



WPI

Errant: An iOS Fitness RPG

A Major Qualifying Project Report
Submitted to the Faculty of the
Worcester Polytechnic Institute
In partial fulfillment of the requirements for the
Degree of Bachelor of Science
On November 10, 2014

Submitted by: Maxwell Perlman and Stefan Alexander

Advised by: Professor Robert Lindeman

Abstract

Errant: An iOS Fitness RPG

By

Maxwell Perlman and Stefan Alexander

Through the use of the iPhone's new M7 chip, in conjunction with iBeacon Bluetooth devices, we created an iPhone role-playing game that is meant to encourage players to have a healthier lifestyle. The player engages in a real world role-playing game experience where the way they exercise in reality impacts their digital avatar's traits and abilities. The goal of this game is to encourage exercise through in-game benefits to the player's character, with the hope of eventually changing the player's lifestyle to incorporate more exercise.

Acknowledgements

This project could not have been completed without assistance from the following Ritsumeikan University students and faculty, as well as all of the Ritsumeikan University play testers and all Media Experience Design Lab members:

- Professor Noma
- Wakao Asuka
- Kohno Hitomi

Additionally, we would like to thank Alex Clemens and Professor Lindeman:

- Alex Clemens
- Professor Lindeman

Table of Contents

| | |
|--|------|
| Abstract..... | ii |
| Acknowledgements | iii |
| List of Figures | vi |
| List of Tables | viii |
| 1: Introduction..... | 1 |
| 1.1: Project Proposal and Purpose..... | 1 |
| 1.2: Gameplay Inspirations and Related Works | 1 |
| 1.2.1: RPG and JRPG Genres | 1 |
| 1.2.2: <i>Ingress</i> | 1 |
| 1.2.3: <i>Dungeons and Dragons</i> | 2 |
| 1.2.4: <i>Find Mii</i> | 2 |
| 1.3: Team Members Roles | 2 |
| 2: Development..... | 4 |
| 2.1: Xcode and iOS..... | 4 |
| 2.2: Swift and Objective-C | 4 |
| 2.3: Frameworks | 4 |
| 2.3.1: Google Maps SDK for iOS..... | 4 |
| 2.3.2: SpriteKit..... | 5 |
| 2.3.4: CoreMotion..... | 5 |
| 2.3.5: CoreLocation | 5 |
| 2.3.6: Accounts | 5 |
| 2.4: Delegates..... | 5 |
| 2.4.1: UIPickerViewDelegate | 6 |
| 2.4.2: CLLocationManagerDelegate | 6 |
| 2.4.3: Notifications..... | 6 |
| 2.4.4: UserDefaults..... | 7 |
| 2.4.5: NSTimer | 7 |
| 2.4.6: AppDelegate | 7 |
| 2.4.7: Server and Database | 7 |

| | |
|--|----|
| 2.5: iBeacon Devices and Multiplayer Gameplay | 8 |
| 2.6: HMSideMenu..... | 11 |
| 3: Design | 12 |
| 3.1: Initial Design Decisions | 12 |
| 3.1.1: Understanding Our Audience | 12 |
| 3.1.2: American and Japanese Audience | 14 |
| 3.1.3: Violent vs. Non-Violent Gameplay | 14 |
| 3.1.4: Experiential Goals | 14 |
| 3.2: Combat System..... | 14 |
| 3.2.1: Entities | 14 |
| 3.2.2: Traits | 15 |
| 3.2.3: Player..... | 15 |
| 3.2.4: Character Growth..... | 15 |
| 3.2.5: Growth through exercise..... | 15 |
| 3.2.6: Growth through gameplay | 16 |
| 3.2.7: Player Actions and Abilities..... | 16 |
| 3.2.8: Enemies | 20 |
| 3.2.9: Possible Growth Rates | 20 |
| 3.2.10: Actions | 26 |
| 3.2.11: Types..... | 29 |
| 3.3: Testing Procedure..... | 39 |
| 3.3.1: Computer Automated Testing..... | 39 |
| 3.3.2: Player Based Testing..... | 40 |
| 3.4: Future design areas and expansion possibilities | 40 |
| 4: Gameplay Experience Description..... | 41 |
| 4.1: First gameplay experience..... | 41 |
| 4.2: Subsequent gameplay experiences | 45 |
| 5: Reference Materials and Tools | 55 |

List of Figures

| | |
|--|-----------|
| <i>Figure 1 – SNS account selection screen</i> | <i>6</i> |
| <i>Figure 2 – Map and pedometer screen</i> | <i>8</i> |
| <i>Figure 3 – Multiplayer setup screen, waiting for peers</i> | <i>9</i> |
| <i>Figure 4 – Battle screen with magical action menu (right) open</i> | <i>10</i> |
| <i>Figure 5 – Age demographics of Candy Crush Saga players</i> | <i>13</i> |
| <i>Figure 6 – Age demographics of Bejewled Blitz players⁸</i> | <i>13</i> |
| <i>Figure 7 – Physical action icon</i> | <i>17</i> |
| <i>Figure 8 – Magic action icon</i> | <i>18</i> |
| <i>Figure 9 – Examine action icon</i> | <i>20</i> |
| <i>Figure 10 – Archer</i> | <i>30</i> |
| <i>Figure 11 – Knight</i> | <i>30</i> |
| <i>Figure 12 – Ninja</i> | <i>30</i> |
| <i>Figure 13 – Wurm</i> | <i>31</i> |
| <i>Figure 14 - Spider</i> | <i>31</i> |
| <i>Figure 15 - Beetle</i> | <i>31</i> |
| <i>Figure 16 – Oorn</i> | <i>32</i> |
| <i>Figure 17 – Mi-go</i> | <i>32</i> |
| <i>Figure 18 – Yugg</i> | <i>32</i> |
| <i>Figure 19 - Minotaur</i> | <i>33</i> |
| <i>Figure 20 - Cerberus</i> | <i>33</i> |
| <i>Figure 21 - Griffin</i> | <i>33</i> |
| <i>Figure 22 – Barrier</i> | <i>34</i> |
| <i>Figure 23 – Shell</i> | <i>34</i> |
| <i>Figure 24 – Husk</i> | <i>34</i> |
| <i>Figure 25 – Zombie</i> | <i>35</i> |
| <i>Figure 26 – Chimera</i> | <i>35</i> |
| <i>Figure 27 – Undead Soldier</i> | <i>35</i> |
| <i>Figure 28 – Fire</i> | <i>36</i> |
| <i>Figure 29 – Water</i> | <i>36</i> |
| <i>Figure 30 – Wind</i> | <i>36</i> |
| <i>Figure 31 – Earth</i> | <i>36</i> |
| <i>Figure 32 – Shoggoth</i> | <i>37</i> |
| <i>Figure 33 – Brain Slug</i> | <i>37</i> |
| <i>Figure 34 – Ooze</i> | <i>37</i> |
| <i>Figure 35 – Djinn</i> | <i>38</i> |

| | |
|---|-----------|
| <i>Figure 36 – Vampire</i> | <i>38</i> |
| <i>Figure 37 – Angel of Death</i> | <i>38</i> |
| <i>Figure 38 – Wyrn</i> | <i>39</i> |
| <i>Figure 39 – Sea Serpent</i> | <i>39</i> |
| <i>Figure 40 – Dragon</i> | <i>39</i> |
| <i>Figure 41 – Title screen.</i> | <i>41</i> |
| <i>Figure 42 – Account creation screen, not logged in</i> | <i>42</i> |
| <i>Figure 43 – Account creation screen, logged in with Twitter</i> | <i>43</i> |
| <i>Figure 44 – Character creation screen</i> | <i>43</i> |
| <i>Figure 45 – Username selection screen</i> | <i>44</i> |
| <i>Figure 46 – Account creation screen, logged in with Errant account</i> | <i>45</i> |
| <i>Figure 47 – Status screen</i> | <i>46</i> |
| <i>Figure 48 – Battle screen</i> | <i>47</i> |
| <i>Figure 49 – The ocean background</i> | <i>48</i> |
| <i>Figure 50 – The city background</i> | <i>48</i> |
| <i>Figure 51 – The mountain background</i> | <i>49</i> |
| <i>Figure 52 – Battle screen with physical action menu (right) open</i> | <i>50</i> |
| <i>Figure 53 – Battle screen, “Examine” used by player</i> | <i>50</i> |
| <i>Figure 54 – Results screen when winning a battle</i> | <i>51</i> |
| <i>Figure 55 – Results screen when losing a battle</i> | <i>51</i> |
| <i>Figure 56 – Results screen when winning a battle and leveling up</i> | <i>52</i> |
| <i>Figure 57 – Status assignment screen shown after leveling up</i> | <i>53</i> |
| <i>Figure 58 – Multiplayer setup screen</i> | <i>54</i> |

List of Tables

| | |
|--|-----------|
| <i>Table 1 – Physical player actions</i> | <i>17</i> |
| <i>Table 2 – Magic player actions</i> | <i>19</i> |
| <i>Table 3 – Very low trait value table: levels 1-10</i> | <i>21</i> |
| <i>Table 4 – Low trait value table: levels 1-10</i> | <i>21</i> |
| <i>Table 5 – Medium trait value table: levels 1-10</i> | <i>22</i> |
| <i>Table 6 – High trait value table: levels 1-10</i> | <i>23</i> |
| <i>Table 7 – Very high trait value table: levels 1-10</i> | <i>23</i> |
| <i>Table 8 – Final trait value table for logarithmic growth</i> | <i>24</i> |
| <i>Table 9 – Final trait value table for linear growth</i> | <i>25</i> |
| <i>Table 10 – Final trait value table for exponential growth</i> | <i>25</i> |
| <i>Table 11 – Enemy actions</i> | <i>26</i> |

1: Introduction

1.1: Project Proposal and Purpose

This project seeks to provide a real world role-playing game experience with the purpose of encouraging exercise and a healthier lifestyle. Combining both traditional role-playing game concepts and the real-world location and exercise monitoring of the M7 chip found on new iOS devices, we created a unique gameplay experience where the player feels that they truly are the character being represented on their device. We hope this inspires players to go out into the world to explore play with others.

1.2: Gameplay Inspirations and Related Works

This project takes inspiration, in terms of both its concept as well as its gameplay, from numerous sources. The following are the major inspirations and influences behind this game.

1.2.1: RPG and JRPG Genres

Some of the major inspirations for this game are games from the role-playing game (RPG) and Japanese role-playing game (JRPG) genres. From these genres we were able to determine the primary method of gameplay that our players would be engaging in (turn based combat). Games such as *Bravely Default*¹ and games from the *Final Fantasy*² series number among our major influences.

1.2.2: Ingress

*Ingress*³ is a multiplayer augmented reality game. Players are divided into two factions and these factions compete to take control of the world. Players use their cell phones in order to set up virtual towers and capture areas of the Earth. Players try to increase the potency and range of their factions control while waging war on the opposing faction. *Ingress* was a major inspiration for this project because of its excellent mapping and use of real-world locations in gameplay. Our goal was to provide an experience with a similar use of real-world locations while at the same time not allowing for some of the poor gameplay habits that *Ingress* gave rise to, specifically playing the game whilst driving.

¹ [ブレイブリーデフォルト フライングフェアリー]

² [ファイナルファンタジー]

³ [Ingress]

1.2.3: *Dungeons and Dragons*

*Dungeons and Dragons*⁴ is tabletop fantasy role-playing game that is typically played by groups of four to eight people. Normally the game features very few physical resources, typically only a set of die and pieces of paper. Guided by a non-playing person (known as the dungeon master) player's band together to travel through elaborate worlds and dungeons, defeat monsters, discover treasure, and even communicate with non-player characters.

We found inspiration through *Dungeons and Dragons* due its high levels of immersion and cooperative gameplay, deep character creation mechanisms, and dynamic difficulty. Our goal was to try and achieve this level of immersion, while not allowing for a gameplay session to consist of hours of sedentary gameplay, which *Dungeons and Dragons* lends itself to.

1.2.4: *Find Mii*

*Find Mii*⁵ is a game built into the Nintendo 3DS handheld system. By leaving their system in sleep mode and bringing it with them during their day-to-day activities, players can collect other 3DS users' characters. Later, these characters can be used to defeat monsters and obtain treasure, which can be shown off to other players. We found *Find Mii* very inspiring in its multiplayer gameplay mechanics as well as its simplicity.

1.3: Team Members Roles

Maxwell Perlman was responsible for the majority of design decisions made regarding *Errant*. He was in charge of designing all of the game's gameplay systems, most prominently the design and balance of the combat system. Additionally, he was responsible for the design and coding of multiple user interfaces, all server-side database programming, and all web programming. Additionally, he was in charge of maintaining the artistic vision of the game and communicating that vision to the artists.

Stefan Alexander was responsible for a large portion of *Errant*'s development. He was in charge of maintaining all development frameworks across different iOS versions, creating multiple user-interfaces, was responsible for all client-side database code, and maintained the code across multiple versions of the language, Swift, which was in development.

Asuka Wakao was responsible for assisting both Maxwell Perlman and Stefan Alexander with iOS development. Additionally, she developed the communication between iOS devices and iBeacon devices used in *Errant*.

⁴ [D&D Official Homepage]

⁵ [ニンテンドー3DS | すれちがい Mii 広場 : 無料で楽しめるゲーム]

Kohno Hitomi was a voluntary artist in charge of all sprite art for *Errant*. She received artistic direction and produced the art for the enemies according to the given descriptions.

Alex Clemens was a voluntary artist in charge of all user interface art for *Errant*. He produced all assets from a list of all necessary buttons, menu blocks, and background art given a brief description of each.

2: Development

This section will discuss the development of *Errant* from the different technologies used to the development process itself. All code produced for *Errant* is available in GitHub.⁶

2.1: Xcode and iOS

Due to the limited period of time over which development of *Errant* could take place, it was deemed best to focus upon developing *Errant* for a single platform. Due to the nature of *Errant*'s gameplay requirements, iOS devices were the focus of development. This is primarily due to the M7 chip found in iOS devices in conjunction with their compatibility with iBeacon devices. In order to develop for iOS devices, Xcode is necessary. For testing purposes, Xcode was used along with two iPhone 5s devices, one of which using iOS 7 and the other using iOS 8.

2.2: Swift and Objective-C

Xcode currently natively supports both Objective-C and Swift for development. Due to Objective-C's steep learning curve and unique syntax, Swift was used for the development of this application.

For the majority of development, the Swift language was in beta, which resulted in new versions of the language being released frequently throughout development. This caused numerous errors to occur as the language developed and the project progressed.

2.3: Frameworks

Apple provides developers a group of development libraries called frameworks. *Errant* makes use of several of these. They are used to assist in the development and use of specific types of applications.

2.3.1: Google Maps SDK for iOS

In addition to the native Apple development frameworks, the Google Maps SDK was used instead Apple's native map framework, MapKit. Google Maps' SDK was used over MapKit for its reliability, simple and well documented API, as well as numerous smaller but significant features, most importantly the changing of pin colors for important objects on the map.

⁶ <https://github.com/MxD-lab/TekuGame>

2.3.2: SpriteKit

Due to the simplicity of our games visual requirements, the SpriteKit framework was chosen over other graphics methods, namely OpenGL ES. SpriteKit also lent itself to presenting the combat centric gameplay in a manner similar to that of traditional role-playing games.

2.3.4: CoreMotion

Newer iPhone and iPad devices feature a special chip, either the M7 or M8 chip depending on the device. These chips accumulate data while the device is not in use. For *Errant*, the primary use of this functionality is to track the pedometer data of the device. The pedometer data provided tracks the number of steps a user has taken, their location, and their method of travel (walking, running, or by automobile). Due to these chips not being present on older devices, *Errant* is not targeted towards those devices.

2.3.5: CoreLocation

In order to use iBeacon devices, the CoreLocation framework was necessary. This framework allowed for *Errant* to search for iBeacon devices within Bluetooth range of the device and report back the identification number and approximate distance from the device the nearest iBeacon devices was.

2.3.6: Accounts

In order to distinguish between players, players have two different options for storing their player data on the *Errant* server. Players can select between associating their Twitter accounts (already registered with their device) and creating an account solely for *Errant*. The Accounts framework allows for the use of a number of Accounts (including Twitter) without the need to worry about the account's password. If the player decides to create an account solely for *Errant*, the username they create is stored to their phone as well as our server. This method is used to ensure that a username is never created twice and also avoids the need to store a user's password on the server.

2.4: Delegates

In order to expand upon the default behaviors an application provides, delegate functions must be used. In iOS development, delegates used to customize the behavior of elements that are normally not handled directly by developer. Both the UIPickerViewDelegate and CLLocationManagerDelegate were used.

2.4.1: UIPickerViewDelegate

The UIPickerViewDelegate is used to programmatically create a picker and customize its appearance and the data used to fill the picker. The UI element shown in the middle of *Figure 1* is an example of a picker used to store and display the available Twitter accounts associated with a player's device. Due to this data being dynamic, the delegate functions control the number of rows necessary for a component of the picker as well as the number of components in the picker. Additionally, delegate functions are used to customize the picker's font style, font size, font color, text alignment and other formatting options.



Figure 1 – SNS account selection screen

2.4.2: CLLocationManagerDelegate

The CLLocationManagerDelegate functions were used to customize the behavior of the device's detection of iBeacon devices. The delegate functions are used to specify the behavior of the application when the device is in the range of an iBeacon device. The delegate function creates an array of the iBeacon devices within in range, sorts them based on their approximate distance from the users' device and selects the iBeacon device that is closest. A bar is shown which represents the user's approximate distance from the nearest iBeacon device. An iBeacon device is considered out of the range of the user's device if it is further than 20 meters away.

2.4.3: Notifications

There are two methods of providing notifications to the user. Remote notifications are sent to the player from a remote server while local notifications originate from the users' device itself.

Errant uses only local notifications. A player will receive a notification each time they encounter an enemy or one of their traits' values increases through exercise. The notifications are sent by delegate functions that are run in the background of the device.

2.4.4: NSUserDefaults

To avoid the player having to constantly log into *Errant* in order to play, the player's username is stored on their phone using NSUserDefaults. NSUserDefaults is a programmatic way to save the user's preferences to their device. The data stored persists even if the app is terminated. The data stored can be of any type and is stored even if the user closes the application. Due to the limited amount of data a device running the new operating system can store in NSUserDefaults, the only data stored here on devices running iOS8 and forward is the player's username, all other data is stored on the *Errant* server, requiring an internet connection. On older devices, a player's username, trait information, level, camera location, and pedometer values are stored.

2.4.5: NSTimer

NSTimer is used to create delays between segments of code being run. This is done to ensure that the player is able to read all status messages during combat before another is shown. It is also used to ensure a player met the goals of increasing their trait value while exercising without the achievement taking place more than once.

2.4.6: AppDelegate

The AppDelegate is where developers specify what happens when application transitions between different states. These states include when the application launched, when the application turned active, when the application turned inactive, when the application when to the background, when the application when to the foreground, and when the application terminated.

2.4.7: Server and Database

The *Errant* server utilizes a LAMP stack with its database handled through phpMyAdmin. The database consists of seven tables that maintain all iBeacon based battles, the enemies in iBeacon battles, iBeacon device locations, player names and locations, player data, the association between players and iBeacon based battles, and all playtest data. Utilizing numerous PHP forms, data is easily inserted into the database. Once inserted into the database; a JSON file containing the contents of each table of the database is created. These JSON files are then read by *Errant* to provide the application with any necessary information.

2.5: iBeacon Devices and Multiplayer Gameplay

Errant makes use of iBeacon devices, connecting to the user's device over a Bluetooth connection, as a marker for multiplayer battles. The *Errant* database contains a table consisting of the iBeacon's identification number, its longitude and latitude, and the number of players needed to trigger the encounter at the iBeacon. When on the map screen, shown in *Figure 2*, a player can select an iBeacon to see its identification number as well as the number of players needed to trigger the encounter at the iBeacon. If the player is within the range of the iBeacon, the meter indicating their distance from the device will begin to fill in. Once the player is within five meters of the device (approximated by the strength of the player's device's Bluetooth connection) the player will be able to join the waiting room for the encounter. Once a player joins the waiting room for an iBeacon device, that information is passed to the *Errant* server and is appropriately stored in the database. Once the required number of players is present, the players are transitioned into multiplayer combat with a single enemy.

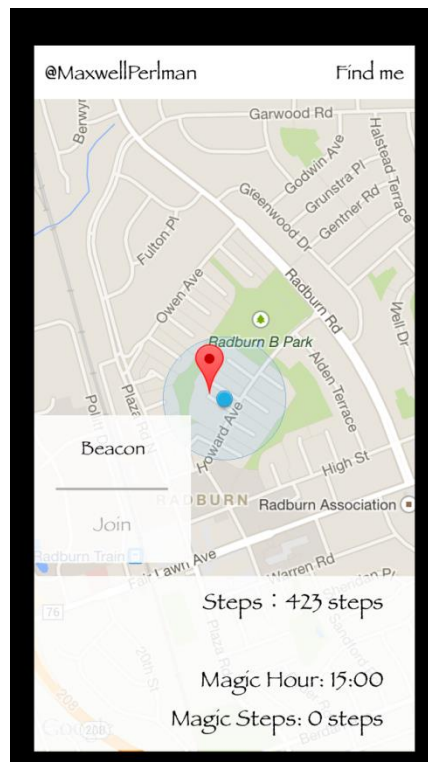


Figure 2 – Map and pedometer screen

Due to the inconsistency of cross iPhone communication via Bluetooth, multiplayer combat is reliant upon the *Errant* database to communicate necessary information to all players in combat. Each iBeacon device has a row of a database table associated with it in the *Errant* database to store the battle. The table stores the battle's identification number, the last action issued, the target of the last issued action, the ID of the target of the action, the enemy's current trait values (health, strength, magic, and speed), the ID of the current player whose turn it is, the last player to take an action, the

current player's trait values (health, strength, magic and speed), and the status of the battle (used to determine if the battle is still going on or has concluded). These values are displayed to the players when they are waiting for other players to join the multiplayer battle as shown in *Figure 3*.

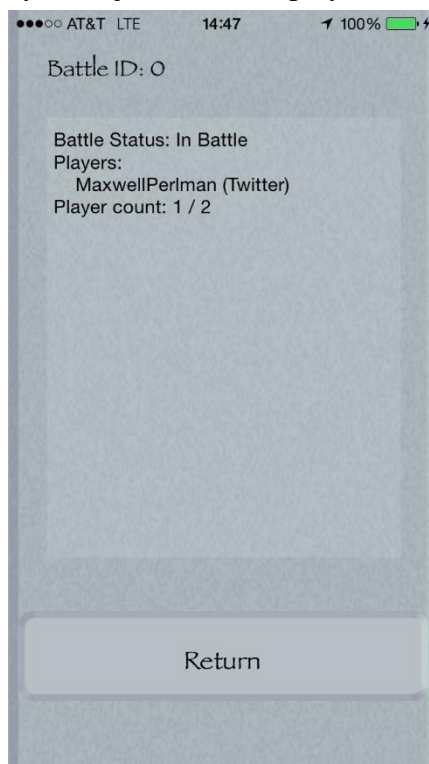


Figure 3 – Multiplayer setup screen, waiting for peers

Once the correct number of players have joined the iBeacon's battle, a host for the battle is determined (the first ID of the player ID's sorted alphabetically becomes the host). Next the host player is transitioned to the encounter screen and all non-host players' trait values are posted to the *Errant* database. Here, the host player receives the ID's and trait values of each player at the iBeacon and sorts the participating players by their speed trait which is the order of players to take action in combat. Additionally, the host player will then randomly generate the type of enemy to be fought and upload all relevant information to the *Errant* server. Once the server has received all the necessary information, the non-host players will transition to the encounter screen shown in *Figure 4*.



Figure 4 – Battle screen with magical action menu (right) open

Once all players have transitioned to the encounter screen, the setup for each of the client players takes place. The setup for the encounter has the client players set the enemy shown (including its trait values) to be the one generated by the host by reading the JSON file from the *Errant* server. Then the update function (run every frame) will begin executing for each player. The update function is the main function in which the game logic is performed.

First, the update function will check if the turn variable is set, if it is not set, it will iterate through all players in the encounter, skipping over those whose health are equal to zero, to determine the next player to take an action.

If it is the enemy's turn, the enemy (controlled by the host player's device) will select a random player and perform one of its possible actions. The status bar on the top of the encounter screen will show all players that it is the enemy's turn. The information about the target and is then sent to the *Errant* server to be read by the host for an action outcome calculation. The results of that calculation are then sent to the server to be read by the other players. The status bar of each player will show the target of the enemy action and the results of said action. Once the next player to perform an action is determined, that information is sent to the database to be received by all players.

Otherwise, if it is a player's turn, that player's action selection menus will appear. The status bar on the top of the encounter screen will show each player the name of the player whose turn it is. Once the player has selected an action to perform, the resulting damage and other effects of the action are sent to the server to be read by the host and other players. Each player is then show through their status bar the result of the action. If the player whose turn it is was a client player, the

host reads a JSON file from the *Errant* server to check if the action was performed. Once that action has been successfully received by the host player, the host player will set the enemy's trait values to the new values from the server.

Finally, each time the update function is run on each player's device, that player's health trait value is checked. If that player's health is zero, that player is eliminated from combat and has lost the fight, however the fight will continue until all players have lost the fight or any number of players have defeated the enemy. Once the fight is over, the database's value for that iBeacon's battle status is set to allow other players to use the iBeacon. The players are then transitioned to the results screen and either rewarded for their victory or punished for their defeat.

2.6: HMSideMenu

Errant makes use of a menu system, HMSideMenu⁷, borrowed from GitHub for the combat menus. This library allows developers to easily integrate animated menus that slide in from any direction into their iOS project.

⁷ [HeshamMegid]

3: Design

This section will cover every aspect of the design and balancing of *Errant*, from initial design decisions to final play testing results and their impact on the design of *Errant*.

3.1: Initial Design Decisions

This subsection will address the initial design consideration that took place before development began.

3.1.1: Understanding Our Audience

The first major consideration when the design process began was to determine our target audience. We determined our audience to be between 21 and 50 years of age. This age group was determined by looking at casual games market research regarding age groups. The image shown in *Figure 5* shows that the age demographics for casual games, in this case *Candy Crush Saga* encompass players from age 10 to 65, with a large focus on players between the ages of 21 and 50. Additionally, we concluded that there were two major subgroups of players that we would be focusing on: those who are looking for a multiplayer gameplay experience (typically players who play turn based social games such as *Words with Friends*) and those who are looking for encouragement to exercise more (with the ever growing popularity of devices such as the *Fitbit*).⁸ Finally, we determined that the game would be targeted to both an American and Japanese audience.

⁸ [entertainment software association]

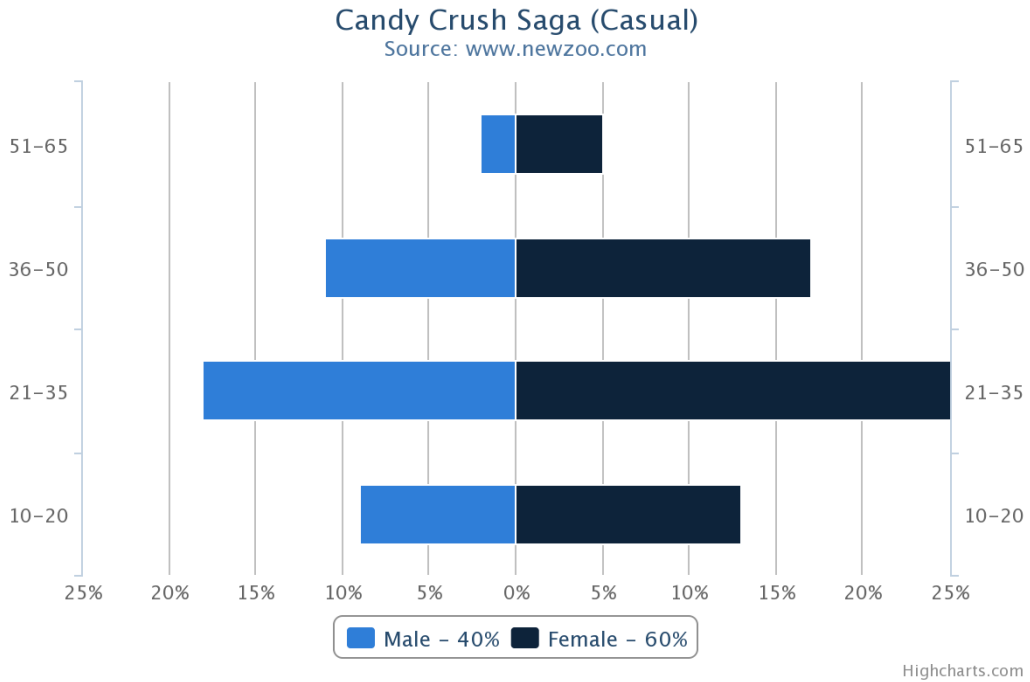


Figure 5 – Age demographics of Candy Crush Saga players⁹

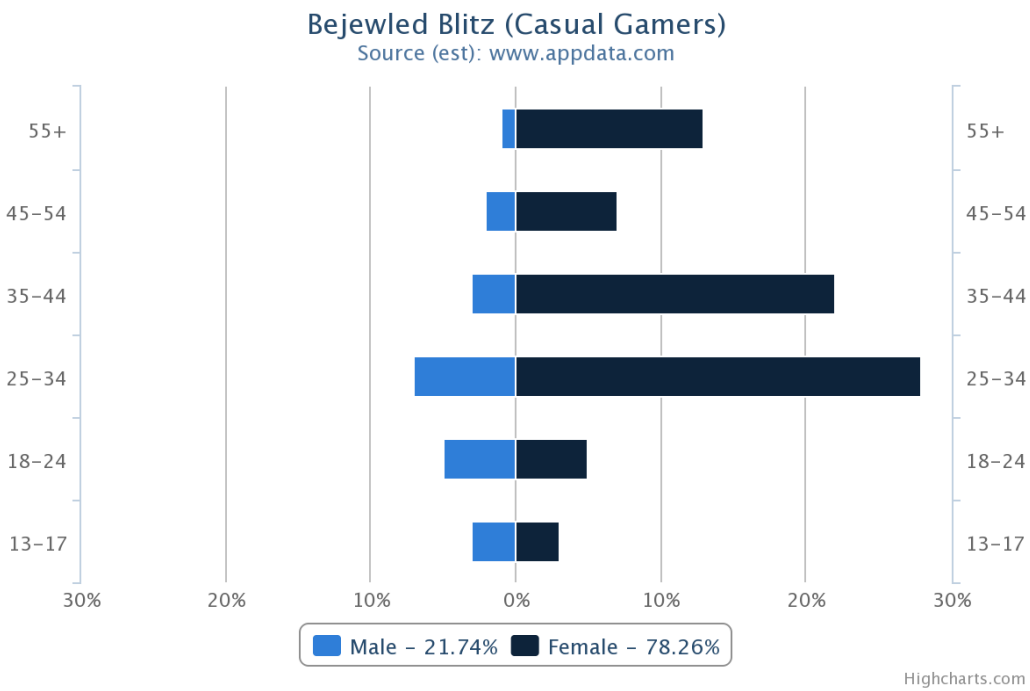


Figure 6 – Age demographics of Bejeweled Blitz players⁸

⁹ [Mason]

3.1.2: American and Japanese Audience

In recent history, mobile games have become exceedingly popular, especially amongst our target audience. This, in conjunction with the omnipresence of iPhones on an international scale, informed our decision to develop a game for iOS.

Based on research we conducted of both the American and Japanese app stores for iOS, we were able to understand the types of applications that were currently popular. In fact, the research confirmed that we were developing the correct game for our target audience. In the American application market, amongst the most popular applications, were those related to fitness; while in Japan, the most popular applications were games, specifically role-playing games.

3.1.3: Violent vs. Non-Violent Gameplay

We considered non-combat oriented gameplay, thinking combat possibly too violent. However, we concluded that due to the age group of our target audience, combined with the vast popularity of combat-oriented games in the Japanese market, that combat-oriented gameplay was not too mature of a concept for the game.

3.1.4: Experiential Goals

The primary experiential goal of this game is to create an engaging gameplay experience that encourages exercise through gameplay. The ultimate goal being that once players have stopped playing the game, that the game will have had a permanent effect on their exercise habits; hopefully encouraging the player to incorporate more exercise into their daily life. The secondary goal of this game is to encourage players to exercise and play games as a group, rather than alone.

3.2: Combat System

One of the two major gameplay elements of our game is the combat system. It is through combat that players test their character's development and further develop their character's traits and abilities. The goal of the level of complexity and depth for the combat system is one deep enough that a devoted player could learn how the different enemies act and develop their character accordingly, while at the same time being simple enough that a casual player would be able to play the game and understand enough to enjoy the gameplay. The following section will explain exactly how the combat system works.

3.2.1: Entities

Both players and enemies fall under the category of an entity. They share the traits that all combat is based on. These traits determine the results of the actions they take and the order of who acts first.

3.2.2: Traits

Both players and enemies are made of the following traits. Though these trait values may change during combat itself, they are set to their initial values after combat has concluded.

- **Level:** A player's level is used to indicate their character's progression through the game. When combat begins, the enemy the player encounters is approximately the same level as the player. The level of the enemy (in conjunction with other variables) is used to determine the other trait values of the enemy.
- **Health:** If the player's health becomes 0, they lose their current fight. If the enemy's health becomes 0, the player has won the fight and receives experience towards growing their character.
- **Strength:** Used to determine the power of physical attacks as well as the player's resistance to physical attacks done by enemies.
- **Magic:** Used to determine the power of magic attacks as well as the player's resistance to magic attacks done by enemies.
- **Speed:** Primarily used to determine the order of entities in combat. Additionally, is used for specific attack damage calculations.

3.2.3: Player

The following sections will explain how the player's character(s) grow, learn abilities, and the impact of real-world exercise on the player's character's growth. In addition to the base traits shared by all entities, players also have an experience trait that is used to express a player's progress towards their character gaining a level.

3.2.4: Character Growth

As the player plays the game, there are two different methods by which their virtual character can grow: through real-world exercise and through gameplay. The behavior we wish to encourage the player to engage in is exercise. For this reason, player growth is designed in such a way that exercising in order to grow becomes increasingly easier as the player grows, while trying to grow their character in a traditional role-playing way (through fighting enemies) becomes increasingly difficult.

3.2.5: Growth through exercise

Each of the player's traits has a method by which it can grow based on the player's exercising habits. For every 5000 steps a player takes in a day, their health trait value is increased by

1. For every three enemies the player encounters (approximately 3000 steps in addition to emerging victorious in the encounters they engage in) the player's strength trait value is increased by 1. At the start of every day, the player is assigned a specific hour of the day, between the hours of 8 am and midnight. For every 1000 steps the player takes during this "magic hour" their magic trait value is increased by 1. For every half of an hour that the player spends running will increase their speed trait value by 1.

3.2.6: Growth through gameplay

In order for the player to increase their trait values through traditional role-playing game means, they must defeat enemies. Specifically, in order for their character to grow a single level, they must 10 enemies times their current level (ex. If they are level 9, they must defeat 90 enemies in order to grow to level 10). Once a player character grows a single level through this method, they are given an opportunity to assign 10 points to any of their traits as they see fit.

3.2.7: Player Actions and Abilities

As the player's trait values increase, they learn new abilities that they can use in combat. The following will explain all of the moves that the player is capable of using. The format of the following explanations will be as follows: an explanation of what the action does denoted by an equation for damage-based actions or by a text based explanation for non-damage based actions accompanied by the icon used to represent the action for the player to select.

The following formulae are consistently used throughout this section. They are the formulae off of which most combat related actions outcomes are actions are based. Additionally, if for some reason any of these formulae were to produce a final value of zero damage to be dealt to the target, one point of damage is dealt instead. Any damage done to an enemy is subtracted from their current health. The term `user.strength` refers to the strength of the user. This is applicable for any entity and any trait of that enemy (such as magic). The words minimum and maximum do not refer to the minimum or maximum value a trait can have, rather they refer to the minimum or maximum the base value of the trait will have.

Physical Damage:

- $$\text{Physical Damage} = \text{user.strength} - \left(\frac{\text{target.strength}}{2}\right)$$

Magic Damage:

- $$\text{Magic Damage} = \text{user.magic} - \left(\frac{\text{target.magic}}{2}\right)$$






Physical Abilities:






The following table (*Table 1 – Physical player actions*) consist of each physical action a player is capable of learning, the strength required to learn the action, and the outcome of that action if it is used during an encounter.



Figure 7 – Physical action icon

Table 1 – Physical player actions

| <u>Action Name</u> | <u>Strength Required</u> | <u>Action Outcome</u> | <u>Action Icon</u> |
|-----------------------|--------------------------|--|---|
| <i>Uppercut</i> | 5 | Deals physical damage to the enemy. |  |
| <i>Charged Strike</i> | 17 | Deals 135% physical damage to the enemy at the cost of 10% of the user's total strength. |  |
| <i>Meditation</i> | 30 | Increases the user's strength by 25% of their total strength and their speed by 25% of their total speed. |  |
| <i>Leg Sweep</i> | 43 | Deals 90% physical damage to the enemy and their strength lowers by 10% of their total strength. |  |
| <i>Turbo Strike</i> | 56 | Deals damage based on the user's speed rather than their strength. $\text{Damage} = \text{user. speed} - \left(\frac{\text{target. speed}}{2} \right)$ |  |

| | | | |
|---------------------------|-----|---|--|
| <i>Heart Strike</i> | 68 | Deals damage equal to 10% of the target's total health. |  |
| <i>Muscle Training</i> | 81 | Increases the user's current strength by 50% of their total strength at the cost of 50% of their total health. |  |
| <i>Stomp</i> | 94 | Deals physical damage to the enemy in addition to damage equal to 16.6% of the enemy's total health. |  |
| <i>Sacrificial Strike</i> | 107 | Lowers the user's health to 1, and then deals damage equal to the health it lost as damage in addition to regular physical damage. |  |
| <i>Overpower</i> | 120 | If the sum of the user's health and strength is greater than the sum of target's strength and health, the user deals damage to the enemy equal to the user's current strength plus its current health; otherwise the target deals damage to the enemy equal to the user equal to the target's current strength plus its current health. |  |









Magic Abilities:



The following table (*Table 2 – Magic player actions*) consist of each magic action a player is capable of learning, the magic required to learn the action, and the outcome of that action if it is used during an encounter.



Figure 8 – Magic action icon

Table 2 – Magic player actions

| <u>Action Name</u> | <u>Magic Required</u> | <u>Action Outcome</u> | <u>Action Icon</u> |
|---------------------|-----------------------|--|---|
| <i>Energy Ball</i> | 5 | Deals magic damage to the enemy. |  |
| <i>Icy Wind</i> | 17 | Deals 90% magic damage to the enemy and their magic lowers by 10% of their total magic. |  |
| <i>Barrier</i> | 30 | Increases the user's current health by 25% of their total health |  |
| <i>Fireball</i> | 43 | Deals 135% magic damage to the enemy at the cost of 10% of the user's total magic. |  |
| <i>Sharpen Mind</i> | 56 | Increases the user's current magic by 35% of their total magic. |  |
| <i>Curse</i> | 68 | Decreases the target's current strength by 25% of their total strength, decreases the target's current magic by 25% of their total magic, and decreases the target's current speed by 25% of their total speed. |  |
| <i>Life Drain</i> | 81 | Deals magic damage to the enemy and increases the user's current health by 75% of the damage dealt to the enemy. |  |
| <i>Decay</i> | 94 | Deals 70% magic damage to the enemy, decreases the target's current strength by 10% of their total strength, decreases the target's current magic by 10% of their total magic, and decreases the target's current speed by 10% of their total speed. |  |

| | | | |
|----------------------|-----|---|---|
| <i>Full Heal</i> | 107 | Increases the user's current health to be equal to their total health. |  |
| <i>Instant Death</i> | 120 | Has a 20% chance to instantly kill the enemy, otherwise nothing happens |  |

Examine:

The final action a player can take is the *Examine* action. The player's "examines" the enemy to discover their trait values at the cost of a turn.



Figure 9 – Examine action icon

3.2.8: Enemies

In order to use the aforementioned abilities, players must encounter an enemy. An enemy will appear after they have taken approximately 1,000 steps. There are 31 types of enemies in the game, broken down into 10 categories. Each of these categories of enemies has a specific way in which the enemy's trait values (health, strength, magic, and speed) are determined. Additionally, each of the 31 enemy types has set actions that it can take.

3.2.9: Possible Growth Rates

In order to maintain control over the balance of the game, different equations were used to determine the growth of an enemy's trait values. An encountered enemy's level has a 50% chance to be the same level as the player and has a 50% chance to be 1 level higher than the player. Based on the level, the trait values of each enemy are determined using different formulae.

Each trait an enemy uses the two different types of equations to have its value determined. First an enemy's trait can have one of five different initial values. The trait can have a value that is either very low, low, medium, high, or very high. After the 3 variables' values are determined by knowing the level of the enemy as well as how that trait's initial values are determined, those values are passed to the second set of equations. The following are the initial value equations and resulting tables (up through level 10).

Very Low: The table shown in *Table 3* shows the offset, minimum, and maximum values a trait with Very Low base values will use during calculation. This is shown for levels 1 – 10.

- Offset = $\frac{\text{enemy.level}}{2}$
- Minimum = enemy .level – 1
- Maximum = enemy.level

Table 3 – Very low trait value table: levels 1-10

| Level | Offset | Minimum | Maximum |
|-------|--------|---------|---------|
| 1 | 0 | 0 | 1 |
| 2 | 1 | 1 | 2 |
| 3 | 1 | 2 | 3 |
| 4 | 2 | 3 | 4 |
| 5 | 2 | 4 | 5 |
| 6 | 3 | 5 | 6 |
| 7 | 3 | 6 | 7 |
| 8 | 4 | 7 | 8 |
| 9 | 4 | 8 | 9 |
| 10 | 5 | 9 | 10 |

Low: The table shown in *Table 4* shows the offset, minimum, and maximum values a trait with Low base values will use during calculation. This is shown for levels 1 – 10.

- Offset = enemy.level
- Minimum = (2 * enemy .level) – 2
- Maximum = (2 * enemy.level)

Table 4 – Low trait value table: levels 1-10

| Level | Offset | Minimum | Maximum |
|-------|--------|---------|---------|
| 1 | 1 | 0 | 2 |
| 2 | 2 | 2 | 4 |
| 3 | 3 | 4 | 6 |
| 4 | 4 | 6 | 8 |

| | | | |
|----|----|----|----|
| 5 | 5 | 8 | 10 |
| 6 | 6 | 10 | 12 |
| 7 | 7 | 12 | 14 |
| 8 | 8 | 14 | 16 |
| 9 | 9 | 16 | 18 |
| 10 | 10 | 18 | 20 |

Medium: The table shown in *Table 5* shows the offset, minimum, and maximum values a trait with Medium base values will use during calculation. This is shown for levels 1 – 10.

- Offset = (2 * enemy.level)
- Minimum = (3 * enemy.level) – 3
- Maximum = (3 * enemy.level)

Table 5 – Medium trait value table: levels 1-10

| Level | Offset | Minimum | Maximum |
|-------|--------|---------|---------|
| 1 | 2 | 0 | 3 |
| 2 | 4 | 3 | 6 |
| 3 | 6 | 6 | 9 |
| 4 | 8 | 9 | 12 |
| 5 | 10 | 12 | 15 |
| 6 | 12 | 15 | 18 |
| 7 | 14 | 18 | 21 |
| 8 | 16 | 21 | 24 |
| 9 | 18 | 24 | 27 |
| 10 | 20 | 27 | 30 |

High: The table shown in *Table 6* shows the offset, minimum, and maximum values a trait with High base values will use during calculation. This is shown for levels 1 – 10.

- Offset = (3 * enemy.level)
- Minimum = (4 * enemy.level) – 4
- Maximum = (4 * enemy.level)

Table 6 – High trait value table: levels 1-10

| Level | Offset | Minimum | Maximum |
|-------|--------|---------|---------|
| 1 | 3 | 0 | 4 |
| 2 | 6 | 4 | 8 |
| 3 | 9 | 8 | 12 |
| 4 | 12 | 12 | 16 |
| 5 | 15 | 16 | 20 |
| 6 | 18 | 20 | 24 |
| 7 | 21 | 24 | 28 |
| 8 | 24 | 28 | 32 |
| 9 | 27 | 32 | 36 |
| 10 | 30 | 36 | 40 |

Very High: The table shown in *Table 7* shows the offset, minimum, and maximum values a trait with Very High base values will use during calculation. This is shown for levels 1 – 10.

- Offset = (4 * enemy.level)
- Minimum = (5 * enemy.level) – 5
- Maximum = (5 * enemy.level)

Table 7 – Very high trait value table: levels 1-10

| Level | Offset | Minimum | Maximum |
|-------|--------|---------|---------|
| 1 | 4 | 0 | 5 |
| 2 | 8 | 5 | 10 |
| 3 | 12 | 10 | 15 |
| 4 | 16 | 15 | 20 |
| 5 | 20 | 20 | 25 |
| 6 | 24 | 25 | 30 |
| 7 | 28 | 30 | 35 |
| 8 | 32 | 35 | 40 |

| | | | |
|----|----|----|----|
| 9 | 36 | 40 | 45 |
| 10 | 40 | 45 | 50 |

The second set of equations is the set of growth rates. As the player’s level grows, the enemies they encounter are also “growing” in that their traits values are changing. There are 3 ways in which an enemy’s trait values can progress as they “level up”. Their trait values can grow logarithmically, linearly, or exponentially. The following are the growth formulas for each of the possible initial values.

For the following equations, the value of “random” is a randomly generated random number between the minimum and maximum values (inclusive). The range of numbers generated is displayed in the following notation: (minimum: maximum). The following tables show the range of values that can be generated for any initial value set and any growth rate up through level 10).

Logarithmic: The table shown in *Table 8* shows the final trait value calculations for all base values growing logarithmically over levels 1 through 10.

- Formula:
$$\text{trait value} = \left(10 * \frac{\ln(\text{entity.level}+1)}{\ln(10)} - (0.0414 * \text{entity.level}) \right) + \text{offset} + \text{random} - \text{entity.level}$$

Table 8 – Final trait value table for logarithmic growth

| Level | Very Low | Low | Medium | High | Very High |
|-------|----------|-------|--------|-------|-----------|
| 1 | 2:3 | 3:5 | 4:7 | 5:9 | 6:11 |
| 2 | 4:5 | 6:8 | 9:12 | 12:16 | 15:20 |
| 3 | 6:7 | 10:12 | 15:18 | 20:24 | 25:30 |
| 4 | 8:9 | 13:15 | 20:23 | 27:31 | 34:39 |
| 5 | 8:9 | 15:17 | 24:27 | 33:37 | 42:47 |
| 6 | 10:11 | 18:20 | 29:32 | 40:44 | 51:56 |
| 7 | 11:12 | 21:23 | 34:37 | 47:51 | 60:65 |
| 8 | 12:13 | 23:25 | 38:41 | 53:57 | 68:73 |
| 9 | 13:14 | 26:28 | 43:46 | 60:64 | 77:82 |
| 10 | 14:15 | 28:30 | 47:50 | 66:70 | 85:90 |

Linear: The table shown in *Table 9* shows the final trait value calculations for all base values growing linearly over levels 1 through 10.

- Formula: trait value = entity.level + offset + random

Table 9 – Final trait value table for linear growth

| Level | Very Low | Low | Medium | High | Very High |
|-------|----------|-------|--------|-------|-----------|
| 1 | 1:2 | 2:4 | 3:6 | 4:8 | 5:10 |
| 2 | 4:5 | 6:8 | 9:12 | 12:16 | 15:20 |
| 3 | 6:7 | 10:12 | 15:18 | 20:24 | 25:30 |
| 4 | 9:10 | 14:16 | 21:24 | 28:32 | 35:40 |
| 5 | 11:12 | 18:20 | 27:30 | 36:40 | 45:50 |
| 6 | 14:15 | 22:24 | 33:36 | 44:48 | 55:60 |
| 7 | 16:17 | 26:28 | 39:42 | 52:56 | 65:70 |
| 8 | 19:20 | 30:32 | 45:48 | 60:64 | 75:80 |
| 9 | 21:22 | 34:36 | 51:54 | 62:72 | 85:90 |
| 10 | 24:25 | 38:40 | 57:60 | 76:80 | 95:100 |

Exponential: The table shown in *Table 10* shows the final trait value calculations for all base values growing exponentially over levels 1 through 10.

- Formula: trait value = $\frac{entity.level^2}{4} + offset + random$

Table 10 – Final trait value table for exponential growth

| Level | Very Low | Low | Medium | High | Very High |
|-------|----------|-------|--------|-------|-----------|
| 1 | 0:1 | 1:3 | 2:5 | 3:7 | 4:9 |
| 2 | 3:4 | 5:7 | 8:11 | 11:15 | 14:19 |
| 3 | 5:6 | 9:11 | 14:17 | 19:23 | 24:29 |
| 4 | 9:10 | 14:16 | 21:24 | 28:32 | 35:40 |
| 5 | 12:13 | 19:21 | 28:31 | 37:41 | 46:51 |
| 6 | 17:18 | 25:27 | 36:39 | 47:51 | 58:63 |
| 7 | 21:22 | 31:33 | 44:47 | 57:61 | 70:75 |

| | | | | | |
|----|-------|-------|-------|-------|---------|
| 8 | 27:28 | 38:40 | 53:56 | 68:72 | 83:88 |
| 9 | 32:33 | 45:47 | 62:65 | 79:83 | 96:101 |
| 10 | 39:40 | 53:55 | 72:75 | 91:95 | 110:115 |

The following is an example of how the trait values for an enemy are determined. The alien subtype (explained below) will be used for this example. We will assume the enemy encountered is level 5.

An alien has the following initial values and growth rates for each of their traits. For a medium value health trait that grows logarithmically, the health of the alien would be between 24 and 27. For a low strength value that grows logarithmically, the strength of the alien would be between 15 and 17. For a very high magic trait that grows linearly, the magic of the alien would be between 45 and 50. And finally for an alien with a low speed trait that grows exponentially, the speed of the alien would be between 19 and 21.

3.2.10: Actions

Similarly to the player, enemies also use actions during combat. The table shown in figure *Table 11* shows all actions an enemy can use during combat. The format of the table is as follows: the first column denotes the name of the action and the second column explains the outcome of the action were it to be used during an encounter.

Formulae: The following formulae are consistently used throughout the following section. They are the formulae off of which most combat-related actions are actions are based. Additionally, if for some reason any of these formulae were to produce a final value of zero damage to be dealt to the target, one point of damage is dealt instead. Any damage done to a target is subtracted from their current health.

Physical Damage:

- Physical Damage = $\text{user.strength} - \left(\frac{\text{target.strength}}{2}\right)$

Magic Damage:

- Magic Damage = $\text{user.magic} - \left(\frac{\text{target.magic}}{2}\right)$

Table 11 – Enemy actions

| <u>Action Name</u> | <u>Action Outcome</u> |
|--------------------|---|
| <i>Punch</i> | Deals physical damage to the target. |
| <i>Rapid Fire</i> | With a 25% success rate the damage dealt to the |

| | |
|---------------------------|--|
| | target is equal to 85% of the user's current speed. This process is performed 10 times. |
| <i>Cleave</i> | Deals 125% physical damage to the target. Decreases the user's strength by 10% of its total strength. |
| <i>Reckless Strike</i> | With a 50% success rate the damage dealt to the target is equal to 200% physical damage. |
| <i>Bite</i> | Deals physical damage to the target. |
| <i>Eat Dirt</i> | Increases the user's health by 15% of their total health. |
| <i>Eight Pronged Stab</i> | With a 12.5% success rate the damage dealt to the target is equal to 85% physical damage. This process is performed 8 times. |
| <i>Horn Smash</i> | Deals 125% physical damage to the target and deals 25% of that much damage to itself. |
| <i>Spray Acid</i> | Deals damage to the target equal to 25% of its current health. |
| <i>Ætherial Fangs</i> | Deals magic damage to the target. |
| <i>Horrify</i> | Deals damage to the target equal to 15% of its current health. The target's current strength and magic are reduced by 15%. |
| <i>Meditate</i> | Increases the user's current health, strength, and magic by 50% of its current health. |
| <i>Mind Invasion</i> | Deals magic damage to the target. The target then deals 50% physical damage to itself. |
| <i>Ætherial Darts</i> | Deals magic damage to the target. |
| <i>Burrow</i> | Increases the user's current health by 15% of its total health. |
| <i>Focus Energy</i> | Decreases the user's health by 15% and increases the user's current strength by 15%. |
| <i>Rushdown</i> | Deals 125% physical damage to the target. Decreases the user's current health by 25%. |
| <i>Tribite</i> | With a 33% success rate, the enemy will deal 33% physical damage to itself. With a 66% success rate, the enemy will deal 33% physical damage to the target. This process is performed 3 times. |
| <i>Unforeseen Attack</i> | Deals 150% physical damage to the target. |

| | |
|-------------------------------|---|
| | Decreases the user's current health by 50%. |
| <i>Defend Core</i> | Increases the user's current health by 25% of its total health. |
| <i>Enrage</i> | Increases the user's current strength and magic by 25%. |
| <i>Crush</i> | Deals physical damage to the target. |
| <i>Head Bash</i> | Deals physical damage to the target. |
| <i>Limb Swing</i> | Deals 115% physical damage to the target. Decreases the user's current health by 15%. |
| <i>Glancing Wing</i> | With a 33% success rate the damage dealt to the enemy is equal to 85% physical damage. This process is performed 2 times. |
| <i>Club</i> | With a 50% success rate, 150% physical damage is dealt to the target. |
| <i>Scalding Conflagration</i> | Deals magic damage to the target. |
| <i>Suffocating Current</i> | Deals magic damage to the target. |
| <i>Slicing Gale</i> | Deals magic damage to the target. |
| <i>Crushing Gaea</i> | Deals magic damage to the target. |
| <i>Drain</i> | Deals magic damage to the target. The user's current health is increased by the damage dealt. |
| <i>Confuse</i> | Deals magic damage to the target. |
| <i>Envelop</i> | Deals 50% physical damage to the target. Decreases the target's current strength and magic by 25%. |
| <i>Blight</i> | Deals 80% magic damage to the target. Decreases the target's current strength and magic by 10%. |
| <i>Smite</i> | Deals magic damage to the target. |
| <i>Engulf</i> | Deals magic damage to the target. |
| <i>Twilight Strike</i> | Deals 50% magic damage to the target. Then deals 50% damage based on speed |
| <i>Eradicating Light</i> | The damage dealt to the target is equal to 80% of 50% magic damage plus 50% physical damage. Decreases the user's current health by 20%. |
| <i>Tail Swing</i> | Deals 80% physical damage to the target. Decreases the target's current strength by 20%. |
| <i>Constrict</i> | Deals 80% physical damage to the target. Decreases the target's current magic by 20%. |

| | |
|-------------------------|---|
| <i>Breath of Ice</i> | Deals 125% magic damage to the target. Decreases the user's current magic by 25%. Decreases the target's current magic by 10%. |
| <i>Crush Under Foot</i> | Deals 90% physical damage to the target. Decreases the target's current strength by 10%. |
| <i>Breath of Fire</i> | Deals 125% magic damage to the target. Decreases the user's current magic by 10%. Decreases the target's current strength by 10%. |

3.2.11: Types

As previously stated, there are 10 different types of enemies. The following section will explain the trait growth rates of each type of enemy, as well as different abilities each of the enemy subtypes can use. Additionally, design inspirations for each of the enemies will be explained.

3.2.11.1: Humanoid

Intended to be the most balanced enemy a player could encounter, each of the humanoid type enemy's traits are the middlemost values an enemy's traits could be. The human enemy type was designed to be the most balanced enemy a player could encounter because the players themselves are human. Assuming the player exercises at an average rate, their encounter with a human was designed to always create an approximately 50% matchup.

All enemies of the humanoid type have a health trait with a medium base value that grows linearly, a strength trait with a medium base value that grows linearly, a magic trait with a medium base value that grows linearly, and a speed trait with a medium base value that grows linearly.

There are three different subtypes of humanoid type enemies a player can encounter.

Archer (*Figure 10*): The archer subtype has a 50% chance to use the action *Punch* and a 50% chance to use the action *Rapid Fire*.

Knight (*Figure 11*): The knight subtype has a 50% chance to use the action *Punch* and a 50% chance to use the action *Cleave*.

Ninja (*Figure 12*): The ninja subtype has a 50% chance to use the action *Punch* and a 50% chance to use the action *Reckless Strike*.



Figure 10 – Archer



Figure 11 – Knight



Figure 12 – Ninja

3.2.11.2: Insect

Designed to be weak, the insect type enemies not only have particularly low trait values, but 2/3 of them are capable of using actions that either do nothing to the opponent or hurt the insect in the process. This was done in an effort to create a matchup that favored almost all players, regardless of how they assign their trait values.

All enemies of the insect type have a health trait with a very low base value that grows linearly, a strength trait with a low base value that grows linearly, a magic trait with a low base value that grows exponentially, and a speed trait with a very high base value that grows logarithmically. There are three different subtypes of insect type enemies a player can encounter.

Wurm (*Figure 13*): The wurm subtype has a 50% chance to use the action *Bite* and a 50% chance to use the action *Eat Dirt*.

Spider (*Figure 14*): The spider subtype has a 50% chance to use the action *Bite* and a 50% chance to use the action *Eight Pronged Stab*.

Beetle (*Figure 15*): The beetle subtype has a 50% chance to use the action *Bite* and a 50% chance to use the action *Horn Smash*.

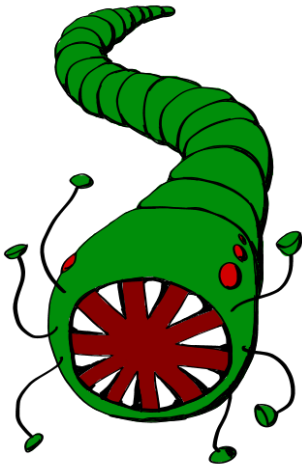


Figure 13 – Wurm

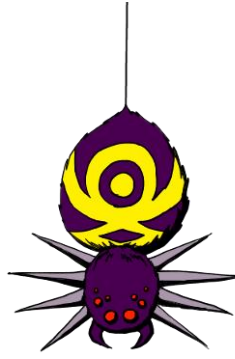


Figure 14 - Spider

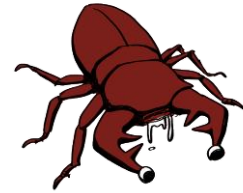


Figure 15 - Beetle

3.2.11.3: Alien

Designed based on extraterrestrial beings from the Cthulhu mythos, all three of the subtypes of the alien type enemies are capable of using the action *Horrify*. *Horrify* is inspired by the major theme of the Cthulhu mythos, encountering things so mind-bogglingly incomprehensible that the viewer is driven to madness.

All enemies of the alien type have a health trait with a medium base value that grows logarithmically, a strength trait with a low base value that grows logarithmically, a magic trait with a very high base value that grows linearly, and a speed trait with a low base value that grows exponentially.

There are three different subtypes of alien type enemies a player can encounter.

Oorn (Figure 16): The oorn subtype has a 33% chance to use the action *Horrify*, has a 33% chance to use *Ætherial Fangs*, and a 33% chance to use the action *Spray Acid*.

Mi-go (Figure 17): The mi-go subtype has a 33% chance to use the action *Horrify*, has a 33% chance to use *Meditate*, and a 33% chance to use the action *Mind Invasion*.

Yugg (Figure 18): The yugg subtype has a 33% chance to use the action *Horrify*, has a 33% chance to use *Ætherial Darts*, and a 33% chance to use the action *Burrow*.



Figure 16 – Oorn

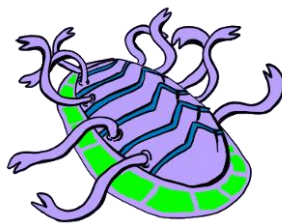


Figure 17 – Mi-go



Figure 18 – Yugg

3.2.11.4: Beast

Designed based upon mythological creatures, the enemies of the beast subtype have particularly high strength and health trait values. Much like the mythological creatures in literature, these enemies are designed to be particularly difficult to defeat, especially for a player who has assigned their trait values in preference of strength, much like the many heroes of mythology.

All enemies of the beast type have a health trait with a very high base value that grows exponentially, a strength trait with a high base value that grows exponentially, a magic trait with a very low base value that grows logarithmically, and a speed trait with a very low base value that grows exponentially.

There are three different subtypes of beast type enemies a player can encounter.

Minotaur (*Figure 19*): The minotaur subtype has a 50% chance to use the action *Focus Energy* and a 50% chance to use the action *Rushdown*.

Cerberus (*Figure 20*): The cerberus subtype has a 50% chance to use the action *Focus Energy* and a 50% chance to use the action *Tribite*.

Griffin (*Figure 21*): The griffin subtype has a 50% chance to use the action *Focus Energy* and a 50% chance to use the action *Unforeseen Attack*.



Figure 19 - Minotaur Figure 20 - Cerberus Figure 21 - Griffin

3.2.11.5: Construct

The construct enemy type is designed based upon the Qliphoth of hermetic qabalah. The constructs are designed to be especially difficult to defeat however due to the randomness of their actions, they do not necessarily act maliciously towards the player, they merely get in the players way.

All enemies of the construct type have a health trait with a medium base value that grows linearly, a strength trait with a high base value that grows linearly, a magic trait with a high base value that grows logarithmically, and a speed trait with a very low base value that grows exponentially.

There are three different subtypes of construct type enemies a player can encounter.

Barrier (Figure 22): The barrier subtype has a 10% chance to use the action *Horrify*, has a 10% chance to use *Enrage*, has a 10% chance to use the action *Crush*, and a 70% chance to use the action *Defend Core*.

Shell (Figure 23): The shell subtype has a 70% chance to use the action *Horrify*, has a 10% chance to use *Enrage*, has a 10% chance to use the action *Crush*, and a 10% chance to use the action *Defend Core*.

Husk (*Figure 24*): The husk subtype has a 10% chance to use the action *Horrrify*, has a 70% chance to use *Enrage*, has a 10% chance to use the action *Crush*, and a 10% chance to use the action *Defend Core*.



Figure 22 – Barrier

Figure 23 – Shell

Figure 24 – Husk

3.2.11.6: Undead

The enemies of the undead type are designed based upon the beings of Mesopotamian mythology. The undead enemies are designed to be weak, each of which never capable of using an attack that does higher than average damage without a downside to the user.

All enemies of the beast type have a health trait with a high base value that grows logarithmically, a strength trait with a medium base value that grows exponentially, a magic trait with a low base value that grows logarithmically, and a speed trait with a very low base value that grows linearly.

There are three different subtypes of undead type enemies a player can encounter.

Zombie (*Figure 25*): The zombie subtype has a 50% chance to use the action *Head Bash* and a 50% chance to use the action *Rushdown*.

Chimera (*Figure 26*): The chimera subtype has a 50% chance to use the action *Head Bash* and a 50% chance to use the action *Glancing Wing*.

Undead Soldier (*Figure 27*): The undead soldier subtype has a 50% chance to use the action *Head Bash* and a 50% chance to use the action *Club*.



Figure 25 – Zombie Figure 26 – Chimera Figure 27 – Undead Soldier

3.2.11.7: Elemental

The enemies of the elemental type are designed to punish players for not assigning trait values to all of their traits or exercising enough. The majority of enemies in *Errant* use physical actions over magic actions. By creating an enemy type that uses only magic actions that also has the 2nd highest possible magic value, it forces players to exercise more during the magic hours that the game randomly declares.

All enemies of the elemental type have a health trait with a low base value that grows linearly, a strength trait with a very low base value that grows exponentially, a magic trait with a very high base value that grows linearly, and a speed trait with a high base value that grows logarithmically.

There are four different subtypes of elemental type enemies a player can encounter.

Fire Elemental (*Figure 28*): The fire elemental subtype always uses the action *Scalding Conflagration*.

Water Elemental (*Figure 29*): The water elemental subtype always uses the action *Suffocating Current*.

Wind Elemental (*Figure 30*): The wind elemental subtype always uses the action *Slicing Gale*.

Earth Elemental (*Figure 31*): The earth elemental subtype always uses the action *Crushing Gaea*.

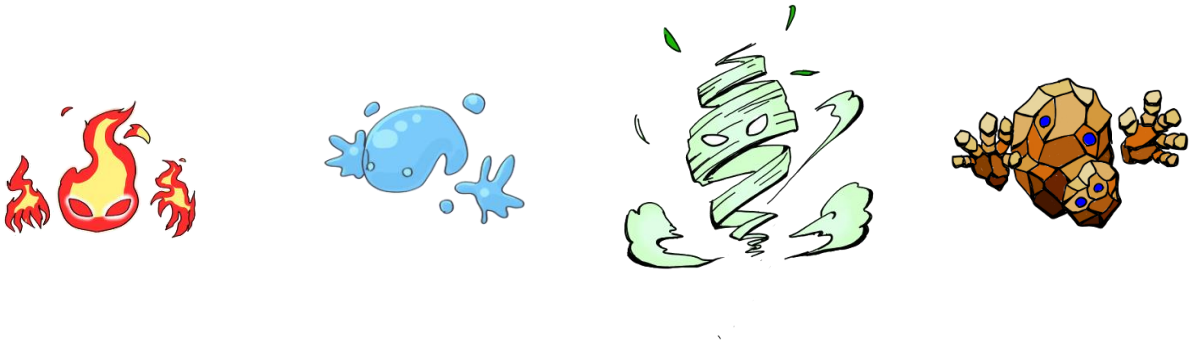


Figure 28 – Fire Figure 29 – Water Figure 30 – Wind Figure 31 – Earth

3.2.11.8: Slime

Designed to be the weakest enemy type in *Errant* there is no point in the game that a player encountering an enemy of the slime type that they cannot defeat. The design inspirations of the subtypes of the slime type come from the Cthulhu mythos as well as numerous television and movie interpretations of slime.

All enemies of the slime type have a health trait with a very low base value that grows logarithmically, a strength trait with a very low base value that grows logarithmically, a magic trait with a very low base value that grows logarithmically, and a speed trait with a very low base value that grows logarithmically.

There are three different subtypes of slime type enemies a player can encounter.

Shoggoth (Figure 32): The shoggoth subtype has a 50% chance to use the action *Drain* and a 50% chance to use the action *Horriify*.

Brain Slug (Figure 33): The brain slug subtype has a 50% chance to use the action *Drain* and a 50% chance to use the action *Confuse*

Ooze (Figure 34): The ooze subtype has a 50% chance to use the action *Drain* and a 50% chance to use the action *Envelop*.

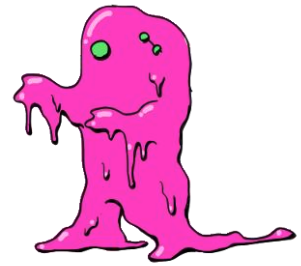
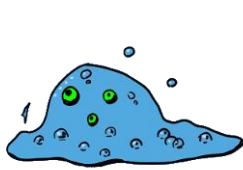


Figure 32 – Shoggoth Figure 33 – Brain Slug Figure 34 – Ooze

3.2.11.9: Demon

Designed to be the most fearsome enemies in *Errant*, the enemies of the demon type attack the player by lowering their trait values while doing less damage to the player overall.

All enemies of the demon type have a health trait with a medium base value that grows logarithmically, a strength trait with a very low base value that grows logarithmically, a magic trait with a very high base value that grows exponentially, and a speed trait with a high base value that grows linearly.

There are three different subtypes of demon type enemies a player can encounter.

Djinn (*Figure 35*): The djinn subtype has a 50% chance to use the action *Blight* and a 50% chance to use the action *Smite*.

Vampire (*Figure 36*): The vampire subtype has a 50% chance to use the action *Drain* and a 50% chance to use the action *Engulf*.

Angel of Death (*Figure 37*): The ooze subtype has a 50% chance to use the action *Blight* and a 50% chance to use the action *Twilight Strike*.



Figure 35 – Djinn



Figure 36 – Vampire



Figure 37 – Angel of Death

3.2.11.10: Dragon

The enemies of the dragon type are designed based upon the many interpretations of dragons from numerous mythological backgrounds. The enemies of the dragon type are capable of using both physical and magic actions well and are among the most powerful enemies in *Errant*.

All enemies of the slime type have a health trait with a low base value that grows exponentially, a strength trait with a high base value that grows linearly, a magic trait with a high base value that grows linearly, and a speed trait with a medium base value that grows logarithmically.

There are three different subtypes of dragon type enemies a player can encounter.

Wurm (*Figure 38*): The wurm subtype has a 50% chance to use the action *Eradicating Light* and a 50% chance to use the action *Tail Swing*.

Sea Serpent (*Figure 39*): The sea serpent subtype has a 50% chance to use the action *Constrict* and a 50% chance to use the action *Breath of Ice*.

Dragon (*Figure 40*): The dragon subtype has a 50% chance to use the action *Crush Under Foot* and a 50% chance to use the action *Breath of Fire*.

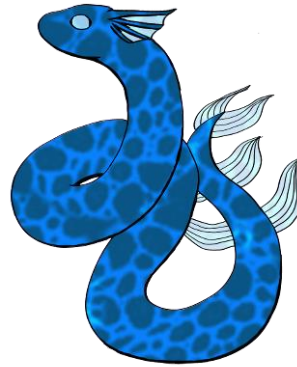


Figure 38 – Wyrn Figure 39 – Sea Serpent Figure 40 – Dragon

3.3: Testing Procedure

In order to balance the combat system of the game two different methods were used. Before using human play testers, an automated combat simulation system was created. Then, once human play testing began: the results of all encounters were recorded and analyzed.

The desired balance for the game was to create a situation in which the player would have approximately a 60% win percentage against a randomly generated enemy, given the player only assigned values to their traits by leveling up their character, not through exercise. The reason for this generally low win percentage (in comparison with traditional role-playing games) was to encourage the player to exercise at an above average rate in order to improve their character and their win percentage against enemies.

In order to begin balancing *Errant*, multiple role-playing game systems were broken down analyzed. Specifically, the games *Diablo 1*¹⁰ and *Final Fantasy 9*¹¹ were the games examined for the purpose of combat system balancing.

3.3.1: Computer Automated Testing

The automated testing system ran simulations of randomly generated enemies fighting against partially-randomly generated players. The simulations generated 11 different players, each of which assigned points to their trait values in a different way. (For example: one player would assign more points towards strength while another would assign more points towards magic). These randomly generated players would then be put into combat with a randomly generated enemy (following the trait value assignment rules stated previously) of approximately equal level (the level

¹⁰ [Faria]

¹¹ [SQUARE ENIX]

of the enemy would be either 1 less than, equal to, or 1 greater than the level of the player). This simulation would then be run for all 31 types of enemies for all generated players level 1 through 10. This resulted in 3410 combat simulations. This process was repeated numerous times, with many alterations to the mathematics behind the combat system, until the desired balance of the game was approximately struck.

The only major difference between the combat simulations and a real person playing the game was the action selection by the simulated player. For all simulations, the player used either the action *Uppercut* or *Energy Ball*, depending on if their strength or magic was higher respectively. These moves are the most balanced moves a player can use and made for the most optimal testing conditions.

3.3.2: Player Based Testing

Player based testing was used in a manner similar to that of automated testing. By examining the outcomes and actions taken during each combat the player's engaged in, further adjustments to the game were made in order to achieve the desired game balance. Additionally, the data obtained from play testers was also used to find bugs in the game.

3.4: Future design areas and expansion possibilities

There are numerous areas of design that were not explored in the design and development of this game. The following are areas of design that should be explored further if there is any future development:

- Non-combat abilities (abilities a player uses while walking)
- Ability for players to place beacons in a sequence allowing them to tell a story.
- Additional expansions to the combat system (items, equipment, class-system, etc.)
 - Friend system and appropriate benefits for multiplayer gameplay
 - Use of HealthKit (an iOS8 feature released near the end of development)

4: Gameplay Experience Description

This section will explain the gameplay experiences the players will encounter when playing *Errant*. The first section will explain a user’s first gameplay experience while the second section will explain subsequent gameplay experiences.

4.1: First gameplay experience

When first running the game, the player will come upon the screen shown in *Figure 41*. From here, the player will select the “New Game” button. That will take the player to the screen shown in *Figure 42*. It is from here that the player can select from one of the two following options. They are able to associate their account with a Twitter account already connected to their phone or they can create an account for our game exclusively.



Figure 41 – Title screen.



Figure 42 – Account creation screen, not logged in

If they wish to associate their account with Twitter they will select the “Login with Twitter” button, bringing them to the screen shown in *Figure 1*. On this screen, the player can find the list of all Twitter accounts associated with their phone. By selecting the “Done” button they can continue to the screen, shown in *Figure 43*. On this screen, the player is given an opportunity to re-select the method by which they select their account. Once they are satisfied with their selection, they can press the “Create Character” button to be brought to the screen shown in *Figure 44*.

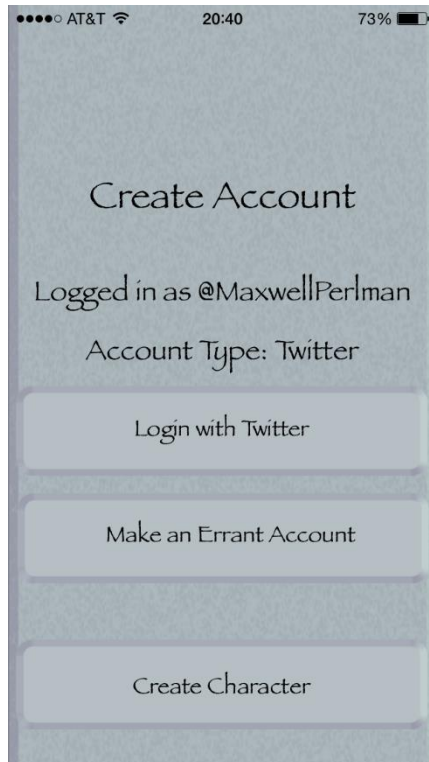


Figure 43 – Account creation screen, logged in with Twitter

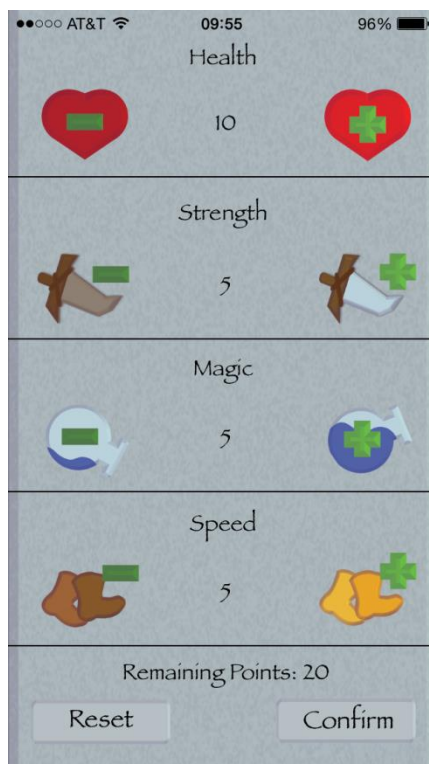


Figure 44 – Character creation screen

If they wish to create an account for *Errant*, they will select the “Make an *Errant* Account” button, bringing them to the screen shown in *Figure 45*. On this screen, the player is given an

opportunity to enter their desired username. If the username already exists on the server, they are informed and asked to enter a different username. If the username is not already in use, the player is then forwarded to the screen shown in *Figure 46*. On this screen, the player is given an opportunity to re-select the method by which they select their account. Once they are satisfied with their selection, they can press the “Create Character” button to be brought to the screen shown in *Figure 44*.

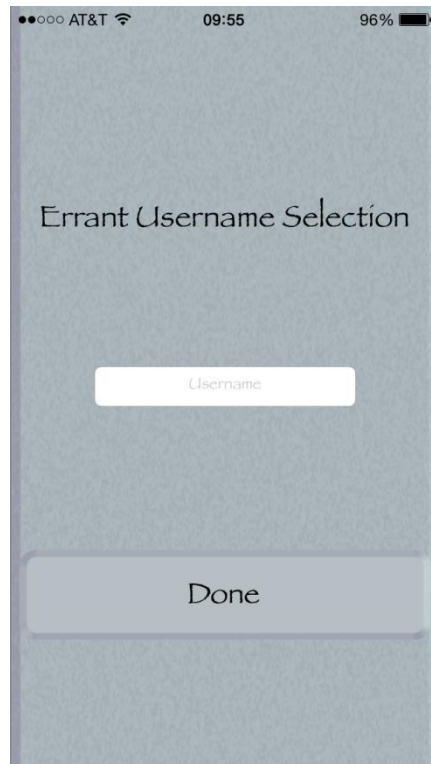


Figure 45 – Username selection screen

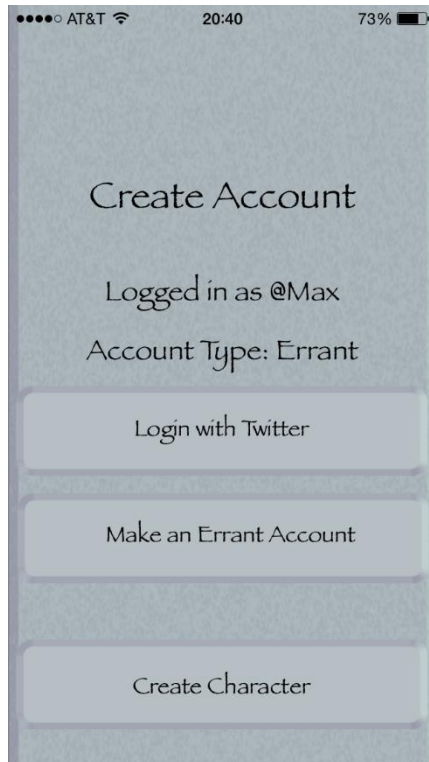


Figure 46 – Account creation screen, logged in with Errant account

On the screen shown in *Figure 44*, the player is given an opportunity to assign up to twenty points to the trait of their choosing. The player begins with a base value of 10 health points and 5 points for each strength, magic, and speed. Once the player is satisfied with the way they have assigned their trait values, they can press the continue button to move to the screen represented by *Figure 2*. Any points left unassigned are stored until the player levels up their character (at which point they are given an additional 20 points to assign). From this point forward, the gameplay experience of the first time player will be the same as the experience of a player playing the game not for the first time. The explanation of the screen shown in *Figure 2* and all subsequent screens will be explained below.

4.2: Subsequent gameplay experiences

When starting the game, the player will select the “Continue” button found on the screen shown in *Figure 41*. This will immediately bring the player to the screen shown in *Figure 2*.

The screen shown in *Figure 2* is the main screen of the game. Here the player will see a map view with a pin representing each other *Errant* player. On this screen the player can find the following information: the number of steps they have taken today, their current mode of transportation (walking, running, and automobile), the magic hour for the day, and finally the number of steps they have taken during the magic hour for the day. There is a “Find Me” button in the top right corner of the screen. If clicked, this button will shift the view of the map to focus on the player’s current position. Additionally, there is a button in the top left corner of the screen

(represented by the player’s name). If clicked, that button will shift the player to the screen shown in *Figure 47*.

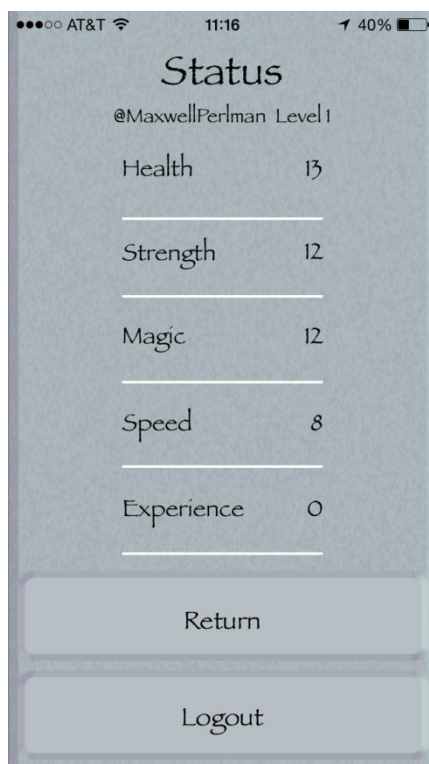


Figure 47 – Status screen

The screen shown in *Figure 47* is the player’s status screen. Here the player can find the value of each of their traits (health, strength, magic, and speed). Additionally, the can find their current progress towards leveling their character up (the experience bar) as well their progress towards raising their trait values through exercise (represented by the bar’s below each of the trait values). On this screen the player will also find both a “Return” button and “Logout” button. If the player were to select the “Return” button, it would send them back to the screen shown in *Figure 2*. If they were to select the “Logout” button, it would return them to the screen shown in *Figure 41*.

The majority of gameplay takes place while the application is running in the background of the user’s phone. After taking approximately 1,000 steps, the player will receive a phone notification informing them that they have encountered an enemy. Once opening the notification, the player is transition to the screen shown in *Figure 48*.



Figure 48 – Battle screen

The screen shown in *Figure 48* is the encounter screen. Here, the player will find an image representative of the type of enemy they encountered, the player's trait information for the battle (their health, strength, magic and speed) as well as their current trait information (showing any changes to the player's traits that occurred during battle), three buttons found on the left side of the screen and a status bar on the top of the screen. There are three different possible background images that can appear for the encounter screen: the ocean (*Figure 49*), the city (*Figure 50*), and the mountains (*Figure 51*). The background is selected randomly each time an encounter begins.

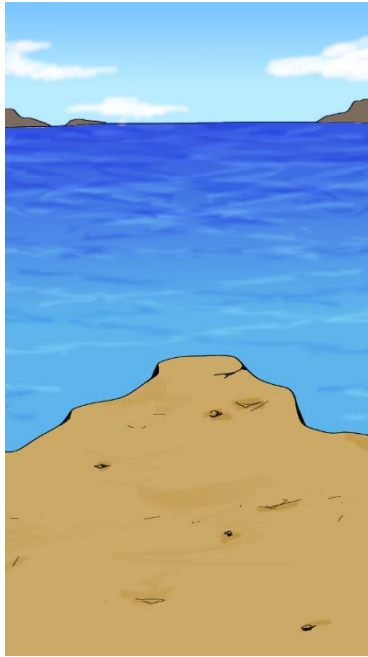


Figure 49 – The ocean background



Figure 50 – The city background

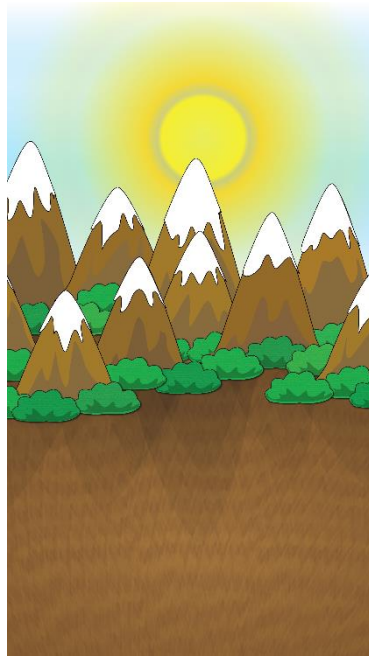


Figure 51 – The mountain background

The status bar is used to show the results of actions the player has taken as well as to inform the player of whether or not it is their turn. The three buttons on the left side are used to perform actions during combat. The first two icons are used to represent the types of actions a player can take, physical and magical respectively. If selected, these buttons will trigger the opening of a menu on the right side of the screen. These new menu's show all of the actions (represented with icons) of the selected type (physical or magical) that a player can use. A player can select one of these icons to use the associated action. If the physical action button is selected, an image similar to the one shown in *Figure 52* will be shown. If the magic action button is selected, an image similar to the one shown in *Figure 4* will be shown. The examine button on the left hand menu is used to perform the action *Examine*, if so an image similar to the one shown in *Figure 53* will be shown..

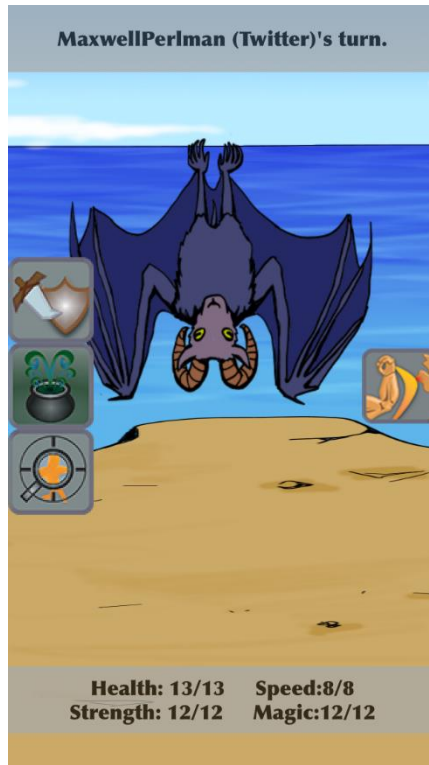


Figure 52 – Battle screen with physical action menu (right) open



Figure 53 – Battle screen, “Examine” used by player

Once the encounter screen is reached, the player will engage in turn based combat with randomly generated enemy until one of them is defeated (by their current health becoming 0). From here, the player will be transferred to one of two screens: if the player was victorious, they will be

transitioned to the screen shown in *Figure 54*. If the player is defeated, they will be transitioned to the screen shown in *Figure 55*.

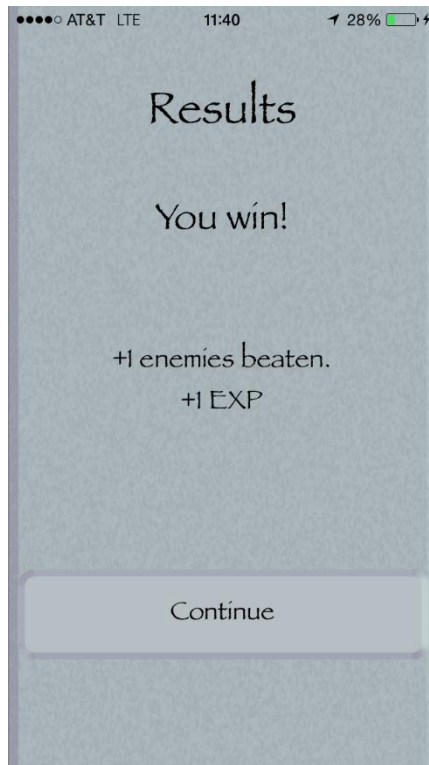


Figure 54 – Results screen when winning a battle

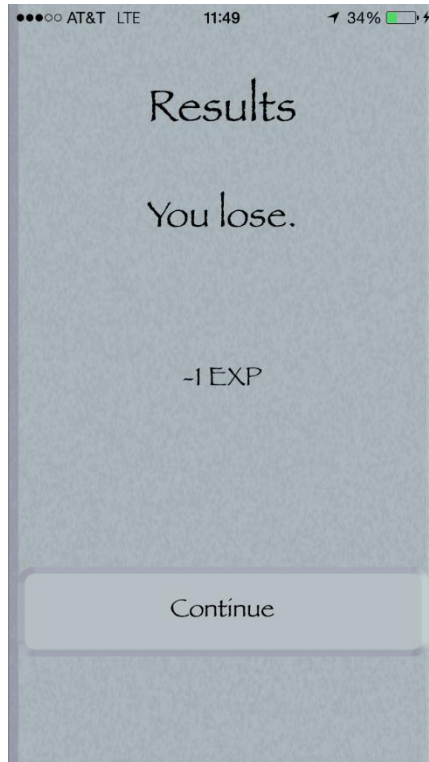


Figure 55 – Results screen when losing a battle

The screens shown in *Figure 54* and *Figure 55* show the user the consequences of their victory. This screen consists of a single button by which the player can transition back to the screen shown in *Figure 2*. Additionally, if the player's level was raised by emerging victorious from their encounter, they will be transferred to the screen shown in *Figure 56*. From here, they can select the continue button to transition to the screen shown in *Figure 57*, where they are able to assign 10 new points to their trait values.

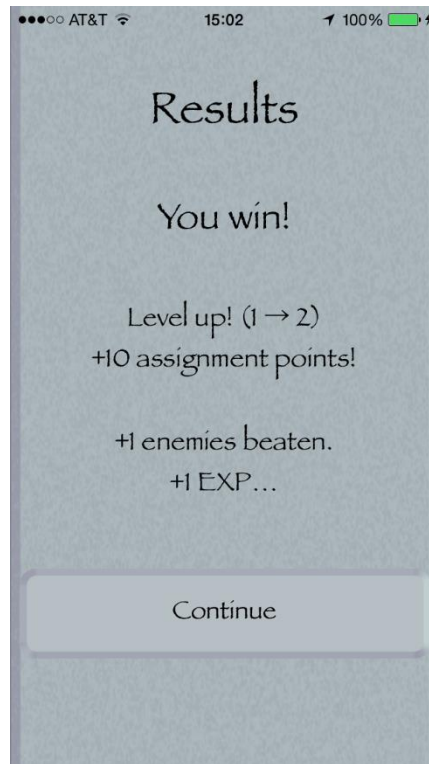


Figure 56 – Results screen when winning a battle and leveling up

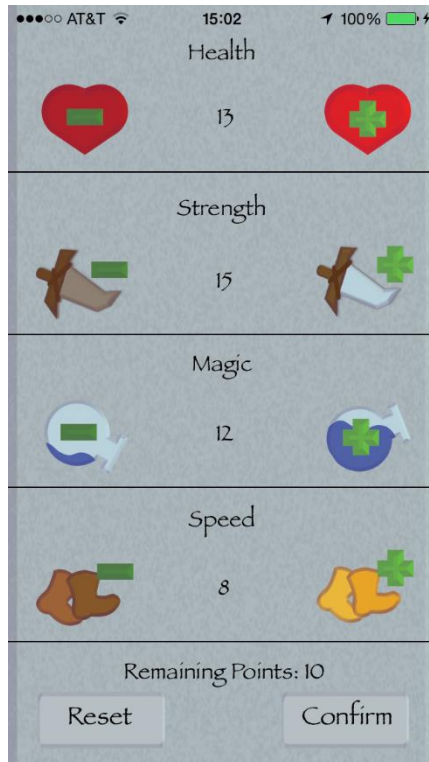


Figure 57 – Status assignment screen shown after leveling up

There is another method by which the player can engage in combat with not only a randomly generated enemy, but with other players as well. If multiple players happen upon a beacon, they chose to join the beacon by pressing the join button found on the left side of the screen shown in *Figure 2*. By pressing the join button, the player will be transitioned to the screen shown in *Figure 58*. The screen shown in *Figure 58* is a lobby in which players can wait for the encounter to begin. Each beacon has a set number of people necessary for the encounter to begin. Players are able to view the names of all players at the beacon until the correct number of players has joined. The screen shown in *Figure 3* shows a single player waiting for another player to join the encounter at the beacon. Once the proper number of people is present, an encounter will begin and the player will be transferred to the screen shown in *Figure 48*.

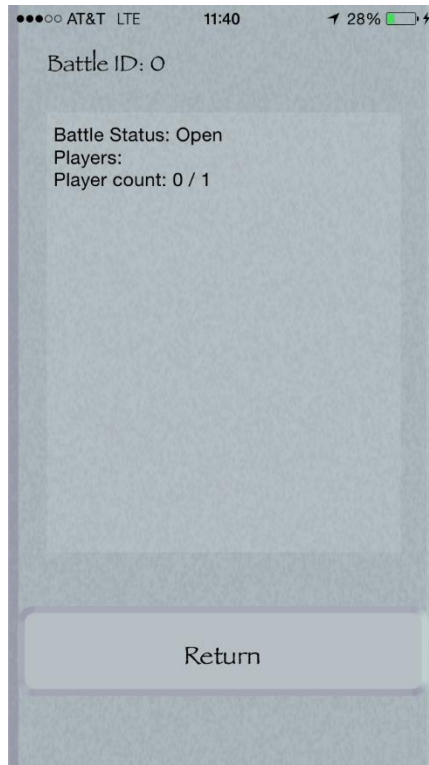


Figure 58 – Multiplayer setup screen

5: Reference Materials and Tools

- 135yshr. *swift* で *twitter* アプリを作ったときにハマったこと. 4 June 2014. 8 November 2014. <<http://qiita.com/135yshr/items/54a999b334f50f1709d5>>.
- An *iOS 6 iPhone UIPickerView Example*. 5 October 2012. 8 November 2014. <http://www.techotopia.com/index.php/An_iOS_6_iPhone_UIPickerView_Example>.
- Coleman, Brain. *TUTORIAL: POST TO WEB SERVER API IN SWIFT USING NSURLConnection*. 2014. 8 November 2014. <<http://www.brianjcoleman.com/tutorial-post-to-web-server-api-in-swift-using-nsurlconnection/>>.
- D&D Official Homepage*. 2014. Wizards of the Coast. 8 November 2014. <<http://dnd.wizards.com/>>.
- entertainment software association. "Essential Facts About the Computer and Video Game Industry." 2013. 9 November 2014. <https://www.theesa.com/facts/pdfs/ESA_EF_2013.pdf>.
- exilias. *Accounts.framework* を使って *Facebook* のアクセストークンとか、メールアドレスを取得する. 17 March 2014. 8 November 2014. <<http://qiita.com/exilias/items/b574e983c18b9360fe34>>.
- Faria, Pedro. "Jarulf's Guide to Diablo and Hellfire." April 2010. *The Lurker Lounge*. Document. 8 November 2014.
- GeoGebra*. 2014. 8 November 2014. <<http://web.geogebra.org/>>.
- Google Maps SDK for iOS*. 2014. Google. 8 November 2014. <<https://developers.google.com/maps/documentation/ios/>>.
- Gordeev, Andrey. *How to reference the current ViewController from a Sprite Kit Scene*. 17 February 2014. 8 November 2014. <<http://stackoverflow.com/questions/21827783/how-to-reference-the-current-viewcontroller-from-a-sprite-kit-scene/21829209#21829209>>.
- griffin-stewie. *M7 と少しだけ戯れてみた*. 22 September 2013. 8 November 2014. <<http://griffin-stewie.hatenablog.com/entry/2013/09/22/130002>>.
- HeshamMegid. *HeshamMegid/HMSideMenu*. 8 May 2013. 8 November 2014. <<https://github.com/HeshamMegid/HMSideMenu>>.
- Ingress*. 2014. Niantic Labs. 8 November 2014. <<https://www.ingress.com/>>.
- iOS Developer Library*. 2014. Apple. 8 November 2014. <<https://developer.apple.com/library/ios/navigation/>>.
- ldindu. *How to implement CMStepCounter CoreMotion - M7 Chip*. 5 January 2014. 8 November 2014. <<http://stackoverflow.com/questions/20936207/how-to-implement-cmstepcounter-coremotion-m7-chip/20938104#20938104>>.
- Mason, Mike. *Demographic Breakdown of Casual, Mid-Core and Hard-Core Mobile Gamers*. 19 December 2013. 9 November 2014. <<http://developers.magmic.com/demographic-breakdown-casual-mid-core-hard-core-mobile-gamers/>>.

- mochizukikotaro. *[Swift]POST* で API を叩く方法。(そして JSON を取得する。)。 8 June 2014. 8 November 2014. <<http://qiita.com/mochizukikotaro/items/e2da2d3186ec24e291a6>>.
- Narrthine, Shiva. *Developing iOS8 Apps Using Swift – Create a To-Do Application*. 10 June 2014. 8 November 2014. <<http://ios-blog.co.uk/tutorials/developing-ios8-apps-using-swift-create-a-to-do-application/>>.
- oggata. *iBeaconDemo/ViewController.swift at master · oggata/iBeaconDemo*. 3 July 2014. 8 November 2014. <<https://github.com/oggata/iBeaconDemo/blob/master/iBeaconDemo/ViewController.swift>>.
- . *Swift と iBeacon を使ってお母さんが自分の部屋に近づいて来た事を警告するアプリをつくる*. 4 July 2014. 8 November 2014. <<http://qiita.com/oggata/items/5de43d71692d1abcff7c>>.
- Raptor. *iOS Coremotion accelerator updates in background/ sleepmode*. 20 May 2013. 8 November 2014. <<http://stackoverflow.com/questions/16647628/ios-coremotion-accelerator-updates-in-background-sleepmode/16647719#16647719>>.
- SQUARE ENIX. *FINAL FANTASY IX アルティマニア*. SQUARE ENIX, 2004. Book.
- Subramanian, Anand. *Install GUI on Ubuntu server 14.04 Trusty Tahr*. 11 July 2014. 8 November 2014. <<http://www.htpcbeginner.com/install-gui-on-ubuntu-server-14-04-gnome/>>.
- Sverdlov, Etel. *How To Install and Secure phpMyAdmin on Ubuntu 12.04*. 7 November 2014. 8 November 2014. <<https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-12-04>>.
- . *How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu*. 5 November 2014. 8 November 2014. <<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu>>.
- vualoaitu and Flimzy. *How to run NSTimer in background and sleep in iOS?* 19 November 2013. 8 November 2014. <<http://stackoverflow.com/questions/15092016/how-to-run-nstimer-in-background-and-sleep-in-ios/19174810#19174810>>.
- Wolfram/Alpha: Computational Knowledge Engine*. 2014. Wolfram Alpha. 8 November 2014. <<http://www.wolframalpha.com/>>.
- Yamashita. *iPhone 5s の M7 チップを試してみる*. 18 October 2013. 8 November 2014. <http://wonderpla.net/blog/engineer/iPhone5s_M7chip/>.
- ニンテンドー3DS / すれちがい Mii 広場 : 無料で楽しめるゲーム. 2014. Nintendo. 8 November 2014. <<http://www.nintendo.co.jp/3ds/software/built-in/miiplaza/legend.html>>.
- ファイナルファンタジー. 2014. SQUARE ENIX. 8 November 2014. <<http://www.finalfantasy.jp/>>.
- ブレイブリーデフォルト フライングフェアリー. 2012. SQUARE ENIX. 8 November 2014. <<http://www.square-enix.co.jp>>.