardware issue
Homework:
- paper
- Get more android devices for multiplayer testing
- figure out async task
  - the only things that need to be the async task is
  - send(int port, String message)
    - this is handled in the networking package
  - receive(String message)
    - this is harder
    - different for each game?
- Grizzly bear -
  - think about prototypes over the weekend
  - no prototype? then create a survey to distribute monday via email

NOT A PRIORITY
Tiers for:
- username color - based on user score
- train model - based on
  - either current week’s score
  - or all time train score

2013_09_18 Meeting Minutes - train game, multiplayer

Agenda:
- Show what we’ve done
- Discuss networking issues

After:
- Confirm date for presentation, email Kiyokawa
- Make an APK for demo - send by tomorrow morning

Notes:
- talk to Kiyokawa sensei about the bikes
- check tablet for android updates
- Send paper Saturday Afternoon
  - week after send it Friday
- Paper - switch to word at some point
  - or Latex
- Speed up production
- SCHEDULE MAIN APP
● figure out how to assign players to games

● Make the best possible user experience you can
  ○ Put Grizzly Bear on the waiting list
  ○ Focus instead on the main app

● Presentation schedule
  ○ ppt Rough draft done Monday
  ○ final draft Monday night
  ○ rehearse Tuesday and Wednesday
  ○ branch code Tuesday (branch = presentation)
  ○ presentation Thursday

---

**2013_09_20 Meeting Minutes - Multiplayer and Main App**

Agenda:

● review what we were assigned to do
  ○ did we complete it?
  ○ did we work on anything else?

● Main App
  ○ Check out mockups
  ○ modify?
  ○ Assign jobs

● Chat
  ○ Whether we should add it back in
  ○ Emoticon, text, or both?

Notes:

● alex: make greyed-out generic person for CC
● andengine - no text fields??
● Jen - ask Lindeman about trading cargo
● can android automatically scan qr codes
● Everyone - talk to Lance Monday or Tuesday about how you’re doing networking
● Lance - don’t use a new packet type for each message type
● Next Friday - check if our games match the descriptions we talked about

Main App Jobs

● User side - **Jen**
  ○ create user account
  ○ login
  ○ profile
  ○ buttons for:
    ■ viewing high scores
● scrolling
  ■ change pw
  ■ change room (can link to “join room” scene)
  ■ log out
  ■ delete account
● Join a Train Room - Lance
  ○ scan qr code - how?
  ○ have drop down list
  ○ how the server handles this
● Main Menu - Merrielle
  ○ buttons and stuff
  ○ More important: joining games
  ○ hosting games
  ○ server side - how does the server put people into games?
  ○ scroll ALL of this
● Chat - Jack
  ○ emoticons?
  ○ text?
  ○ server side - not super hard but you have to do it
  ○ “ready to play” popups / game info pane at bottom

2013_09_24 Meeting Minutes - Sprint week

Agenda:
● We have two weeks until testing
  ○ That means everything has to be DONE
  ○ No “Oh I was having a bug”
  ○ You need to plan for those or be prepared to drop that aspect of the project when it’s not done on time
  ○ Do this by planning to finish early, not on time
● What do we have left to do?
  ○ QR Codes / Main menu - connecting to games and Train Room
    ■ if we can’t connect to other players, our app has no point.
  ○ core game functionality
    ■ This should be done already
    ■ What is left?
      ● winning screens
      ● allotting gold after game
      ● storing data
      ● creature creator win screen
      ● Train game
- events
- stats screen
  - update every minute
- core game networking
  - Should be 90 - 100% done (except Wizard)
  - What is left?
    - hp networking
    - wizard networking
- Database
  - finishing last networking
  - updating games to store and retrieve values
- chat / other users view
  - important, but in a serious emergency can be dropped
- user profile
  - mostly adjusting settings / viewing stats
  - helps with user experience (switching Train Rooms)
- polishing games
  - we want a polished project, but save this for testing / bug fix / polish week
- Art
  - this is in its own parallel universe
  - What’s left?
    - gingerbread man (CC)
    - wizard UI
    - BGs - CC, Amida, Wizard, Animated train, train stats
- What is the most important?
- Schedule for this week
  - This is going to be our sprint week
  - That way hell week isn’t right before our presentation
  - Layout/no networking by Thursday - endish
  - Networking / basic functionality done Monday morning after meeting
  - The rest of the week can be used for tweaking, or freaking out and finishing what you were supposed to get done
  - Paper due Saturday Night again
    - make a new doc so we have the old copy
  - Please talk to me after the meeting if you believe this is unreasonable for your portion of the project
- android vs. andengine
  - android!
- game name christening

Notes:
● how do you call andengine from android?
● go directly into the game
● camera / resolution?
● sending info to andengine

2013_09_25 Meeting Minutes

Agenda:
● Show what we’ve done
  ○ New names:
    ■ Main App: Railroad Runner
    ■ My Train
    ■ Elemancer
    ■ Bomb Pass Panic
    ■ Treasure Trap
    ■ Creature Creator
● Paper revisions?

Notes:
● make a dedicated credits screen
● send bike IDs tonight
● change ordering of main menu
  ○ scroll starting soon
  ○ put profile at bottom
  ○ change “host” to “start a new game”
● Jack: fix CC swiping
  ○ anywhere in the rectangle
● Change to Japanese names in paper

2013_09_27 Meeting Minutes presentation outline

Agenda:
● go over the work we've done
● write the presentation outline
● assign work for presentation
● check if our games match the descriptions we talked about
  ○ later
● Clothes
  ○ Business casual
  ○ GET AN IRON - email kiyokawa sensei
  ○ last resort- get clothes dry cleaned
Notes:
- Jen: add function to store so you can’t go over 100 health or 20 speed
  - use TrainStats class
  - put in vending machine art
- Meeting every day?
- Adjust schedule like crazy - get more agile
- Check out dress clothes over the weekend
- Wednesday - fix orientation issue / possibly break games into separate activities
- Everyone be here from noon - 6pm every day next week

Presentation Outline:
- intro - Merrielle - 1 min
- goals - Jack - 2 mins
- overarching structure - Merrielle - 3-4 mins
  - three main components
  - Users - you and others
  - multiplayer games
  - train game
- game descriptions - 3 minutes per game
  - Treasure Trap - Lance
  - Creature Creator- Jack
  - Bomb Pass Panic- Jen
  - My Train - Alex
  - Elemancer - Jen
- technical stuff -
  - android - Jen - 2 mins
    - game engine
  - networking - Lance - 5 mins
    - kryonet
    - joining Train Rooms
    - joining games
  - database - Merrielle - 4 mins
- art - alex - 4 mins
- goals achieved - Jack - 1-2 mins
- Questions to be prepared for?
  - How’d you design these games?
  - is there sound?
- Demo - 20 mins
- Questions again
Total time: 38 mins, 10 mins questions, 20 mins demo = 68 mins!

2013_10_02 Meeting Minutes multiplayer demo

Agenda
- show work done
- multiplayer demo
- elemancer and bpp done?

Notes
- Alex
  - International sign for railroad tracks/crossing instead of rails
  - Recolor Treasure Trap chests
    - All one color for one variation
  - finish everything by the end of this week
- Lance
  - give a background to the train history buttons
  - polish week: make amida line color more visible (like yellow)
  - commit your stuff
- Merrielle
  - Get MySQL daemon to start automatically
  - init.d
  - hostname
- Jack

Report
- Amida
  - English needs polishing
  - Write in the past tense, not future or present
  - Be more formal

Presentation
- High-level architecture slide
  - Architecture before minigames
- Overall app slide
- Avoid redundancy
- Lots of visuals
  - Diagrams with simple labels
  - Especially for games
- Thanks slide
- End with demo
Polish Week

- Creature Creator
  - Look into swiping
    - Head works well, other areas less responsive

2013_10_04 Meeting Minutes

NAMES:
  Elemancer
  Bomb Pass Panic
  Treasure Trap
  Creature Creator

Agenda:
- go over work
- what is left?
- how to connect everything?

Notes:
- BOMB PASS PANIC
  - timeout error
  - fix animation
  - check winner with server
  - connect to specific connections
- Treasure Trap:
  - pretty much done! :D
- Loading screen
  - Elemancer: loading: fix?
  - Bomb Pass Panic: Swipe from bomb to throw it
- Alex:
  - MAKE THE BOOT
  - chests for gold silver bronze
- How to start the game when I send the signal
- Ending game
  - send back to Main Menu
- Add intents to main menu
  - switch to profile
  - switch to chat
  - Is GameJoin server still connected??
- Elemancer
put in art
connect to specific connections

Vending machine
art

2013_10_09 Meeting minutes - preparing to present

Jack:
• fix orientation problems
• set scale for train sprite
• fixing grey frame
• back button for My Train
• update blue button to vending machine
• chat - multiple connections for one user
• fix “everyone shows up as Rob” bug
• add cc loading screen

Merrielle:
• show list of games in main menu
• set orientation based on gameNameID
• presentation script (sorry, I was working on the main menu)
• look up MySQL dump for later
• put in 4 buttons - 1 per game

Alex:
• elemancer loading screen
• handling presentation
• add “loading” text to splash screen

Lance:
• treasure trap loading screen art
• help merrielle with viewing games

Jen:
• elemancer / bomb pass panic loading screen
• Train Room
  • bomb pass panic
  • elemancer
• art for : elemancer
• fix vending machine art

Presentation: 15 - 20 mins
• Ask Kiyokawa/Takemura sensei to connect w/ professor Lindeman
• PRACTICE DEMO - script for what to do when
  • on paper
- Demo: have document camera?
  - prepopulate db with user1 pw 1234
- call it a senior project, not MQP
- have WPI logo
- and osaka university logo
- 1 slide per game
- networking 2 slides
- db 1 slide
- number slides
- no full sentences in bullets!
- don’t repeat anything
- condense goals to one slide
- change Train Room to Train Room
- get it down to 25 slides
- more pics - esp treasure trap
- No Sound slide - unnecessary

Other:
- it’s a good idea to have a website somewhere
  - not required though

**2013_10_12 Last Friday Meeting**

**Agenda**
- what do we have left before we leave japan?
- what do we have left total?

**Before leaving japan:**
- turn in tablets - Monday or today?
- turn in key cards -
- turn in house keys
- tell housing people we’re leaving
- clean house (once tonight, once over on Monday)

**PROGRAMMING**
- All: get server Jars, get server at WPI
- Merrielle
  - fix main menu bugs
    - delete game isn’t picked up
    - still says “join game”
    - duplicate users
- handle ready bool if server returns error
- test start game
- add ID to join game button
- add view users list (requires Train Room)
- update played game at game launch
- Later: remove demo buttons
  - add Train Room stuff to game server
  - other users / user info page
  - dump db
- Alex
  - nothing
- Jen
  - Wizard
    - only pass to other person in-game (Y)
    - first spell isn’t showing up (Y)
    - why loading as single player (Y)
    - if someone has to block and we don’t hear from them in 67 seconds
      - server declares loss
    - money (Y)
    - user score - count rounds (Y)
    - send score to db (Y)
    - test > 2 games w/ 2 player
  - Hot Potato
    - fix animation
    - have losses handled by server and then passed to everyone
    - potato sometimes doesn’t land on the person
    - if someone has potato and we don’t hear from them in 7 seconds
      - server declares loss
    - money (Y)
    - user score (Y)
    - send score to db (Y)
    - test > 4p
  - vending machine (Y)
    - fix health (Y)
    - fix door and button (Y)
- Jack
  - creature creator
    - user score
    - money
    - game ID back on
- send score to db
  ○ Later: orientation bug
  ○ Chat:
    ■ enter key to send message
    ■ UI bug - don’t let keyboard hide top ui
  ○ Train:
    ■ get it to update when you return from the vending machine
    ■ switch update rate to 30mins
    ■ make game start new game the first time you load it up on monday
    ■ store score
    ■ update userscore
    ■ send train score to db on update

  ● Lance
    ○ high scores screen
    ○ credits screen (button in Profile)
    ○ Train Room - show merrielle
      ■ pass train string as intent
    ○ Treasure Trap
      ■ send scores to db
      ■ update userscore

Before submitting:

  ● paper
    ○ finish rough draft
    ○ edit rough draft
    ○ get it critiqued
    ○ get it critiqued again
    ○ final draft

  ● Returning: Tuesday

  ● Meeting: Wednesday 4pm in the library
    ○ have rough draft done
    ○ SPELL CHECK

2013_10_22 Meeting Minutes

Next Meeting: when either the VM or Lindeman’s review is up
Agenda:

  ● combine anything new
  ● what's left to do?
  ● get images
    ○ zip file, each image title = where it goes
- get appendices
  - send one word doc compiled with all the stuff you want to add
- email about vm
- email about eCDR date
- ask jack about getting his parts in by 11am tomorrow

What’s left:
- Merrielle
  - edit grammar
  - GameJoin section
  - Train Room Stuff
  - Walk in Registrar tomorrow
  - email about devices
- Alex
  - edit grammar
  - paragraph writing
- Jen
  - win and loss HP art
- Lance
  - Event timer - in My Train, every time you load the activity store the current time and the last time it was opened. Based on the difference in time, calculate the percentage of event, roll random number, and display event if necessary
  - figure out how to get rid of the old loading screen
- Jack
  - need writing by 11am tomorrow

**2013_10_25 Meeting Minutes - Paper Editing**

Have everything edited by 12pm Saturday so Alex can send the paper to Lindeman
Next Meeting: 2pm Sunday

- look for
  - grammar
  - flow
  - clarity - can new people understand?
  - consistency
  - technical level
- Ask Lindeman
  - if we need to do testing
  - “keep headings with their respective texts?”
- Alex
  - add to the abstract
  - take second half for flow, starting at Game Engines
  - add a footnote to direct reader to a table of all events
- Merrielle
- fix table of contents and cover formatting
- look at sending userscore db func

- Jen
  - add text between every set of headings
  - something to introduce all the sub sections that come next
  - ask lindeman about player numbers comment
  - change “chances” in elemancer to lives
  - read through first half for flow, stop at game engines

- Jack
  - Write about initial ideas
  - Look up an article to cite android as most popular mobile platform
  - edit CC section
  - write about CC networking (what kind of things do you send to the server?)
    - look at the TT networking section for an example

- Lance
  - Revise the technical detail level in the Treasure Trap description
    - (Make a google doc so I can paste stuff in)

- Everyone
  - images
  - appendices
  - screenshots
    - once the VM is up / we can get devices to test on
    - when we borrow phones have the users test the games
Appendix B: Database Structure

```
mysql -u root -p TrainDB
password: r0b0ts

mysql -u server@localhost -p TrainDB
password: serverpw

service mysqld start

USING PORT: 1111

+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
| Tables in TrainDB |
+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
| game_id           |
| played_with       |
| scores            |
| users             |
+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+

```

Table: game_id

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>varchar(24)</td>
<td>NO</td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>int(1) unsigned</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
```

Table: game_id contents

```
<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>amida</td>
<td>4</td>
</tr>
<tr>
<td>creature creator</td>
<td>3</td>
</tr>
</tbody>
</table>
```
| hot potato | 1 |
| train      | 5 |
| wizard     | 2 |

played_with

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID_HASH</td>
<td>int(8) unsigned zerofill</td>
<td>NO</td>
<td>PRI</td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>plays</td>
<td>int(4) unsigned</td>
<td>NO</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>cargo_trades</td>
<td>int(4) unsigned</td>
<td>NO</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

scores

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>int(8)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>UID</td>
<td>int(4) unsigned zerofill</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>GID</td>
<td>int(1) unsigned</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>score</td>
<td>int(8)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

users

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>int(4) unsigned zerofill</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>username</td>
<td>varchar(20)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>varchar(32)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>user_score</td>
<td>int(12) unsigned</td>
<td>NO</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>current_train_score</td>
<td>int(12) unsigned</td>
<td>NO</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

TABLE CONTENTS
mysql> select * from users;
+---------+---------------+-----------+-----------+-----------------
<table>
<thead>
<tr>
<th>UID</th>
<th>username</th>
<th>password</th>
<th>user_score</th>
<th>current_train_score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>merrielle</td>
<td>applejuice</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0002</td>
<td>testuser</td>
<td>pw</td>
<td>1414</td>
<td>0</td>
</tr>
<tr>
<td>0003</td>
<td>testuser2</td>
<td>pw</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0004</td>
<td>user1</td>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
+---------+---------------+-----------+-----------+-----------------

mysql> select * from played_with;
+-------------+-------+-------------+-------+-----------------
<table>
<thead>
<tr>
<th>UID_HASH</th>
<th>plays</th>
<th>cargo_trades</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00010002</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00020003</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00030004</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
+-------------+-------+-------------+-------+------------------

mysql> select * from scores;
+-----+-----+-----+-----+
<table>
<thead>
<tr>
<th>SID</th>
<th>UID</th>
<th>GID</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0004</td>
<td>2</td>
<td>2345</td>
</tr>
<tr>
<td>2</td>
<td>0004</td>
<td>2</td>
<td>2345</td>
</tr>
<tr>
<td>3</td>
<td>0004</td>
<td>2</td>
<td>2345</td>
</tr>
<tr>
<td>4</td>
<td>0004</td>
<td>2</td>
<td>2345</td>
</tr>
</tbody>
</table>
+-----+-----+-----+-----+
Appendix C: Goal Creation

Features of Game:

Make the program structure
Make the games themselves
Make the program look good
Make the game look good

Login / Logout
Password
Profile?
Highscores
Settings - local
User data - from DB

Play games synchronized with server
Connect to users on same train

SOCIAL:

-------------------
Play games anonymously
Choose friends to play with
Play games alone
View how many games you've played with users
Play with people who don't speak your language

Possible: tell you when to get off the train?
How to leave the game when it’s your stop? (Penalty?)

New Brainstorming:

Based off Jen’s thoughts of FTL being a ‘theme’

Sign in, enter your start and destination

the app uses information online to figure out when you will reach your destination, assuming the train is on time

you automatically enter the persistent train game

you can battle other trains in the minigames as camaraderie with the people on your train

(How are these battles started?)

(This requires games where you can cooperate with the people on your train.)

(With everyone getting off at different stops, it might be too complicated.)

you can play games with people on your train to earn points towards your trains stats, etc

you can also play games by yourself.

(This is entertaining, but how does it fit in with the main theme?)

You can see a list of people online for your train only, but not other trains

you can invite people to play or you can play anonymously in automatically spawned games

the app will warn you when it's time to get off
Appendix D: UI Mockups

Figure 31: Rough UI mockup
This rough UI Mockup shows the Loading, Sign In, Train Room, Main Menu, My Train, Start Game, and Join Game Screens. Note that the Start Game and Join Game screens were originally intended to be popups, then later were changed to new activities.

Figure 32: Rough UI mockup ctd.

This rough UI mockup shows the User Profile, High Score, Change Password, Log Out, Delete Account, Other Users, and Single User Screens. Note that the team originally planned to have a Cargo Trading feature, which was later dropped.
## Appendix E: Initial Game Ideas

Mini game ideas:

<table>
<thead>
<tr>
<th>Name</th>
<th>Light My Fire (or something else if we don't want to pay The Doors)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>Players work together to try to light a fire. Each player is assigned a different task to perform. These tasks can be performed incorrectly, so players must coordinate to win.</td>
</tr>
<tr>
<td><strong>Mechanics</strong></td>
<td>Each player is assigned a different task at the start of the game. Swiping on the screen to rub sticks together. Blowing on the microphone to blow on the embers. Dragging items on the screen to drag kindling into the fire. Tilting the phone to pour on fuel. These tasks can be performed incorrectly, i.e. swiping too fast, blowing too hard, tilting too quickly, etc. If this happens, all players must start over. To win, all players must coordinate and perform their tasks correctly at the right times.</td>
</tr>
<tr>
<td><strong>Num. Players</strong></td>
<td>1-4</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>10-30 seconds</td>
</tr>
<tr>
<td><strong>User Behaviors</strong></td>
<td>Swiping screen Blowing on microphone Tilting</td>
</tr>
<tr>
<td><strong>Phone Assets Used</strong></td>
<td>Touchscreen Microphone Gyroscope</td>
</tr>
<tr>
<td><strong>Server Interaction</strong></td>
<td>Continuous Send user action success boolean Send notification to all users if action unsuccessful</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>Players have to learn to coordinate and work together even if they're not physically together. Also, the setting of the game becomes more and more ridiculous the more players there are, so lasting longer and getting more players to join are also reasons to play.</td>
</tr>
<tr>
<td>Name</td>
<td>Mummy TPing</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Summary</td>
<td>You are a mummy. Flick your mummy wrappings at the house the fastest to TP it!</td>
</tr>
<tr>
<td>Mechanics</td>
<td>Each player is a different color and is randomly selected to start on the left or right side of the screen. A large house (and possibly a tree) are shown in the middle. Swipe from your character in a direction towards the house to TP Faster swipes make the TP go higher. Changing the direction of your swipe changes the direction of the throw. Points are given for time taken to get rid of all of your wrappings, and also for the area of the house/tree you have covered. You can cover up your opponent’s work to reduce their score and increase yours. As you use up your wrapping, it shows on your character. Near the end you look more like a zombie.</td>
</tr>
<tr>
<td>Num. Players</td>
<td>2</td>
</tr>
<tr>
<td>Time</td>
<td>45s – 60s</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>Flicking screen</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>Touchscreen</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>Continuous (send on every x ticks) Send swipe location (incd. Direction) and speed Send indicator if no swipe (in case of lag)</td>
</tr>
<tr>
<td>Motivation</td>
<td>Competition Strategy (some)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Grizzly Bear Waitress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>You are a grizzly bear employed at a restaurant. Serve food to customers! Push your bear in the right direction. Be careful, don’t tip your phone and drop the food! Whoever serves the most plates to the right table wins!</td>
</tr>
<tr>
<td>Mechanics</td>
<td>Top down view of restaurant with tables</td>
</tr>
</tbody>
</table>
Each user has a different color (maybe shown on their apron?)
The table you are delivering to matches your color. This allows users to strategize if they want to bump into other players.
Pick up a plate of food at a designated starting point (probably labeled “Kitchen”)
Move by pushing your character on the touch screen. Movements aren’t exact. Bears are stubborn!
Balance food with the accelerometer - by default food has a small randomized wobble, so leaving your phone on a table won’t work.
Push your character into another character to make them drop their food. If both characters are in a head on collision, both drop food. Turning around to bump into someone head on might take a minute. So it’s a risk!
Bears might possibly be given a small amount of inertia and momentum, depending on playtesting.
When a player touches a table of the correct color, they automatically deliver the food and their score goes up.
It may also be possible to receive points by bumping into other players.

<table>
<thead>
<tr>
<th>Num. Players</th>
<th>1 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>1 - 2 minutes</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>Touching screen</td>
</tr>
<tr>
<td></td>
<td>Tilting phone</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>Touchscreen</td>
</tr>
<tr>
<td></td>
<td>Accelerometer</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>Continuous</td>
</tr>
<tr>
<td></td>
<td>Send swipe location and speed</td>
</tr>
<tr>
<td></td>
<td>Send if no swipe</td>
</tr>
<tr>
<td></td>
<td>Send accelerometer data</td>
</tr>
<tr>
<td>Motivation</td>
<td>Competition</td>
</tr>
<tr>
<td></td>
<td>Strategy</td>
</tr>
<tr>
<td></td>
<td>Skill building - this game isn't easy at first! You'll have to watch out for the train speeding up and stopping.</td>
</tr>
<tr>
<td></td>
<td>Hilarity</td>
</tr>
</tbody>
</table>

| Name               | Hot Potato (or something more creative) |
**Summary**
Set a timer with your friends and start passing around a ball (or other object). Try not to be the one with it when the timer ends.

**Mechanics**
Swipe left or right when you have the ball to pass it in that direction. If you swipe fast enough you can pass it two spaces in that direction instead of one (I may consider instead having two-figure swipes be how you pass it two spaces)

**Num. Players**
3-7 but I think it may be best with 5 or 6

**Time**
Set by the player, probably a minute or 30 seconds

**User Behaviors**
swiping the screen (maybe two finger swipes)

**Phone Assets Used**
touch screen

**Server Interaction**
session id
list of players
start time
end time
who has the potato
who is throwing
who they’re throwing to

**Motivation**
It’s a fast paced and short game that can be done with a large group of people.
Competition
Potential for silly theming or funny consequence animation for the one stuck with the potato/ball/bomb

**Name**
Wizard War, Element Master, Spell Spectacle (or other)

**Summary**
Two wizards set off to settle their differences with a casting battle. There are four elemental spells, each countered by one other. One sends a spell first and the other must counter within a certain time limit to save themselves. Then it is their turn to cast first. If one falters they lose a point. Players can play such that one point lost means they are out, or first to two mess-ups loses. If it goes on for a few rounds with no mistakes than players will send two spells that must be countered in order. The number will increase over time, though at any time the Rwizards may call a draw if needed.
<table>
<thead>
<tr>
<th>Mechanics</th>
<th>Players will either have to draw the symbols, connect dots to form the symbols, or tap certain dots for a given symbol. As of now I’m probably leaning towards the second one.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. Players</td>
<td>1 or 2 (if 1 we can just have the computer sending attacks and it will be a survival for the player)</td>
</tr>
<tr>
<td>Time</td>
<td>Varies, I predict maybe 2 minutes</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>dragging their finger to draw, or tapping the screen</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>touch screen</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>continuous location of finger on touch screen indicator if finger is not on the screen</td>
</tr>
<tr>
<td>Motivation</td>
<td>Competition Skill building (being able to react to the symbols quickly) memory they may like the idea of casting spells (maybe have some interesting animation)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Traveling Garden</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>As a daily event the player can plant a flower, tree, or bush in the garden. He/she also has some amount of resources he/she may choose to give to plants in the garden. When tapped plants may show a username indicating who planted it and maybe some bars showing how much they need each resource. People may be able to look at the garden while away from the train but can only interact with it while on. Maybe each day the user gets a point and can choose different plants to plant using points, so if he wants a tree he/she will not plant that day to save up, but there will be some common flowers that only take one point. He/she can still take care of the plants already there though without spending his/her daily points. Resources used to care for plants refresh each day.</td>
</tr>
<tr>
<td>Mechanics</td>
<td>Touching plants to see details Touching a location to put a new plant</td>
</tr>
</tbody>
</table>
Selecting plants and resources from a menu

<table>
<thead>
<tr>
<th>Num. Players</th>
<th>Any</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Persistent</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>Tapping, maybe dragging</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>Touch Screen</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>Continuous</td>
</tr>
<tr>
<td></td>
<td>Send touch location and speed</td>
</tr>
<tr>
<td></td>
<td>Send if no touch</td>
</tr>
<tr>
<td></td>
<td>Send data on plants</td>
</tr>
<tr>
<td>Motivation</td>
<td>Caring for something over time</td>
</tr>
<tr>
<td></td>
<td>Art/creativity/self expression (can sort of create images with plants)</td>
</tr>
</tbody>
</table>

Name | Washing (something)

Summary | There are generally two kinds of players in this game. The first one is the person putting the detergent on the stuff we want to wash. Players will simply tap the screen to put detergent on. More detergent will give the second player more instructions, which can give the players good score but with longer time. The second player is the people cleaning. Players will follow the instructions shown on the bottom of the screen. The directions will be generated randomly. The final score will be determined by the number of instructions finished properly.

Mechanics | Solo Mode:
| Game start -> The player will get a plate to wash -> The player will clean the plate by following the instructions below. -> When the player done washing the plate, the player will throw the plate out and get one point. -> Washing another plate. The number of instruction will increase with the number of plates washed.
Cooperate Mode:
| Game start -> The first player putting the detergent on the first stuff, and the second player is waiting for the first player to finish work. -> The first player will send the stuff to the second player, and the second player will clean it by following the instructions...
below. Then the first player will get another stuff to put detergent on. The first player can’t pass the stuff to the second player until the player finished cleaning. -> The score will be added to the team after the second player finished washing. -> Everything should be done in a limited amount of time.

| Num. Players | 2 |
| Time         | 60 seconds |
| User Behaviors | Tapping and drawing stuff. |
| Phone Assets Used | Toucing, Data Transfer |
| Server Interaction | Receive/Sends States(e.g. DetergentOn? DoneWashing? GameOver? GameStart?), Important Figures(e.g. Score, How clean was the dish) |
| Motivation | Try to let the users compete and cooperate with each other by some simple, interesting and silly games. |

Sudoku

Other Ideas:

<table>
<thead>
<tr>
<th>Name</th>
<th>Team Train(?) ‘Cause it’s like Steam Train? FTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>A ship/train is traveling down a set path over time toward a goal. The ship/train proceeds automatically, but it can take damage over time and stop. Players must assign crew members to various stations to keep the ship/train going for as long as possible. The game ends in victory when the ship/train reaches the goal.</td>
</tr>
<tr>
<td>Mechanics</td>
<td>When a player joins the game, they are given one or two crew members to assign to various stations. They can make these assignments in any arrangement. Assigning crew to the engines makes the ship/train move toward the goal faster. Weapons makes the ship/train take less damage from random encounters (or with other teams should we implement that). Shields/Hull makes the ship/train take less ambient damage over time. If the</td>
</tr>
</tbody>
</table>
ship/train takes enough damage from any source, it will stop unless a player intervenes. If the ship/train has stopped, a player has to assign a crew member to Repairs to get it moving again. This benefits no systems, but it is required to continue making progress.

<table>
<thead>
<tr>
<th>Num. Players</th>
<th>Any</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>However long it takes to reach the goal. Possibly capped at a few hours for balance?</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>Assign crew members to various stations.</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>Touchscreen</td>
</tr>
<tr>
<td>Server Interaction</td>
<td></td>
</tr>
<tr>
<td>Trigger random story event after a period of time</td>
<td></td>
</tr>
<tr>
<td>Advance ship/train state after a period of time (position and damage)</td>
<td></td>
</tr>
</tbody>
</table>

Motivation
Players can get a sense of camaraderie from working together to crew their own ship/train. Also, once players participate in the game, they can spectate (either from their phones or their computers once they go home or go to work) and get story updates over time. A series of very short and simple but randomly generated and assigned story segments can lead to very interesting stories in the long run. See the original FTL.

<table>
<thead>
<tr>
<th>Name</th>
<th>Person Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Next to each username, show a count of how many times you’ve seen that user</td>
</tr>
<tr>
<td>Mechanics</td>
<td>Increase this counter each time User A logs on in the same location as User B. Do not increase if this login is within ~two hours of last incrementation.</td>
</tr>
<tr>
<td>Num. Players</td>
<td>-</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>-</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>-</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>update whenever any user enters and leaves an area</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Motivation</td>
<td>Social interaction - having seen someone before is a good conversation starter!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Chat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Create a chat room for all users in the same location</td>
</tr>
<tr>
<td>Mechanics</td>
<td>The chat room includes all users either within a certain distance of each other, or connected to the same server (depending on how we manage connections)</td>
</tr>
<tr>
<td>Num. Players</td>
<td>any</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>typing messages to other users</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>touch screen</td>
</tr>
<tr>
<td></td>
<td>GPS (possibly)</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>push updates like any other chat client</td>
</tr>
<tr>
<td>Motivation</td>
<td>Social interaction</td>
</tr>
<tr>
<td></td>
<td>User expression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Persistent Daily Tower (Daily Bonus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Players will get a randomly generated daily bonus one at a day. Then player will put their daily bonus to a tower owned by players with the same lines.</td>
</tr>
<tr>
<td>Mechanics</td>
<td>Everyday at 0am, we will set every players’ state of daily bonus to 0. When the player logged in, the app will set the state of daily bonus to 1 and give player a randomly object to put on the tower, which is categorized by the train routes.</td>
</tr>
<tr>
<td>Num. Players</td>
<td>-</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>User Behaviors</td>
<td>Touch Screen</td>
</tr>
<tr>
<td>Phone Assets Used</td>
<td>Display</td>
</tr>
<tr>
<td>Server Interaction</td>
<td>Store states by different areas, Send Tower status, Receive data of the stuff users want to put on the tower, Generate the tower with existing objects.</td>
</tr>
<tr>
<td>Motivation</td>
<td>Giving users interesting stuff to share on the internet. Let people check in everyday to keep them frequent playing.</td>
</tr>
<tr>
<td></td>
<td>fun</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>Jennifer</strong></td>
<td></td>
</tr>
<tr>
<td>Fire</td>
<td>8</td>
</tr>
<tr>
<td>Mummy Tp</td>
<td>9</td>
</tr>
<tr>
<td>Grizzly Bear</td>
<td>7</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>7</td>
</tr>
<tr>
<td>Wizard</td>
<td>9</td>
</tr>
<tr>
<td>Garden</td>
<td>7</td>
</tr>
<tr>
<td>Washing</td>
<td>8</td>
</tr>
<tr>
<td>Sudoku</td>
<td>7</td>
</tr>
<tr>
<td><strong>FTL theme</strong></td>
<td>10</td>
</tr>
<tr>
<td><strong>Person count</strong></td>
<td>6</td>
</tr>
<tr>
<td>Chat</td>
<td>6</td>
</tr>
<tr>
<td>Daily Tower</td>
<td>7</td>
</tr>
<tr>
<td><strong>Alex</strong></td>
<td></td>
</tr>
<tr>
<td>Fire</td>
<td>4</td>
</tr>
<tr>
<td>Mummy Tp</td>
<td>9</td>
</tr>
<tr>
<td>Grizzly Bear</td>
<td>8</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>10</td>
</tr>
<tr>
<td>Wizard</td>
<td>4</td>
</tr>
<tr>
<td>Garden</td>
<td>6</td>
</tr>
<tr>
<td>Washing</td>
<td>7</td>
</tr>
<tr>
<td>Sudoku</td>
<td>3</td>
</tr>
<tr>
<td><strong>FTL theme</strong></td>
<td>8</td>
</tr>
<tr>
<td><strong>Person count</strong></td>
<td>10</td>
</tr>
<tr>
<td>Chat</td>
<td>8</td>
</tr>
<tr>
<td>Game</td>
<td>fun</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Daily Tower</td>
<td>10</td>
</tr>
<tr>
<td>Merrielle fun</td>
<td>6</td>
</tr>
<tr>
<td>Novelty</td>
<td>5</td>
</tr>
<tr>
<td>Mummy Tp</td>
<td>9</td>
</tr>
<tr>
<td>Grizzly Bear</td>
<td>9</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>9</td>
</tr>
<tr>
<td>Wizard</td>
<td>8</td>
</tr>
<tr>
<td>Garden</td>
<td>7</td>
</tr>
<tr>
<td>Washing</td>
<td>8</td>
</tr>
<tr>
<td>Sudoku</td>
<td>5</td>
</tr>
<tr>
<td>FTL theme</td>
<td>8</td>
</tr>
<tr>
<td>Person count</td>
<td>10</td>
</tr>
<tr>
<td>Chat</td>
<td>8</td>
</tr>
<tr>
<td>Daily Tower</td>
<td>8</td>
</tr>
<tr>
<td>Lance fun</td>
<td>8</td>
</tr>
<tr>
<td>Novelty</td>
<td>7</td>
</tr>
<tr>
<td>Mummy Tp</td>
<td>7</td>
</tr>
<tr>
<td>Grizzly Bear</td>
<td>8</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>7</td>
</tr>
<tr>
<td>Wizard</td>
<td>7</td>
</tr>
<tr>
<td>Garden</td>
<td>7</td>
</tr>
<tr>
<td>Washing</td>
<td>8</td>
</tr>
<tr>
<td>Sudoku</td>
<td>5</td>
</tr>
<tr>
<td>FTL theme</td>
<td>8</td>
</tr>
<tr>
<td>Person count</td>
<td>8</td>
</tr>
<tr>
<td>Chat</td>
<td>7</td>
</tr>
<tr>
<td>Daily Tower</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>fun</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>Fire</td>
<td>7</td>
</tr>
<tr>
<td>Mummy Tp</td>
<td>7</td>
</tr>
<tr>
<td>Grizzly Bear</td>
<td>7</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>7</td>
</tr>
<tr>
<td>Wizard</td>
<td>7</td>
</tr>
<tr>
<td>Garden</td>
<td>6</td>
</tr>
<tr>
<td>Washing</td>
<td>7</td>
</tr>
<tr>
<td>Sudoku</td>
<td>7</td>
</tr>
<tr>
<td>FTL theme</td>
<td>5</td>
</tr>
<tr>
<td>Person count</td>
<td>5</td>
</tr>
<tr>
<td>Chat</td>
<td>5</td>
</tr>
<tr>
<td>Daily Tower</td>
<td>7</td>
</tr>
<tr>
<td>Average</td>
<td>6.6</td>
</tr>
<tr>
<td>Fire</td>
<td></td>
</tr>
<tr>
<td>Mummy Tp</td>
<td>8.2</td>
</tr>
<tr>
<td>Grizzly Bear</td>
<td>7.8</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>8.0</td>
</tr>
<tr>
<td>Wizard</td>
<td>7.0</td>
</tr>
<tr>
<td>Garden</td>
<td>6.6</td>
</tr>
<tr>
<td>Washing</td>
<td>7.6</td>
</tr>
<tr>
<td>Sudoku</td>
<td>5.4</td>
</tr>
<tr>
<td>FTL theme</td>
<td>7.8</td>
</tr>
<tr>
<td>Person count</td>
<td>7.8</td>
</tr>
<tr>
<td>Chat</td>
<td>6.8</td>
</tr>
<tr>
<td>Daily Tower</td>
<td>7.8</td>
</tr>
</tbody>
</table>

**Final Games:**
- Hot Potato
- Grizzly Bear
- Mummy Tp
Wizard
....to be cont.

Other Ideas:
FTL - needs fleshing out
Person Count
Emoticon Status
Amida
Head switcher

Table E 1: Team members were asked to rate each game (from 1 – 10, 10 being the best) based on fun, feasibility, novelty, and coolness in order to pick which games to develop. When several games tied, team members were asked to rank each game (1 being the best) in these same categories. The winning games are seen at the bottom of the table.
## Appendix F: Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Person 1 Interested</th>
<th>Person 2 Interested</th>
<th>Person 3 Interested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>Merrielle</td>
<td>Jack</td>
<td></td>
</tr>
<tr>
<td>Canvas Drawing</td>
<td>Jen</td>
<td>Lance</td>
<td></td>
</tr>
<tr>
<td>Logic Control</td>
<td>Jen</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Jen</td>
<td>Lance</td>
<td>Merrielle</td>
</tr>
<tr>
<td>UI Design</td>
<td>Merrielle</td>
<td>Jack</td>
<td></td>
</tr>
<tr>
<td>Game Design</td>
<td>Jack</td>
<td>Jen</td>
<td>Alex</td>
</tr>
<tr>
<td>Artificial Intelligence</td>
<td>Jack</td>
<td>Jen</td>
<td></td>
</tr>
</tbody>
</table>

### Interest Level:

- High
- Med
- Low

### Alright guys, excluding Alex on Art there are 9 jobs and 4 of us. We should each have 2 jobs total.

One person should have three smaller jobs or if we don’t see anything that balances out one person might be taking on extra work. Though we might think of MORE jobs that we hadn't thought of. Some of the jobs have descriptions in today's minutes.

### Decided

<table>
<thead>
<tr>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Manager</td>
</tr>
<tr>
<td>Networking</td>
</tr>
<tr>
<td>Art</td>
</tr>
</tbody>
</table>

*everyone should research

### Table F.1 Table showing initial interest in project roles

<table>
<thead>
<tr>
<th>Decided</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Design</td>
<td>Everyone</td>
</tr>
<tr>
<td>Architecture</td>
<td>Jack</td>
</tr>
<tr>
<td>Networking</td>
<td>Lance</td>
</tr>
<tr>
<td>Databases</td>
<td>Merrielle</td>
</tr>
<tr>
<td>Game Design</td>
<td>Jen</td>
</tr>
<tr>
<td>Canvas Drawing</td>
<td>Jen</td>
</tr>
<tr>
<td>Art</td>
<td>Alex</td>
</tr>
<tr>
<td>Logic Control</td>
<td>Jen</td>
</tr>
<tr>
<td>AI</td>
<td>Jack</td>
</tr>
<tr>
<td>UI Design</td>
<td>Alex / Merrielle</td>
</tr>
</tbody>
</table>

*everyone should research

*Table F.2 Table showing resulting project roles*
<table>
<thead>
<tr>
<th>Time Estimate (days)</th>
<th>Description</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Total Days</td>
<td>13</td>
</tr>
<tr>
<td>1.5</td>
<td>Plan what to store</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>Table organization</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Protocol</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Method stubs return static data</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Android classes to send / receive data</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>building tables on server</td>
<td>server running</td>
</tr>
<tr>
<td>3</td>
<td>testing / bug fixes</td>
<td></td>
</tr>
<tr>
<td><strong>UI</strong></td>
<td>Total Days</td>
<td>11.5</td>
</tr>
<tr>
<td>0.5</td>
<td>list features</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>learn android design philosophy</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>serveral different layouts</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>user pre-testing</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>programming interface with stock art</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>adding own buttons, fonts, bg, etc</td>
<td>assets have been completed by Alex</td>
</tr>
<tr>
<td>4</td>
<td>bug fixes / user testing / refactoring</td>
<td></td>
</tr>
<tr>
<td><strong>Networking</strong></td>
<td>Total Days</td>
<td>9.5</td>
</tr>
<tr>
<td>1</td>
<td>Finish watching tutorial</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Make basic server-client program - demo</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>set up server on campus</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Testing on real server</td>
<td>Server ready at Osaka University</td>
</tr>
<tr>
<td>1.5</td>
<td>Testing multiple clients</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Building/Testing multiple threads</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>Building/Testing game threads</td>
<td>Game design</td>
</tr>
<tr>
<td><strong>Game Engine</strong></td>
<td>Total Days (Based on 2 people working at the same time)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Game Loop</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Graphics Manager</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Resources Manager</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Input Manager</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Networking hook-up</td>
<td></td>
</tr>
<tr>
<td><strong>AI</strong></td>
<td>Total Days</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Dummy AI</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Basic AI (Path Finding, FSM)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Advanced AI (PCG...)</td>
<td></td>
</tr>
<tr>
<td><strong>Game design</strong></td>
<td>Total Days</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wizard prototype: done, testing = 3 hours</td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>Time Estimate</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Amida prototype</td>
<td>done, testing done</td>
<td></td>
</tr>
<tr>
<td>Hot potato prototype</td>
<td>design: 1 hour testing = 2 hours</td>
<td></td>
</tr>
<tr>
<td>Creature Prototype</td>
<td>Use concept art on flash cards cut in 3. Ask Alex how long concept art takes.</td>
<td></td>
</tr>
<tr>
<td>Mummy Prototype (if not programmed)</td>
<td>2 days to design and get/create proper materials. 1 hour to test (assuming multiple tests).</td>
<td></td>
</tr>
<tr>
<td>Grizzly Prototype (if not programmed)</td>
<td>2 days to design and get/create proper materials. 2 hours to test (assuming multiple tests).</td>
<td></td>
</tr>
<tr>
<td>Train game prototype</td>
<td>3 days to design a board game version, 1 day to create materials to use (if art not required), 4 hours to conduct multiple tests (likely less)</td>
<td></td>
</tr>
<tr>
<td>Writing up specific mechanic docs/game docs</td>
<td>4 days + 1 day to check over</td>
<td></td>
</tr>
</tbody>
</table>

**Logic Control + Canvas drawing**

(initial game coding after engine and networking is in place. Note first attempt = no AI)

- Creature First Attempt = 1 day
- Wizard first attempt = 5 days to get both single player and multiplayer mode
- Amida first attempt = 3 days
- Hot Potato first attempt = 2 days
- Mummy first attempt = 6 days
- Grizzly first attempt = 7 days

*Table F 3 Table showing breakdown and time estimates for each role. This helped tremendously with scheduling.*
Figure F-1: Resulting Gantt Timeline made in Gantt Project. This timeline was continually updated as the project progressed.
Appendix G: My Train Prototype

Instructions:

Intro:
You are a conductor on a train. You need to maintain your train’s health and speed while carrying valuable cargo. The amount of points you earn depends on how far you travel and how much cargo you carry. You earn money through the mystical power of minigames.

Your Goal:
You have 10 hours, and 10 turns. Move your train the greatest distance with the most cargo!

How to do it:
Your train needs two things to make it move: **health** and **speed**. You get these by buying them with in-game money.

**Health**: If your train breaks down, you can't move. Your health ranges from 0 - 100. Health will decrease every turn as everyday wear and tear on your train.

**Warning**: The greater your speed, the more your health will decrease each turn. If your speed is 1, health decreases by 1. A speed of 2 means your health decreases by 2, and so on.

**Speed**: Speed determines the number of spaces you can move each turn. If your speed is 1, you move one space. A speed of 2 moves two spaces, and so on. Speed ranges from 1 - 20.

**Cargo**: You also earn points by buying cargo. If you need to, you can sell cargo back to earn some extra cash. The sale price, however, is lower than the buying price.

**Trading**: For each new player you meet, you can trade cargo once a day. This cargo is marked as foreign cargo, and can be sold for a higher price.

**Money**: Everything in the game can be bought. You earn money by playing minigames!

**Event Cards**: Each turn you have a 50% chance to get an event. Simply draw one and follow the instructions.

**Minigames**: When you play a minigame, you get a chance to win money. Roll a RNG. High rolls win. *Note: In the real Android app each minigame will be an actual minigame.*

**Challenge Wager**: If you’re feeling especially brave, you can bet money to play double or nothing. You cannot wager if you don’t have the cash on hand. Winning a challenge wager...
earns double the usual reward, and you get your wager back. Losing your challenge wager means you lose the money you bet.  

Minigame Table:

<table>
<thead>
<tr>
<th>Game</th>
<th>Reward</th>
<th>Challenge Wager</th>
<th>Chance of Winning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amida</td>
<td>big - 40</td>
<td>20</td>
<td>1/4 big win</td>
</tr>
<tr>
<td></td>
<td>med - 20</td>
<td></td>
<td>1/2 med win</td>
</tr>
<tr>
<td></td>
<td>lose - 0</td>
<td></td>
<td>1/4 lose</td>
</tr>
<tr>
<td>Creature</td>
<td>10</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>Grizzly</td>
<td>big - 60</td>
<td>15</td>
<td>1/4 big win</td>
</tr>
<tr>
<td></td>
<td>med - 20</td>
<td></td>
<td>1/4 med win</td>
</tr>
<tr>
<td></td>
<td>lose - 0</td>
<td></td>
<td>1/2 lose</td>
</tr>
<tr>
<td>Hot Potato</td>
<td>big - 50</td>
<td>20</td>
<td>1/6 big win</td>
</tr>
<tr>
<td></td>
<td>med - 10</td>
<td></td>
<td>4/6 med win</td>
</tr>
<tr>
<td></td>
<td>lose - 0</td>
<td></td>
<td>1/6 lose</td>
</tr>
<tr>
<td>Mummy TP</td>
<td>big - 30</td>
<td>15</td>
<td>1/3 big win</td>
</tr>
<tr>
<td></td>
<td>med - 15</td>
<td></td>
<td>1/3 med win</td>
</tr>
<tr>
<td></td>
<td>lose - 0</td>
<td></td>
<td>1/3 lose</td>
</tr>
<tr>
<td>Wizard</td>
<td>win - 40</td>
<td>15</td>
<td>1/2 win</td>
</tr>
<tr>
<td></td>
<td>lose - 0 (5 in app)</td>
<td></td>
<td>1/2 lose</td>
</tr>
</tbody>
</table>

Prices:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Start</th>
<th>Range</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cargo</td>
<td>0</td>
<td>0 - no max</td>
<td>$10 / 1 cargo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sale price: $5, $50 foreign cargo</td>
</tr>
<tr>
<td>Health</td>
<td>50</td>
<td>0 - 100</td>
<td>$4 / 1 health point</td>
</tr>
<tr>
<td>Speed</td>
<td>1</td>
<td>1 - 20</td>
<td>$25 / 1 speed point</td>
</tr>
</tbody>
</table>

Setup:

• Set the clock at 0
• All players place their tokens on Start
• All players place their health markers at 50
• All players place their speed markers at 1
• All players take $100
• Roll to see who goes first. Continue play clockwise from there.

Each turn:
1. **Event**: Each player rolls to see if they draw an Event Card - 50% chance.
   
   *In app: If the player is uninterested in events, they can dismiss them before looking at what the event is.*

2. **Minigame**: Each player can choose to play a minigame
3. **Money**: Each player invests money as they see fit
   a. Cargo can be traded
   b. Cargo can be sold
4. **Movement**: The train moves according to its Speed and Health
   a. If Health was 0 at the beginning of the turn, movement is 0
   b. Keep a record of the number of laps a train has made
5. **Health**: The train's health decreases according to its Speed
6. **Clock**: The clock ticks forward another hour after all players have moved

Winning:
At the end of 10 turns, count:
• The number of spaces (40 per lap)
• The number of cargo

Your Score = Spaces * (1 + 0.1*Cargo)

The player with the greatest score is the winner!

**Event List**

1. Protective wheel coating: Any traveling done over the next hour (turn) will cost the train two fewer health than usual.
2. Clearance sale at the cargo shop: For the next hour (turn) cargo will cost 2 dollars fewer.
3. There's oil in those hills: You conductor struck oil on her free time! You may either use it to increase the speed of your train by 1 point, or trade it for 2 cargo and 5 dollars.
4. A kind donation: A grateful passenger decided to donate $50 dollars to your train for an excellent experience during their trip.
5. Investors are watching: If you are able to increase your speed by 3 points by the end of the next 2 hours (2 turns) investors will donate $125 to your train.
6. Traveling merchant: A merchant you meet on your travel claims to have very rare merchandise. He agrees to paint your train with a special paint that will give it 20 health if you pay him 50 gold. (Must be done right now, at the beginning of the turn, before playing any more games).

7. Short cut: You may pay 30 gold (Must be done right now, at the beginning of the turn, before playing any more games) to take a secret portion of the track with no stops or detours. As a result at the end of the hour (turn) you will have traveled an additional 10 spaces.

8. Turbulence: The next section of travel is a bit rough at the moment. Either slow down and lose 2 speed points or lose an additional 10 health at the end of the hour (turn).

9. Poorly maintained track: Someone was slacking slightly on track maintenance. At the end of the hour (turn) travel 2 fewer distance and lose an extra health.

10. Train robbers attack: you may either pay them off for 50 gold or loose 6 cargo.

11. You capture a high profile bandit: You know that you can get a $100 reward for turning him in, but it is slightly out of your way and will cause you to cover half distance rounded up this hour (turn). He offers his cargo for his release, and shows that he has 5 cargo on him. You may either take his cargo or turn him in.

12. Train inspection: your train is stopped for inspection. If it has fewer than 30 health you are fined $30 at the end of the hour (turn). If you do not have $30 when the hour (turn) ends you are required to lose 2 speed points. If on the other hand you have over 70 health you are rewarded for excellent care. The inspectors give you fuel that increases your speed by 1.

13. Challenge town: you are passing through a town known for their love of risk and challenge. While in the town (for the next two turns) all challenges become the town’s signature super challenges. This provides 4 times reward instead of two times reward but requires 4 times the wager.

14. Rock slide: There is a rock slide ahead! You must either choose to wait for the path to be clear, causing you to travel half distance this hour (turn), or you can continue through at your own risk. If you continue through you have a 25% chance of losing half your health rounded up (if you have 10 health or fewer remaining you will instead lose all of your health)

15. Strange animal: you see an unheard of species of animal on the side of the road, and know a scientist nearby who would meet up and buy it from you. If you take it onto your train you will get $50 at the beginning of your next visit to the store, but there is a 50% chance of the creature damaging the train for 5 health.
16. I Was Promised Cake: Your crew members are demanding cake. You may either pay $100 to feed them, or lose 15 health.

17. Come On Baby, Hold Together: Your health does not decrease this turn.

18. Barrel Roll: A barrel of cargo rolls across the tracks. You may travel 1 space fewer this hour (turn) to pick it up and gain 3 cargo, or just smash through it.

19. We Rob Banks: You may gain $200 at the cost of 50 health.

20. Runaway Train: For the next hour (turn), you may increase your train’s speed points to maximum at the cost of 66% of your health. Once the hour (turn) ends, return to previous speed.

21. These Go To Eleven: Gain 11 health.

22. Snakes On a Train: Your train has been invaded by murderous snakes. You may lose 30 health to get rid of them now, or lose 15 health per hour (turn) for the next 3 hours (turns).

23. You Spoony Bard!: A random passenger has insulted your honor! You may either beat him up and steal his $50 lunch money, dealing 10 damage to your train, or you can be mature and ignore him.

24. Buy Somethin’, Will Ya?: You find a shopkeeper who will sell you cargo in a buy-one-get-one-half-off-deal. You can do this as many times as you want this hour (turn).

25. Welcome To Warp Zone!: If you choose to use the warp zone, you have a 50% chance of moving forward 20 distance or moving back 10 distance. You may also choose to ignore the warp zone and move on.

26. Sheep crossing: Move three spaces less than usual.

27. Mechanic: You have the option of trading any amount of speed points for 5 health each.

28. Nitrous: Your train moves an extra 5 spaces this turn, and you lose 5 extra health.

29. Energy burst: Your train moves an extra 5 spaces this turn, and you gain 5 health.

30. R&R: Your train moves 5 fewer spaces this turn, but you gain 5 health.

31. Hot Potato: If you play hot potato this turn, winning pays double.

32. Time warp: Minigames pay double this turn.

33. Rogue Monkeys: Your train is attacked! Lose 7 health or don’t spend any money this turn.

34. Double Rainbow: Draw two more event cards

35. Hidden Treasure: You find a treasure map and see that there’s a chest buried nearby. You can choose to look for it, in which case you travel 4 spaces less than usual. You then have a 50% chance of getting either $150 or $0.
36. Yer a wizard!: A wizard is you! If you play the wizard game this turn and win, you conjure 6 cargo.

37. Throw it Overboard: You can choose to drop 4 cargo for 2 extra speed points.

38. Plague: A nasty plague runs rampant on your train. If you keep going, you will lose 30 health. Otherwise you must stop to find a doctor, and your movement is 0 for the turn. However, the passengers reward you with $100 for providing aid.


40. Monsters: Monsters appears on the train! Create a new creature (by playing Creature Creator) to defeat it! You will get 3 cargos (of monster meat) after the game.

41. The legendary engineer: An engineer uses magic to repair your train. Gain 10 health.

42. The legendary driver: A super elite driver came to show the newbies how to drive a train! Gain 2 speed points.

43. Superheroes on the train: Famous Japanese superheroes are here! Everyone is distracted, including the driver. Go 3 fewer distance this hour (turn).

44. Big Campaign: Draw 2 events and pick 1. Shuffle the other back into the deck.

45. Thunderstruck: The power plant has been struck by lightning. Lose 3 speed points.


47. Cellphone issue: Someone is talking on the phone on the train! Including me! If you play Hot Potato to blow them up, the other passengers give you $10.

48. Poltergeist Pillagers: Ghosts are on the train and they stole 6 cargo!
Appendix H: My Train Final Events

1. Short cut: You may pay 30 gold (Must be done right now, at the beginning of the turn, before playing any more games) to take a secret portion of the track with no stops or detours. As a result at the end of the hour (turn) you will have traveled an additional 10 spaces.
2. Poorly maintained track: Someone was slacking slightly on track maintenance. At the end of the hour (turn) travel 2 fewer distance and lose an extra health.
3. Strange animal: you see an unheard of species of animal on the side of the road, and know a scientist nearby who would meet up and buy it from you. If you take it onto your train you will get $50 at the beginning of your next visit to the store, but there is a 50% chance of the creature damaging the train for 5 health.
4. Come On Baby, Hold Together: Your health does not decrease this turn.
5. Superheroes on the train: Famous Japanese superheroes are here! Everyone is distracted, including the driver. Go 3 fewer distance this hour (turn).
6. A kind donation: A grateful passenger decided to donate $50 dollars to your train for an excellent experience during their trip.
7. Sheep crossing: Move three spaces less than usual.
10. Thunderstruck: The power plant has been struck by lightning. Lose 3 speed points.
11. Poltergeist Pillagers: Ghosts are on the train and they stole 6 cargo!
## Appendix I: Art Asset List

<table>
<thead>
<tr>
<th><strong>Main Application</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>railroad runner logo XS.png</td>
</tr>
<tr>
<td>railroad runner logo S.png</td>
</tr>
<tr>
<td>railroad runner logo M.png</td>
</tr>
<tr>
<td>railroad runner logo L.png</td>
</tr>
<tr>
<td>railroad runner logo icon.png</td>
</tr>
<tr>
<td>next round.png</td>
</tr>
<tr>
<td>portrait lose.png</td>
</tr>
<tr>
<td>landscape lose.png</td>
</tr>
<tr>
<td>landscape win.png</td>
</tr>
<tr>
<td>portrait win.png</td>
</tr>
<tr>
<td>elemancer loading v2.png</td>
</tr>
<tr>
<td>bomb pass loading.png</td>
</tr>
<tr>
<td>splash.png</td>
</tr>
<tr>
<td>portrait loading.png</td>
</tr>
<tr>
<td>landscape loading.png</td>
</tr>
<tr>
<td>treasure trap loading.png</td>
</tr>
<tr>
<td>elemancer loading.png</td>
</tr>
<tr>
<td>creature creator loading.png</td>
</tr>
<tr>
<td>main app background.png</td>
</tr>
<tr>
<td>main button unpressed.9.png</td>
</tr>
<tr>
<td>main button pressed.9.png</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Treasure Trap</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>bronze chest.png</td>
</tr>
<tr>
<td>boot.png</td>
</tr>
<tr>
<td>silver chest.png</td>
</tr>
<tr>
<td>gold chest.png</td>
</tr>
<tr>
<td>amida background.png</td>
</tr>
<tr>
<td>amida character 1.png</td>
</tr>
<tr>
<td>amida character 2.png</td>
</tr>
<tr>
<td>amida character 3.png</td>
</tr>
<tr>
<td>amida character 4.png</td>
</tr>
<tr>
<td>amida coin.png</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Creature Creator</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>background.png</td>
</tr>
<tr>
<td>creature creator head sheet v2.png</td>
</tr>
<tr>
<td>creature creator body sheet v2.png</td>
</tr>
<tr>
<td>creature creator legs sheet v2.png</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Bomb Pass Panic</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>hot potato background landscape.png</td>
</tr>
<tr>
<td>bomb sheet v2.png</td>
</tr>
<tr>
<td>potato char 1.png</td>
</tr>
<tr>
<td>File Name</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>generic ry background.png</td>
</tr>
<tr>
<td>generic po background.png</td>
</tr>
<tr>
<td>generic gb background.png</td>
</tr>
<tr>
<td>generic bw background.png</td>
</tr>
</tbody>
</table>