Project Number: RL1-P309

Food Fight

Promoting self-care behaviors for the management of diabetes through a game

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

by:

James Thompson

Date: April 24, 2010

Professor Robert W. Lindeman, Major Advisor

Professor Charles Rich, Major Advisor

Abstract

Diabetes is a chronic disease, but a manageable one. The game *Food Fight* focuses on the importance of balancing diet and exercise in the management of diabetes especially in people over the age of 45. Although aimed towards diabetes patients, understanding this balance is beneficial to anyone.

Food Fight is a tower defense style flash game where different foods replace the classic creeps and exercise and diet towers replace the classic towers. Each tower and food type acts in ways that reflect how they act in real life. There are also random events, representing the unforeseen circumstances of life, which force the player to use helpful problem solving techniques.

In addition to the game, *Food Fight* communicates with a database that holds information about the player. During the game, questions appear from time to time when the player finds himself in trouble. The questions bring a certain aspect of diabetes management to the player's attention, and the player's answer, which is stored in the database, reflects his knowledge of management techniques. With enough players, and hence enough answers to questions, patterns in people's diabetes management techniques can be found.

Table of Contents

List of Figures

List of Tables

1. Introduction

In 2007, there were an estimated 23.6 million people in the United States with diabetes, accounting for about 7.8% of the population, as stated on American Diabetes Association website. Although diabetes is a chronic disease, there are management strategies that allow patients to live long, healthy lives. The American Association of Diabetes Educators (AADE) developed seven self-care behaviors to follow for the effective management of diabetes. As stated on their website, they are healthy eating, being active, monitoring, taking medication, problem solving, reducing risks, and healthy coping. For an individual, the virtually constant need to manage diabetes can easily become frustrating, increasing the importance of outside motivation in the management of diabetes.

One of the available sources for diabetes management incentive is provided by a company called Glymetrix. In their own words, Glymetrix "provides innovative engagement strategies to motivate better diabetes management through the use of rewards and incentives" by using "a game based interface to engage patients with the sophisticated data model at the core of the system". Glymetrix also extracts behavioral patterns of users from the information given through the game based interface, aiding medical professionals in their decision making.

Food Fight is a tower-defense style game designed to motivate the practices of three out of the seven AADE self-care behaviors: healthy eating, being active, and problem solving. For a description of a tower-defense game, see Appendix A – Tower Defense Explained.

This document is a detailed description of the design and implementation of *Food Fight.*

2. Game Design

Food Fight is a tower defense style game where the player must use towers in order to manage enemies trying to make their way across the screen. This is accomplished through clever choices of which type to use and strategic tower placement. The use of these mechanics is designed to reinforce good self-care behavior in the player's everyday life.

Since *Food Fight* was developed in order to promote beneficial self-care behavior in real life, the majority of elements in the game reflect an aspect of real life. The enemies in the game, classically called creeps, represent different food groups that people often find in their diet. The choosing and placement of diet towers represent a person's choice of diet. Exercise towers, then, represent how physically active a person would choose to be and what activities he or she would partake in.

In *Food Fight*, players use the placement of both exercise and diet towers to balance the consumption of food and use of energy just as someone would in their daily life. It is easy for a person to say that they are going to eat right or stick to an exercise plan, but actually carrying one out is difficult. To mirror this in the game, there is an economic resource used to 'purchase' towers, called motivation. Players start the game with a certain amount of motivation, use it when they place towers in the game, and regain it when their exercise towers absorb an enemy.

The better a player balances food consumption (through diet towers) and energy use (through exercise towers), the higher their score throughout the game will be. The game ends when all rounds of enemies have either made it to the end of their route or have been absorbed by the player's towers. There are no win or loss conditions in the game. The player's goal is to achieve the highest score possible.

2.1 Difficulties

Before the beginning of the game, the player is presented with the screen shown in Figure 1. The player can choose from taking a tutorial in order to learn the game basics, answering a question in order to get a boost in the game, or starting a real game in either easy or difficult mode.

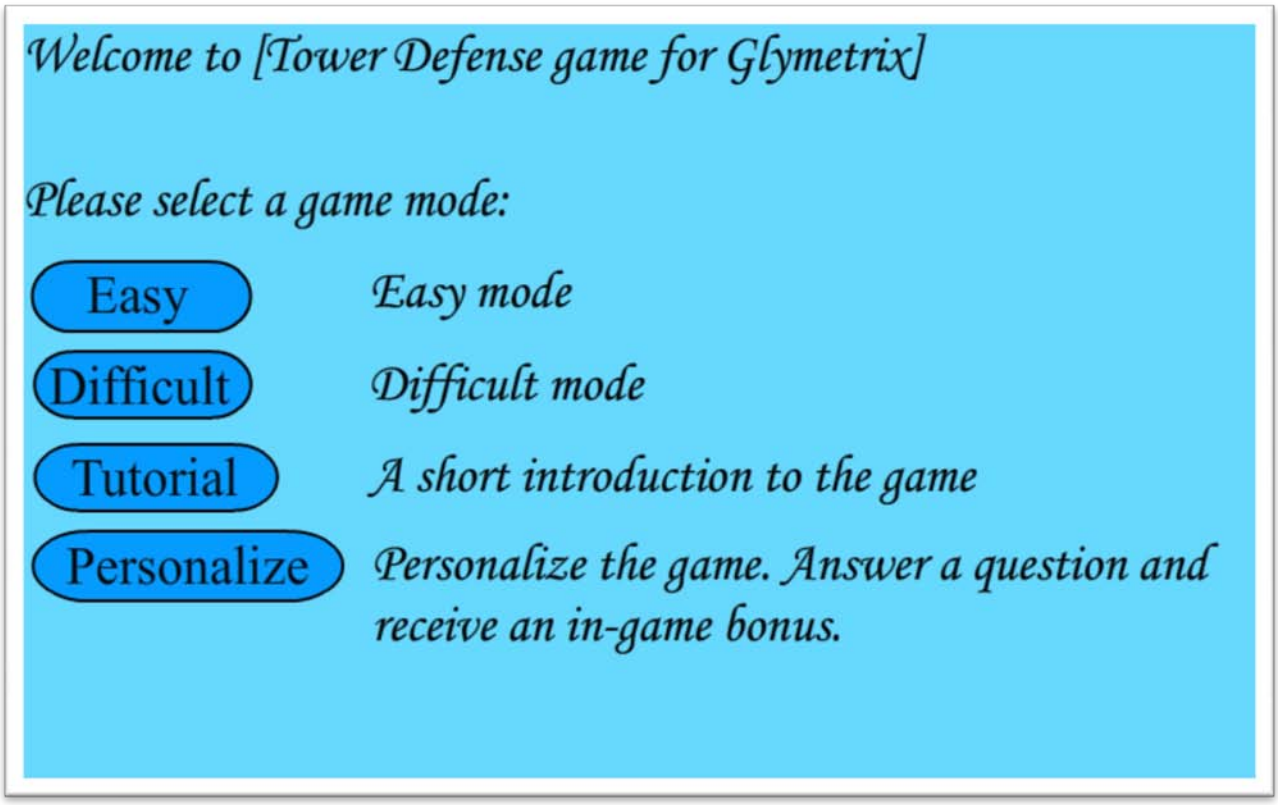

Figure 1: Choose Difficulty Screen

2.1.1 Tutorial

The game tutorial describes each element present on the game screen and how to utilize it. A small amount of basic strategy is also given to the player. After teaching the player the basic skills needed to play the game, the tutorial allows the player to experiment with the new found skills in a live game setting without adverse effects.

2.1.2 Easy Mode

In easy mode, the player is presented with enemies that try to travel horizontally through the arena.

2.1.3 Difficult Mode

In difficult mode, two directions of enemies present themselves to the player. Along with the horizontal enemies that are present in easy mode, there are enemies that travel through the arena vertically. The player is provided with the same amount of starting resources as in easy mode, however.

2.1.4 Personalization

If this option is chosen, the player is presented with a question regarding either knowledge of diabetes or knowledge of the management of diabetes. The player can choose to disregard the question, but if they give any answer, they receive an extra boost in motivation at the beginning of their game.

2.2 Towers

The player uses towers in order to attack enemies, dictate which enemies will be presented (See Diet Towers section), and influence enemy movements. The player is only allowed to place towers within a specific game area – called the arena. Towers are restricted to a grid-like placement and cannot overlap each other. In addition, towers cannot be placed in such a way that enemies are unable to reach their destination without having to go through a tower or outside of the arena.

2.2.1 Exercise Towers

Exercise towers attack enemies that get too close to the tower position. The three types of available exercise towers are walking, weight lifting, and swimming (Figure 2). These three activities were chosen because of their familiarity and availability.

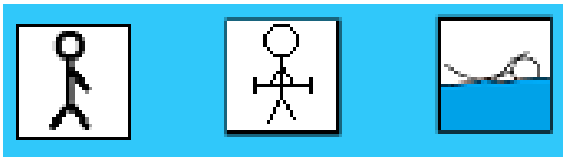

Figure 2: Exercise Towers

While enemies avoid walking and swimming towers, they move closer to weight lifting towers. The inclusion of weight lifting towers emphasizes the importance of balancing aerobic and anaerobic exercises.

Each tower type attack enemies within its range. Walking towers have a small range while weight lifting towers and swimming towers have respectively incremental ranges (Table 1). The discrepancies in ranges reflect the different intensities of each activity. Each enemy has a certain amount of hit points. The three types of exercise towers decrease the hit points of enemies within their range at different rates, with walking being the slowest and swimming being the fastest.

Table 1: Exercise Tower Attributes

Tower Type	Range	Damage Rate per Frame	Cost	Enemy Effect
Walking	52.5	0.15	15	Repels Enemies
Weight Lifting	70	0.2	25	Attracts Enemies
Swimming	87.5	0.25	35	Repels Enemies

The player also has the ability to activate and deactivate exercise towers at any time. Only exercise towers can be activated and deactivated because exercise towers are the only type that has the ability to attack enemies. Deactivating towers helps to conserve motivation by keeping the tower available for use later on in the game. Also, enemies continue to move around deactivated towers, allowing the player to keep the enemy movement constant while decreasing the number of attacking towers.

2.2.2 Diet Towers

Diet towers do not attack enemies that come close. They can be used, however to influence the types of enemies that are presented and the movement of enemies in order to make the exercise towers more effective.

The three diet towers available are protein, carbohydrate, and fat towers (Figure 3). The ratio between the different types of towers placed in the game is used to determine which types of enemies are sent during each wave. A bad ratio of diet towers results in an imbalance in the types of enemies that are generated. Too big of an increase in any type of enemy results in the change of an attribute in order to increase difficulty (See Enemies section).

Figure 3: Diet Towers

2.2.3 Selling Towers

Selling a tower causes the player to gain a percentage of the tower's cost back and removes the tower from the arena. This ability allows the player to fix mistakes relatively easily or change their strategy mid-game.

2.3 Motivation

It takes effort to eat a healthy diet and have a well balanced exercise routine. This effort comes from a person's ability to motivate him or herself. This is reflected in game by having the player use a resource called motivation to purchase towers. Destroying enemies is the only action that provides the player with new motivation. Selling towers, however, causes the player to regain some of the motivation they used in order to place the tower originally. A player's motivation amount does not influence his or her score.

2.4 The Meter

The meter represents a blood sugar level that reflects the lifestyle represented by the towers placed in the game. The meter is divided into five different sections, with each representing 50 units of the meter (Figure 4). When an enemy is moving through the game area, the meter increases, like a person's blood sugar level would when he is eating. Similarly, the meter decreasing when exercise towers destroy enemies correlates to someone using energy gained from eating food, resulting in a decrease in their blood sugar level. The meter does not move in biologically accurate ratios.

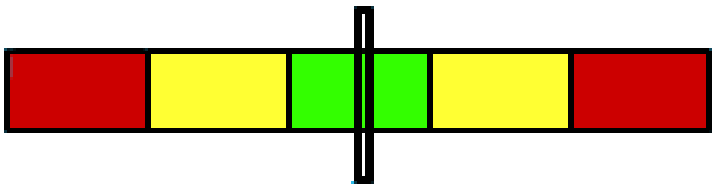

Figure 4: Meter

The meter automatically increases in direct proportion to the number of enemies on screen at any given time. More enemies on screen mean the meter will increase faster. If an enemy is destroyed, the meter is decremented by an amount dictated by the type of enemy destroyed (see Enemies section).

Meter position influences the rate at which a player's score will increase. If the meter is in the middle green area, then the player will gain the highest scores possible when they perform a scoring action (see Score Section). The player gains smaller percentages of these scores as the meter travels through the yellow and red areas.

If the meter moves to either the left or right extreme for more than ten seconds, a question is presented to the player. The purpose of the question is to both educate the player about diabetes and garner information about what the player already knows. The player again has the option of not answering, but if they do, the meter moves the equivalent of one section towards the green section.

2.5 Enemies

Enemies in *Food Fight* try to travel either from the left of the arena to the right or from the top to the bottom. They move around towers and avoid towers that attack them as much as they can (with the exception of the weight lifting tower), but the need to get to their destination forces them to expose themselves to tower attacks at times.

Once the player presses the start button, enemies begin to travel through the arena in waves. Every 15-20 seconds, the next wave of enemies is generated. The number of each type of enemy sent in each wave is determined by the ratio between the amounts of each type of diet tower that the player has placed in the game.

2.5.1 Types

There are three main types of enemies representing proteins, carbohydrates, and fats. As the number of carbohydrates increases, their speed increases, allowing them to get through exercise towers' ranges quicker. The speed of each carbohydrate starts at about 0.6 and increases

by .2 for every carbohydrate enemy after the first three. Similarly, as the number of fats enemies increases, their toughness increases, making them harder to absorb by exercise towers. Fats enemies gain a full hit point for every four fat enemies on screen.

2.5.2 Effect

Each enemy, while moving across the screen, causes the meter to increment at a low rate. A greater number of enemies on screen at the same time causes the meter to move at an increased rate. When an enemy is destroyed by a tower, the meter is decremented by an amount that is dictated by the type of enemy that was just destroyed. It is the balance of these two forces that the player tries to keep in check throughout the game.

3. Implementation

The most difficult part of the implementation turned out to be learning the intricacies of the programming language and the development tools used with it.

3.1 Tools

This game was developed using Flash. The main reason for using Flash was because the framework that Glymetrix already had in place was set up to incorporate Flash based games. Besides that, however, Flash was chosen because it is well-supported on virtually all web browsers, and can be viewed by anyone who has the Flash plug-in. Flash is also completely cross-platform, able to run on UNIX, Windows, and Mac computers.

3.1.1 Unit Tests

One of the goals of the design was to service future changes and additions to the game. Unit testing not only provides validation that units of code work properly, but also simplify integration of new units and features. In order to develop unit tests, an open-source framework called AsUnit was used.

3.1.2 Procedural Testing

Unit testing only makes sure each individual unit of code works independently. The interactions between different objects (e.g. making sure events are thrown and caught correctly), are not caught in unit testing.

In order to address this issue, a tool was built that runs through a game at a sped up pace and picks out any errors that are thrown. Every action that takes place in a game corresponds to a string that the testing tool will recognize and emulate. Each event string includes an identifying string assigned to the event along with any vital information that may be needed in order to carry

out the event accurately. For example, the event string “AeTp0Xp-109Yp132Tc0” parses as follows:

Ae – “Add enemy”: an enemy was generated.

Tp0 – “Type 0”: The enemy is of the type corresponding to 0.

Xp-109 and Yp132: The enemy was generated at the x position of -109 and y position of 132.

Tc0: The event happened when the tick count was 0.

A list of event strings simulates a game.

There are two ways that a list of event strings can be produced. One is by hand (See Appendix B), and the other is to run the game in debug mode. Running the game in debug mode will cause it to run slower because (1) the string is printed out periodically in order to ensure that errors do not cause the game to exit without giving a debug string, and (2) the string can become lengthy, which uses processing time to print. Even so, the tool was used throughout development in order to gather a library of game debug strings. These strings were then run whenever a major addition was made to the game in order to ensure the addition did not interfere with anything.

3.2 Classes

3.2.1 Class Hierarchy

Each enemy has a specific speed, resistance, icon, and other variables of the same kind. Therefore, each enemy has their own class. The enemies are then divided into the enemies that go across the screen from left to right (CreepLat) and enemies that move across the screen from top to bottom (CreepLong). Because of the different movement directions, each type of enemy needs its own movement calculations. Every enemy falls under the main class Creep (Figure 5), which holds the basic get and set functions along with some helpers which calculate such things

as the arena's array index that corresponds to the current position of the enemy.

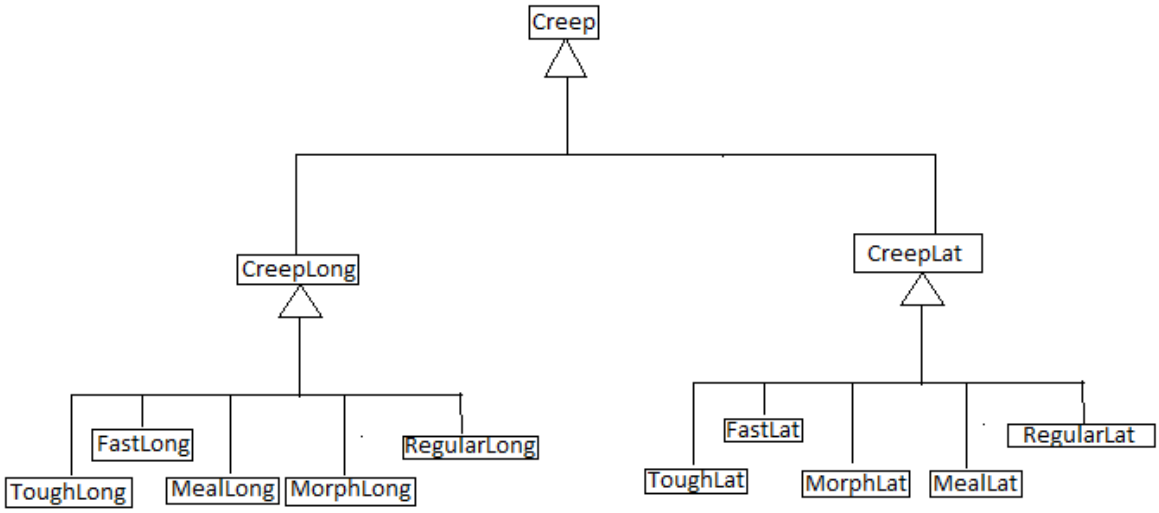

Figure 5: Enemy hierarchy

All TowerButtons represent an action that the player can take in the game arena. This includes placing any one of the six towers, selling a tower, and disabling or re-enabling a tower (Figure 6). All other buttons are children of the ButtonBase class, which holds the code for enlarging and reducing the button size on mouse over and mouse out events, respectively.

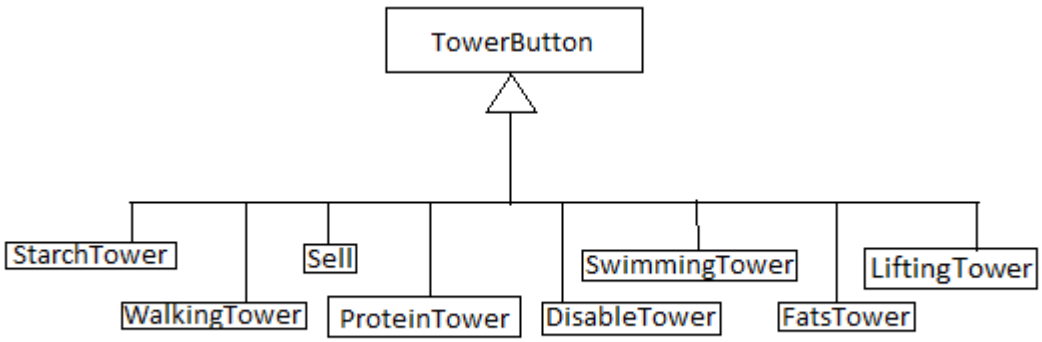

Figure 6: TowerButton hierarchy

3.2.2 Arena

The area of the screen called the arena is where the majority of the game takes place (Figure 7). A simple array represents the arena. Each element in the array represents a 20 x 20 area. Every tower occupies four areas when placed in the arena. Every array element holds data such as how many towers can attack the area the element represents and where enemies in that area should move towards next.

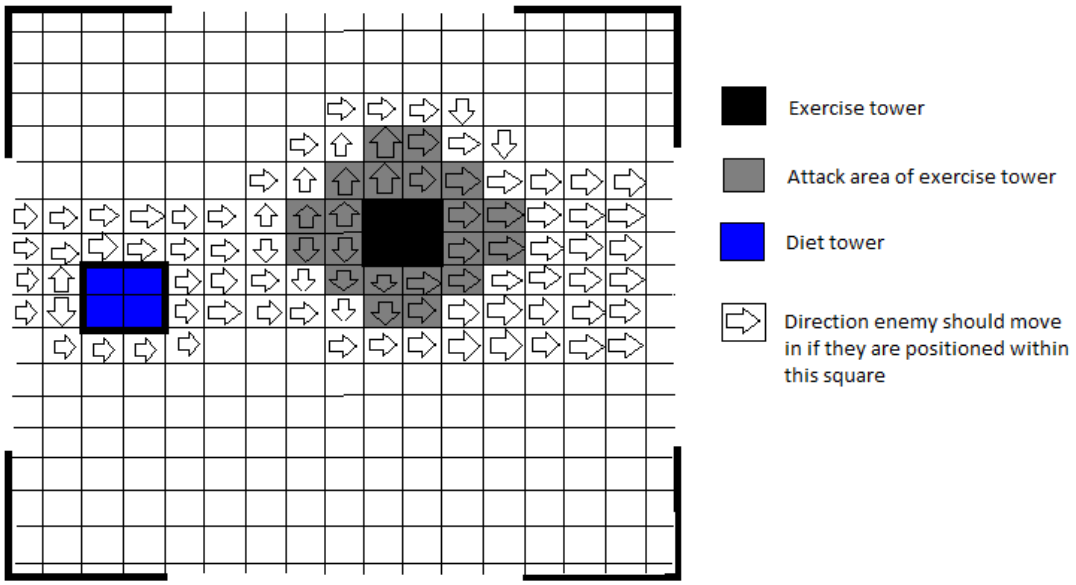

Figure 7: The Arena

3.2.3 Enemies

The enemies rely on the arena for optimal directions through the arena. At the beginning of each enemy movement calculation (which is different from enemy movement), the arena runs a global A* algorithm in order to find distances and costs from each area in the arena to an end node. Each element in the arena's representative array holds an identifier for the direction the enemies should travel if they should find themselves in the area represented by the array element.

Each array element holds two identifiers, one for horizontally oriented enemies and one for vertically oriented enemies.

3.2.4 Towers

Towers rely on the arena for information management. The arena checks for tower overlap and blocking during tower placement and also tracks tower densities. This then allows the arena to handle user requests to either sell, activate, or deactivate towers efficiently in addition to providing accurate movement services to enemies and keeping the game rules enforced.

For attacking towers, collision detection between the enemies and the tower icon – which includes a virtually invisible circular shape representing its range – determines when the tower starts its animation and displays the attack icon. All towers attack the enemy that is furthest along its respective path. For example, assuming they are both in the tower's range, an enemy moving vertically that is 10 spaces away from its end is attacked before a horizontally moving enemy 12 spaces from its end. The distance an enemy has left to go is kept in the array representing the arena and retrieved based on the enemy's location.

3.3 Score

The score is changed whenever an enemy is removed from play. *Food Fight* is not designed as a classic tower defense where all enemies should be destroyed as quickly as possible. The goal is to balance the number of enemies with the number and types of towers used. That is why even enemies that make it through the arena get points scored for them. The areas in the meter act as multiplier regions where the green area will multiply the score for a removed enemy by 3, the yellow will multiply it by 2, and the red by 1.

4. Back End Integration

Along with helping players manage their diabetes, *Food Fight* was designed to gather information from the player in order to develop behavioral models of diabetes patients. This is done by presenting questions to the player and then recording the player's answers. In order to do this, *Food Fight* communicates with a database.

4.1 Tools

4.1.1 Ruby on Rails

The web framework used by Glymetrix is Ruby on Rails - a relatively new web framework based on the Model-View-Controller architecture. The framework is used as the bridge between the MySQL database and the flash game.

4.1.2 MySQL

MySQL databases are used worldwide, making them the most popular open source databases. They are known for their quick performance, reliability, and ease of use. Like flash, MySQL runs on a wide range of platforms, including Linux, Windows, Mac, and Solaris.

4.2 Overview

A MySQL database holds the information used in the game, including the questions asked to the player and the options for their answer. A Ruby on Rails architecture is then used in order to retrieve information from the database. Within the Ruby on Rails architecture, web pages use html for design and formatting. Finally, the flash game is embedded within this html (Figure 8).

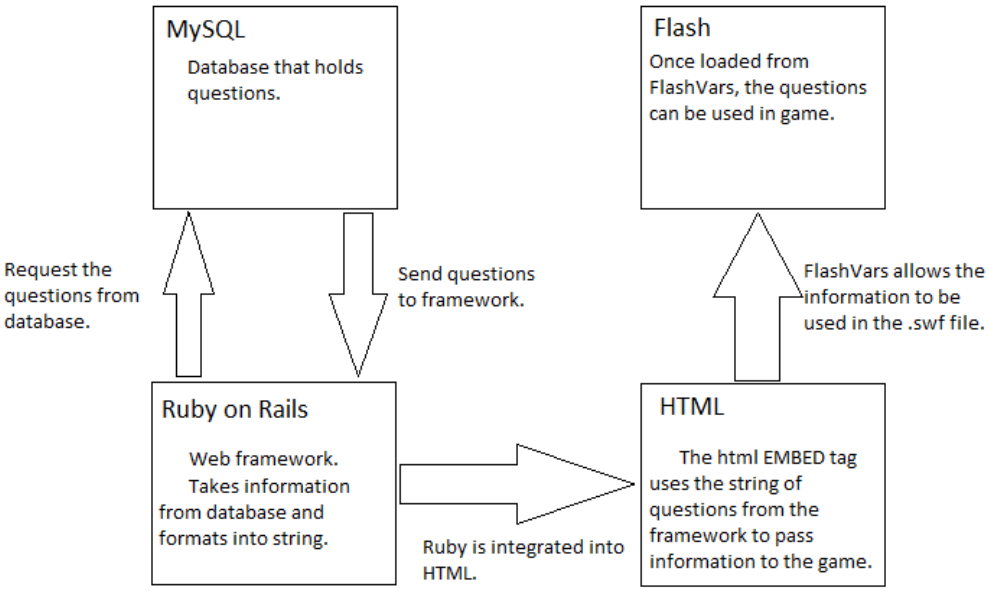

Figure 8: Back End

The naming scheme connects Ruby on Rails with the MySQL database. Singularly named controllers are associated with the database table with the plural version of the same name. For example, a controller in the Rails architecture called ‘Question’ is associated with the database table called ‘questions’.

The code for the Ruby on Rails and HTML portions is located in an rhtml file in the hierarchy followed by Ruby on Rails. The rhtml is mainly an html file, but has the ruby code integrated in with a special tag (‘<%’ or ‘<%=’).

Variables being passed into the flash game are passed by placing the variables in a specifically formatted style at the end of the string specifying where the game file is located. After using a loader class within flash, the variables that originated from the database can be used in game.

5. Evaluation

Many of the testers for the game were either friends or people involved with Glymetrix in some capacity, whether professional or not. Even though the game target audience is diabetes patients, the effects of the game apply, and are beneficial, to everyone. Therefore, feedback from non-diabetic testers was still beneficial towards game evaluation.

There were 10 testers from WPI, aged 20-22. Each tester was asked to play through the game 2 or 3 times, taking between 10 and 15 minutes per person. Video was taken of their on screen actions so that patterns could be found in game strategies that both worked and failed. They were also asked to talk into a tape recorder, dictating what their thought process was during their play. With these two resources, elements of the game that were missed by the player or elements that caused confusion could be identified and fixed.

5.1 Lessons Learned

While testing helped to find errors in game code and balance out the numerical relationships between different aspects of the game, there were also some higher-level deficiencies that were pointed out.

One of the most important lessons learned from testing feedback was the importance of simplicity. The movement of the game meter was originally designed to depend on a large number of game factors including the number of enemies on screen at any given time, the numbers of each tower type present in the game and the amount of motivation the player had left. While the design physically worked, players were becoming confused with how to manage the meter. Meter movement was then changed to rely on only the number of enemies that were on screen and the number of enemies the player destroyed. Subsequently, players had a much easier time managing the meter, leading to a more enjoyable experience.

Players were also confused about what the diet towers actually did in the game. It was explained in the tutorial, but there was no explicit clue in the game. This problem was never properly remedied, but brought focus to the fact that giving visual feedback to the player is critical.

Similarly, players struggled with the user interface due to false visual clues and lack of affordances. The icon used to represent the mouse cursor had been ignored, and the default settings in flash had the cursor change to the same icon when it rolled over both buttons and static text areas. Players then thought that the text areas were clickable but nothing happened when they clicked, causing confusion. To fix this, the mouse cursor icon behavior was fixed to only change when rolled over buttons and buttons were made to enlarge themselves when rolled over.

5.2 Project Goal Evaluation

One of the purposes of *Food Fight* is to emphasize the importance of a good balance between diet and exercise. Six out of the ten testers from WPI commented on the relatively small number of exercise towers needed in order to keep the game in balance, suggesting that the game is getting the player to think about the relationship between exercise and diet. Similarly, four out of the ten testers commented on the number of fat towers necessary to keep the game in balance, suggesting that the game causes a relatively large portion of the players to think about dieting habits.

Some game intentions did not manifest themselves so clearly in the player comments. The random events are designed to help the player with problem solving, reflecting real life events that are uncontrollable by the player. In four of the ten tester's comments, the random events

were described as irrelevant or annoying. These comments suggest that the purpose of the random events is not communicated well to the player.

6. Conclusion

The goal of the project was to have a complete game done by the end. While that goal was not fulfilled, the project resulted in a game prototype that could become a full game with a little more work.

6.1 Next Steps

An important change to make to the game is to develop better art. The current icons and colors were placed as filler, and more professional looking artwork is needed for a finished game. Similarly, the written descriptions and questions asked to the player are filler and need to be replaced with the correct data.

The game has not been hooked into the actual database that it will be conversing with either. It currently runs off of a stand-in. Connecting the game to the correct database will help with getting the correct questions into the game, as mentioned above, and will start implementing data collection from the player.

6.2 Reflection

In addition to using the lessons learned from the testing phase [see Testing section], there were aspects of this project that could have been executed better. For example, more planning probably should have occurred right at the beginning of the project. More specifically, more research and practice with actionscript would have been most beneficial. It was not until about halfway through the project that I discovered exactly how actionscript had designed events as the main communication mechanic between objects, which would have helped speed up development code from the beginning.

Writing test cases and procedural testing modules would also have helped earlier in the development process. Subtle code errors would have been much easier to catch and behavior

issues would have been much faster to test, causing the whole process to progress much faster and easier.

Finally, the code could have been developed to accommodate change more readily. There was a point where a suggestion to increase the size of the playing area was put forward, but it turned out to be a long and involved process because of the way the graphic of the area was connected to the logic in the code. More well-designed connections between an element's graphic and its controller would have greatly improved the project.

Appendices

Appendix A – Tower Defense Explained

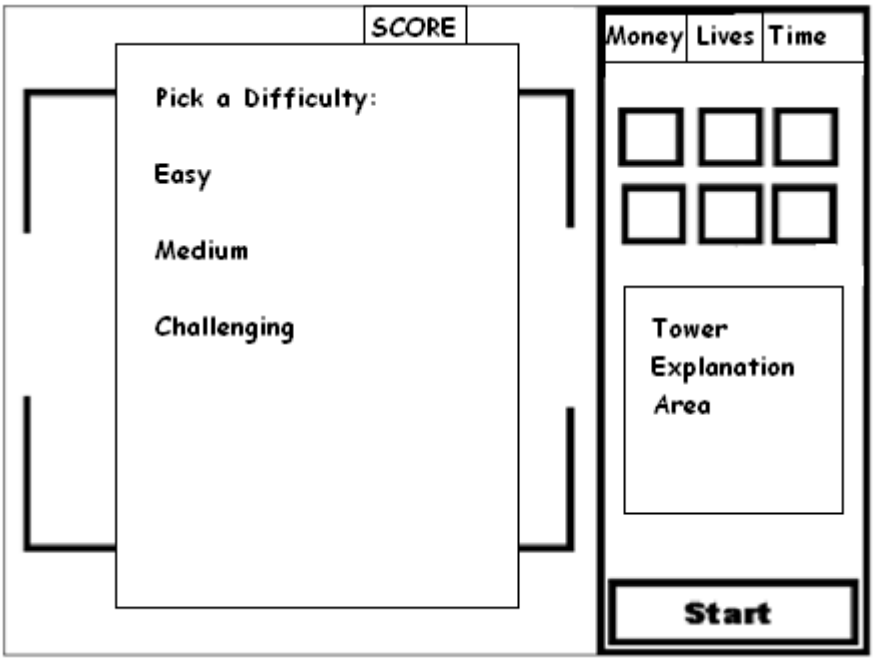

Figure 9: Choose Difficulty

Many tower defense games start with asking the player to select a difficulty level to play at (Figure 9). Once that happens, the player is brought to the actual game, but nothing starts happening. There may be an intermittent step asking the player to pick a scenario, but that is not always the case.

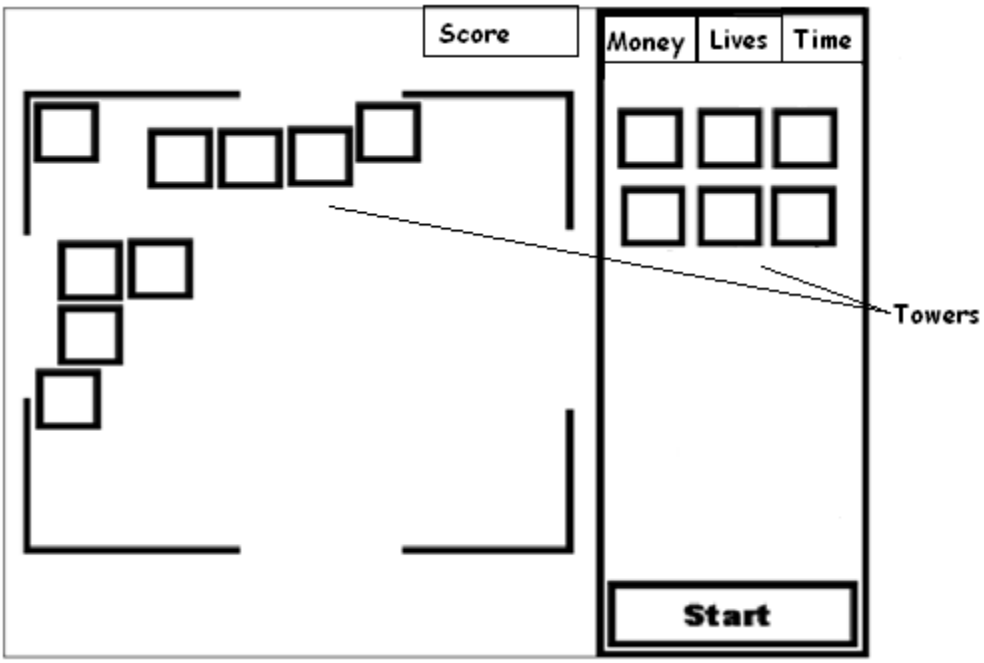

Figure 10: Tower Defense Game

The player can then start placing towers in the game in order to prepare for the enemies. Every tower costs money, and the player begins with a given amount of money. Some versions of tower defense have paths that the enemies travel through, which towers cannot be placed on. Other versions (Figure 10) have the whole area open to both towers and creeps, but towers cannot be placed so as to block the path from one side of the area to the other.

Once a player has used all of his money, or is ready to proceed, they press the start button to initiate the enemies. Enemies usually come in waves every set amount of seconds. Sometimes, there is even a scrolling bar at the bottom of the screen notifying the player of what wave types are coming next.

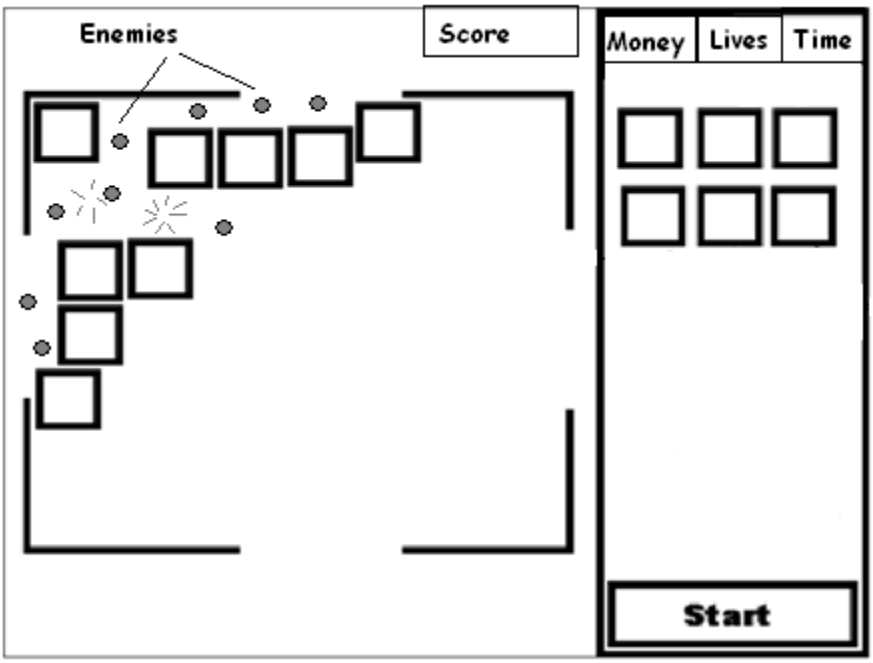

Figure 11: In-game Tower Defense

The goal of the player is to defeat all the enemies that come on screen before they reach the other side of the screen. Easier game modes have only those enemies going from left to right while the more difficult modes have enemies going from the top to the bottom as well (Figure 11). Many game versions have small health bars over the enemies in order for the player to know how much damage is left before they are defeated.

The player is given a certain number of enemies that can get through its defenses before the player loses. A player wins if he survives all waves of enemies before the number allowed through reaches 0.

Appendix B – Test Strings

Debug string	In-game action represented
Eb	Player begins a new game on easy mode
Hb	Player begins a new game on difficult mode
St	Player pressed the start button
Cm	All the creeps move
Sl	Player selected a tower type
Pl	Player placed a tower
Pa	Game was paused
Re	Game was restarted
Ae	A single enemy was generated and put in the game
Nw	The next wave of enemies was sent
Cd	Calculations for enemy orientation are made
E0, E1, E2, E3	A random event occurred
He	Player pressed the help button
Sb	Player alerted of an attempted blocking move
Hb	The blocking alert is removed from the game
S0, S1, S2	The active random event has expired

Appendix C – Extra Reference Materials

These links provide information about games dealing with health issues and companies that are working with them.

The Project Sponsor:

http://glymetrix.com/

More information on similar games:

http://www.gamesforhealth.org/video.html

American Association of Diabetes Educators:

http://www.diabeteseducator.org/

Helpful implementation references:

"Learn Flash Professional CS4." *Adobe*. Web. <http://tv.adobe.com/show/learn-flash-professional-cs4/>.

Lenz, Patrick. "Learn Ruby on Rails: The Ultimate Beginner's Tutorial." *Sitepoint*. N.p., 31 Jan 2007. Web. 1 Apr 2010. <http://articles.sitepoint.com/article/learn-ruby-on-rails>.

"MySQL Commands." *Pants.org*. N.p., 21 Jan 2010. Web. 1 Apr 2010. <http://www.pantz.org/software/mysql/mysqlcommands.html>.

"ActionScript 3.0 Language and Components Reference." *Adobe*. N.p., 16 Jun 2008. Web. 1 Apr 2010. <http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/>.