

Creating an Online Game for Farm Safety

A Master's Thesis

Presented to

Master's Program in Information Design and Technology

In Partial Fulfillment
of the Requirements for the

Master of Science Degree

State University of New York
Institute of Technology

By

Robert D. Begley

April 2013

© 2013 Robert D. Begley

SUNY Institute of Technology

DEPARTMENT OF COMMUNICATIONS AND HUMANITIES

CERTIFICATE OF APPROVAL

Approved and recommended for acceptance as a thesis in partial fulfillment of the requirements
for the degree of Master of Science in Information Design and Technology

DATE

Dr. Ibrahim Yucel

Thesis Adviser

Dr. Russell Kahn

Second Reader

Abstract

The New York Center for Agricultural Medicine and Health (NYCAMH) has a need for a web-based educational game to educate families on farm safety. This project demonstrates a proposed game design that combines the elements of gaming that enhance learning with the feedback received from NYCAMH. Feedback was solicited through a series of prototypes delivered to NYCAMH through an agile software development process. The proposed design follows a constructivist approach to place the learner in a context based on reality. The aspects of the game design that engage and motivate students by blending entertainment with learning are discussed. A prototype for this project can be found at http://web.cs.sunyit.edu/~begleyr/nycamh/prototype_5/.

Table of Contents

Abstract iii

1. Introduction.....1

 Overview.....1

 Research Questions1

2. Literature Review2

 Games and Learning2

 Student Motivation4

 Game Design.....5

 Platform and Framework7

 Programmatically Logging Game Play Data8

 Literature Review Summary8

3. Methodology and Design.....9

 Target Audience9

 Industry Survey9

 Development Blog.....10

 Game Concept and Overview10

 Educational Objectives12

 Integration of NYCAMH Educational Elements13

 An Agile Development Process14

 Theoretical Foundation for Game Design15

 Development Tools16

4. Game Design Framework17

The Farm Canvas and Window Frame	18
Animating the Farmer.....	19
Farm Object Types	21
Other Farm Object Attributes	21
Passing Game Components Down to the HTML Document.....	22
Saving Games Using AJAX.....	23
Animations and Adobe® Edge Animate	24
Initialization of Database Tables.....	24
5. Anticipations and Limitations.....	25
6. Findings and Results.....	26
Agile Development and Educational Game Development	27
Test and Evaluation	27
Further Study and Game Enhancements.....	28
7. Conclusion	30
References	33
Appendix A. Play It Safe Screen Shots.....	36
Appendix B. Source Files.....	43

List of Figures

Figure 1: A Typical “Play It Safe” Game Play Flow	11
Figure 2: Information Sign Post.....	12
Figure 3: Question Sign Post	13
Figure 4: NYCAMH Play It Safe Feedback Sheet Category Breakdown.....	14
Figure 5: The Agile Development Process.....	15
Figure 6: NYCAMH Play It Safe Game Design.....	17
Figure 7: Canvas scrolls under CSS frame in browser window	19
Figure 8: NYCAMH Approved Male Farmer Image.....	20
Figure 9: mySQL table data delivered to HTML Document.....	23
Figure 10: Creating Animations with Adobe® Edge Animate	24
Figure 11: Attempting task without necessary inventory.....	36
Figure 12: Reading information sign post by silo.....	37
Figure 13: Checking inventory list	38
Figure 14: Door opens during silo animation.....	39
Figure 15: Player attempts question about silo.....	40
Figure 16: High score lists.....	41
Figure 17: Play It Safe game menu	42

1. Introduction

The New York Center for Agricultural Medicine and Health (NYCAMH) is an organization that enhances agricultural and rural health by preventing and treating occupational injury and illness. NYCAMH has a board game called “Play It Safe”. The board game includes a large amount of safety information. They plan to replace the board game with a web based farm safety game. This next generation Play It Safe game will need to include all of the farm safety information as the board game.

Overview

This project attempts to fill the need NYCAMH has for an online farm safety game. Through a thorough literature review and survey of other online games, a concept for an adventure style or role playing game (RPG) game was created that creates constructivist learning environment.

Research Questions

The main objective of the game is to educate the players on farm safety. So we would like to have the question “What are the best strategies and techniques to maximize learning when designing online educational games?” answered before we start a design or even a concept. The literature review was extremely informative as it summarized the different approaches and how they related to theories on learning.

Once a concept and design strategies were formed we were face with a more technical question of “What web technologies are most appropriate for creating an online game?” The answers to this question seem to be changing quickly as the web itself becomes more mature and as web development tools become more robust and automated.

2. Literature Review

The concept of using games as an educational tool is not new and the technology that delivers the gaming environment has grown and expanded as technology has developed over time. This not the first thesis paper for an educational safety game for the NYCAMH. Michael Van Dusen's (2012) "GAMES THAT CAN SAVE LIVES: Designing the next generation NYCAMH safety game" documents the testing of an electronic prototype game. Coupling the information contained in the Van Dusen study with the practices, theories, and strategies found in the collected article list, a new plan for a web-based safety game can be designed and implemented that better meets the needs for the NYCAMH.

Games and Learning

Of course there are many theories on most effective educational methods and there also seem to be just as many theories on educational gaming. The underlying goal in all articles is to motivate a student through the entertainment aspects of a game and insure that the subject knowledge received and retained. "Effective games embed learning in meaningful situations that are endogenous to the game itself. The personally meaningful and valued social and material worlds in which game learning takes place may be 'virtual' from an outsider's perspective; however, they have a psychological reality for the player that directly mediates the player's level of immersion, persistence in the face of challenges, and intrinsic desire to learn." (Wideman, Owston, Brown, Kushniruk, Ho, & Pitts, 2007, p. 2-3)

Kafai (2006) writes "The instructionists, accustomed to thinking in terms of making instructional educational materials, turn naturally to the concept of designing instructional games" (p 37). These instructionist perspectives on educational gaming follow a more linear pattern to game play and follow the traditional instructional educational patterns. Most of the

articles follow a more constructivist approach discussing the advantages of students learning alone or collaboratively in a group using hands on gaming applications where they become designers based on choices they make either through game play, preferences, modding, and sometimes programming. “Without wanting to deny the value of instructional games, constructivists have focused their efforts in a very different direction. Rather than embedding ‘lessons’ directly in games, their goal has been to provide students with greater opportunities to construct their own games” (p. 38).

“Hypermedia technologies facilitate constructivist approaches (Dillon & Gabbard, 1998; Jonassen, 1994; Merrill, 1991) that leverage human psychological traits such as curiosity or satisfaction with achievement” (Moreno-Ger, Burgos, Martínez-Ortiz, Sierra, & Fernández-Manjón, 2008, p. 2531).

“Several researchers (e.g., Begg, Dewhurst, & Ellaway, 2004; Delwiche, 2006; Herz, 2001; Pivec & Dziabenko, 2004) argue that a specific mode of networked learning, learning through online games, constitutes a promising constructivist learning paradigm that should be embraced by TE institutions with a view to providing students with powerful active learning environments and lifelong learning concepts” (Papastergiou, 2008, p. 20).

Several of the articles speak to the idea of conceptual learning where the student is put in a similar, but virtual context of the subject matter being taught. According to Wideman et al (2007), “It is generally accepted that useful knowledge is contextualized knowledge—the learner must know when and where to use it (Bransford, Brown, & Cocking, 2000). Games can provide experiences across multiple situated contexts that enable learners to understand complex concepts ‘without losing the connections between abstract ideas and the real problems they can be used to solve’ (Shaffer et al., 2004, p. 5). Such situated learning is considered to be critical in

two central theories of learning: situated cognition (Lave, 1998; Lave & Wenger, 1991) and constructivism (Bransford et al., 2000). These theories emphasize that learning is shaped and constrained by the intentions of the learner and situational factors (Squire, 2002). Evidence of the importance of meaningful contextualization in educational games can be seen in a series of experiments by Lepper and colleagues, who found that providing concrete contextualizations for solving game like puzzles by embedding them in a simple fantasy narrative markedly enhanced student motivation and learning (Cordova & Lepper, 1996; Parker & Lepper, 1992)” (p. 3).

Shaffer, Squire, Halverson & Gee (2005) “have argued that video games are powerful contexts for learning because they make it possible to create virtual worlds and because acting in such worlds makes it possible to develop the situated understandings, effective social practices, powerful identities, shared values, and ways of thinking of important communities of practice” (p. 5).

Student Motivation

All articles seem to indicate that the main advantage to using games in education is to increase student motivation and engagement. Lin, Wu, & Wang (2011) wrote “Researchers believe combining web-based digital games with learning processes is an effective way to enhance learners’ motivation thus improve outcome” (p. 736). And they later go on to state, “*Game-based learning*- It’s an effective way to improve learners’ motivation by combining game play in the learning process. A. Martens, H. Diener, and S. Malo pointed out that the game creates a virtual environment, allowing the players to play a completely different role from real life and into the game as part of the game” (p. 738).

“When the learning process is routine and boring, a game-like learning environment can effectively make this activity enjoyable because a game brings learners sense of curiosity,

challenging, and interactivity by providing unpredictable outcome, feedback, and appropriate complexity” (Lin, et al, 2011, p. 737). However, Moreno-Ger, Burgos, Martínez-Ortiz, Sierra, & Fernández-Manjón (2008) warn us about educational games that fail to sustain and adequate entertainment level, “When the entertainment aspects fail to shine in the design, most of the advantages of game-based learning in terms of motivation and engagement are lost and the learning experience suffers” (p. 2532).

Yucel, Zupko, Seif El-Nasr (2006) found that there is increased motivation when students have control over how the game works, looks, and feels (modding). “Students also may experience increased motivation, as they are working with tools that are close to what they would be using if they were working as game professionals” (p. 2)

Game Design

Several of the articles discuss good game design to not only engage and motivate the student but sustain the engagement.

Gee (2005) discusses sixteen learning principles that can be found in well designed games. A player assumes an Identity within the game by creating a character. "No deep learning takes place unless learners make an extended commitment of self" (p. 34). Gee reflects on Plato's description of books as being passive and stresses the importance of the Interaction principle. "Games do talk back. In fact, nothing happens until a player acts and makes decisions" (p. 34). Gee states that players become producers if the game follows the Production principle, their decisions and the paths they take through the game make them co-authors.

Gee's (2005) Risk Taking principle states that a good game will lower the consequences of failure and encourage a player to take chances. "Players are thereby encouraged to take risks, explore, and try new things. In fact, in a game, failure is a good thing" (p. 35)

Yucel, et al (2006) researched using game modding as a vehicle for learning, as opposed to game play or game creation. They evaluated several games and focused on “(1) the flexibility of the engine, (2) its storytelling ability, (3) the learning curve required of a middle-to-high school age student, and (4) the included art content” (p. 3). After evaluating games for use in a class they selected Warcraft 3 mainly due to its ease of customization. By surveying students prior to the course and again after the course was completed they saw an increase in student self-efficacy. The class titled “*Gaming for Girls*”, taught IT skills and “demonstrates the utility of game modding for (a) increasing motivation and self-efficacy when working with computers, and (b) teaching IT-based skills to an all-female audience” (p. 12)

Squire, Jan, James, Wagler, Martin, Devane & Holden (2007) in developing best practices submit design principles to follow when designing augmented reality games for learning. The *Identifying contested spaces* principle involves selecting an area or space for the game to take place. “Identifying conflicts over space gives designers a hook into a particular place” (p. 287). The “*Interactive Storytelling*” principle consists of the creation of a storyline by developers that players will inhabit. The principle of *Using Transformative Objects to Trigger Memorable Moments and Transformative Experiences* is to design “experiences that transform or provide a new framework for understanding phenomena” (p. 289). A game that follows the *Games as a Context into Inquiry* principle allows the learner to experiment without consequence. “Games might be considered a classic example of a ‘practice field’, in that games of contexts marked off from the world” (p. 290).

Rankin, Vargas & Taylor (2009) discuss how to test an educational game. They created two games that were Unreal Tournament mods. These games were FPS (first person shooter). Like the Van Dusen study they used a survey to collect data from the sampled college students

who participated. Their “surveys suggest that it could be worthwhile to develop tools and tutorials for teachers to develop metaphorical educational FPS games for their classes” (p. 5).

Platform and Framework

Choosing a platform and software framework to design a game is critical for quality of game play, maintainability, upgradability, and its ability to reach a large audience which spans a large geographic region. Many of the articles spoke about modding existing games for educational purposes by both teachers and students, however these types of games are not web based and are more appropriate in a classroom having computers with the games installed.

Juracz (2010) who developed a framework (Holorena) for web-based game based education discusses the issues of browser compatibility issues when using the web as a development platform, “Programming interactions in the client’s browser can be implemented in JavaScript; multimedia, audio and video content can be displayed with HTML and CSS features; however developing complex applications in standard HTML targeting a wide set of browsers on International Journal of Software Engineering & Applications (IJSEA), Vol.1, No.4, October 2010 3 different operating systems can still raise compatibility issues. Despite that browsers started to support some aspects of the being developed HTML5 and CSS3 specification, the speed and the smoothness of rendering the visual content can vary significantly [16]. To eliminate compatibility problems, and to be able to provide an outstanding, game-like, highly interactive experience we think, that among all the currently available technologies, Adobe Flash is a good platform for developing educational games on the web” (p. 2-3). Brom, Šisler, & Slavík (2010) also chose to utilize Adobe Flash in the development of their game Europe 2045, and Súilleabháin, Walsh, & Sime (2009) utilized Flash for their Comet Chaser Exhibition.

Moreno-Ger, Burgos, Martínez-Ortiz, Sierra, & Fernández-Manjón (2008) discuss the

use of the e-Adventure engine and e-Adventure which uses the JavaScript API to develop client server communications between the student's browser and a server based Learning Management System (LMS).

Programmatically Logging Game Play Data

When creating a web-based board game for teaching literacy skills, Markey, Swanson, Jenkins, Jennings, Jean, Rosenberg, & Frost (2008) logged game play data for analysis, "While SI 110 students played the game, the project team logged selected transaction data about game play such as questions attempted and answered correctly by type, licenses purchased by type, and time elapsed since the start of the game. When the game ended, we transferred logged data to an Excel workbook for data analysis" (p. 671). While this article was of interest due to the fact that it was a board game, and the existing NYCAMH game is a board game, the idea of logging game data for future research and game revisions is worth considering to foster the idea of continuous improvement.

Literature Review Summary

For the NYCAMH project to be successful, the game needs to be entertaining. If it fails to motivate and engage players it will not be successful educational tool. How abstract concepts are converted to concrete knowledge will be done by combining input from the NYCAMH, concepts learned in the article list, and reflecting Michael Van Dusen's research. Van Dusen (2012) speaks to the technology adaption of the farm community and keeping the game simple as students may not have had a lot of prior computer use that we have grown to expect in the younger generation. Van Dusen (2012) also looks forward past his research, "In the future I think that more research can be done to see if another type of game can be made to make teaching farm safety a little easier. Some of the types of games that could be looked into are 3D

type games or virtual reality type games. Putting the farmer or game player closer to the game is going to increase the game experience” (p. 60). Finally platform and software framework must be evaluated and decided to best reach the wide audience and achieve the goal of delivering quality farm safety education.

3. Methodology and Design

NYCAMH has a need for the development of a web based educational game for teaching farm safety. This design describes a game that attempts to fulfill the needs of NYCAMH, follow good information design practices, and integrate the elements needed for an effective educational game.

Target Audience

This educational game is targeted toward the adults and children of farm families, the same demographic as the Play It Safe board game. The board game was created to help teachers and 4H leaders teach farm safety to ages 10 to adult. The animated nature of the online RPG game will appeal to this age group. The farm safety information text found on the farm will be adequate for the age group since it was taken directly from NYCAMH’s Play It Safe feedback sheet which is used in the board game. The online game will eventually be integrated into the NYCAMH website and is not part of an instructional program or class. Game participation is recommended but ultimately optional.

Industry Survey

A survey of other farming games was done in order to find anything that could be reused as a component for the NYCAMH game. Farmville, and Agricola were evaluated. There were no reusable components from a technical standpoint such as open source modules that could be

incorporated into the game design. Agricola is more of an online board game and did not fit the contextual learning environment needed to support the constructivist approach being taken. If time permitted playing both games for extended periods I may have found some reusable ideas that could be incorporated into the new design.

A program called *Construct 2* was also evaluated as a tool to create the game. Construct 2 is a completely GUI driven game development platform that enables web based games with HTML 5 animations to be created without writing a single line of code. By using Construct 2 it would not have been necessary to write any HTML, CSS, PHP, or JavaScript.

A learning curve was anticipated with Construct 2 and there was a concern about the possibility of running into limitations using the tool. However the idea of having a GUI driven tool to create the HTML 5 animations was intriguing and I had considered using Construct 2 for the sole purpose of generating the animations. After further research, I had found that Adobe[®] makes a free GUI driven tool called *Edge Animate* for creating HTML 5 animations. Adobe[®] Edge Animate was the perfect fit for what was need for this project.

Development Blog

A development blog was created to document the design process. Google Sites was used as the platform to host the blog. The URL for the blog is

<https://sites.google.com/a/sunyit.edu/nycamh-safety-game-blog/>.

Game Concept and Overview

The NYCAMH farm safety game project is a web based educational game. The game is in the style of an adventure or role playing game (RPG). Van Dusen's (2012) study indicated that an RPG style game would be beneficial. "Some of the types of games that could be looked into are 3D type games or virtual reality type games. Putting the farmer or game player closer to

the game is going to increase the game experience” (p. 60). The main character that the player controls is a farmer. The player is able to move the farmer around a farm using the arrow keys or the mouse. The farm is laid out on a large canvas. The canvas scrolls beneath HTML frame within the user’s browser window much like maps do on popular map websites.

The farmer has a list of tasks to be done around the farm. The game objective is to complete all of the items on the task list. However the farmer must take certain safety precautions before performing each task by way of obtaining certain items of inventory. For instance he must have leather boots before mowing the grass.

The player has the ability to pick up inventory items, display the current inventory, and drop inventory items. Games can be saved and resumed, and a high score list can be displayed.

Figure 1 below illustrates a typical game play flow.

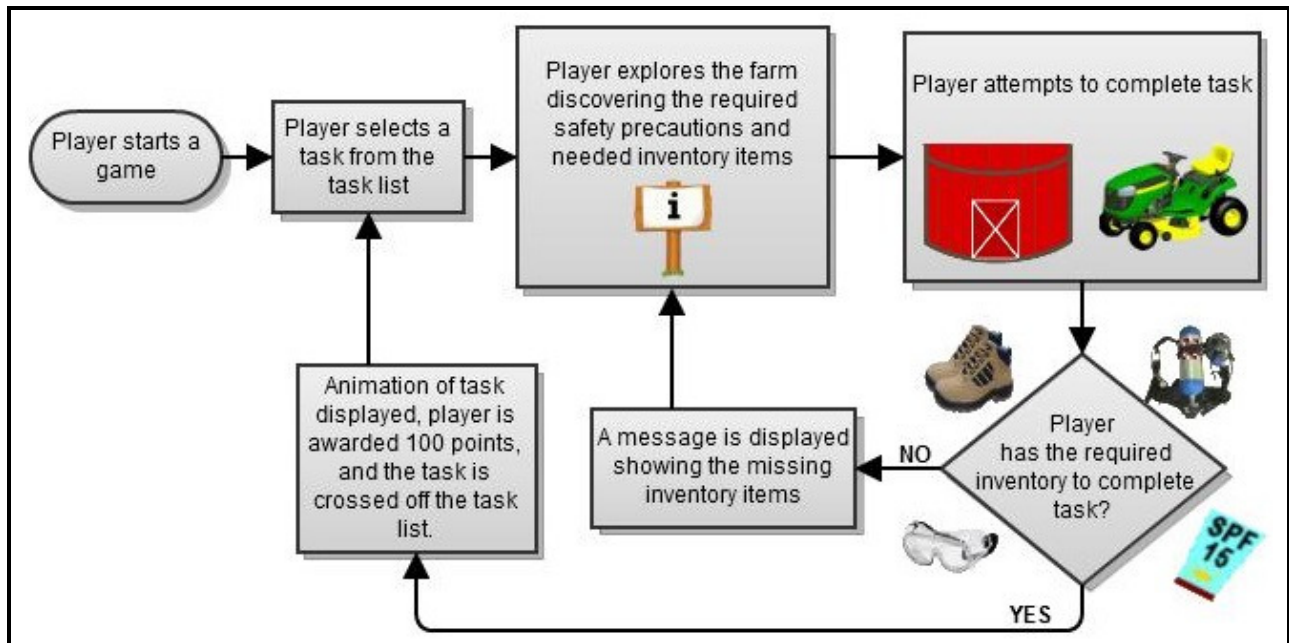


Figure 1: A Typical “Play It Safe” Game Play Flow

Screen shots depicting what the player sees during this journey are in Appendix A.

Educational Objectives

Game play does not unfold in a linear fashion; the player has complete control of where he travels on the farm and which tasks he completes first. A constructivist educational approach is used where the problems are given to the players to solve on their own. The player must move the farmer around the farm and obtain the information and inventory items needed to complete the jobs on the task list. The player is put in the context of real farmer to be played out in an animated world.

The game must be entertaining to engage and motivate players. The current game design has been prototyped and feedback from NYCAMH has been received that would indicate while we are going in the right direction, there are specific changes needed to better engage younger players. An example of this was the recommendation to replace the informational sign posts with people the player can interact with.

As the player moves around the farm there is farm safety information scattered around the farm that will help to make informed choices when completing items on the task list. For now the farm safety information is contained on wooden signs (Figure 2).



Figure 2: Information Sign Post

There are also question sign posts where players can answer multiple choice questions on farm safety for additional points. Answering these questions is not required to complete the items on the task list. These question signs are used as a stop gap to fill voids where a contextual learning scenario has not yet been added to the game that covers a specific safety fact. It is a design goal that these signs eventually be eliminated from the game. In the mean time it will be the competitive players who will seek out the correct answers to these questions for the few extra

points they will receive.



Figure 3: Question Sign Post

The Play It Safe feedback sheet states that 40% of farmers will suffer a serious accident in their lifetime. The information in the feedback sheet addresses the causes of these incidents and while the objective of the game is to educate players through entertainment, a measurable expectation would be that a player that successfully completes all items on the task list will have learned at least 3 farm safety facts from each of the 6 categories in feedback sheet. Expanding the audience of the game and meeting objectives will help reduce this overwhelming 40% statistic. NYCAMH does not have specific objectives for the game but sees it as a way to raise awareness, and one small aspect of effecting changes in behavior. They would like the game to focus on highlighting the top injury trends.

Integration of NYCAMH Educational Elements

Educational elements are derived from a NYCAMH document called the *Play It Safe Feedback Sheet*. From this spreadsheet farm scenarios can be created with inventory items, and new items on the farmer's task list. There are 120 questions on the spreadsheet and are broken down into 6 categories of 20 questions each (Figure 4). The questions in each category go from least difficult to most difficult.

Category	Play It Safe Color Code	Number of Questions
Clothing / Ear & Eye Protection	Orange	20
Exposure to elements, sunburn, frostbite	Purple	20
Disease and Injuries from Animals	Pink	20
Physical Hazards, Home Accidents, CPR	Green	20
Fire, Chemical Hazards	Yellow	20
Farm Equipment	Blue	20

Figure 4: NYCAMH Play It Safe Feedback Sheet Category Breakdown

The goal is to make all elements in the spreadsheet learnable through contextual scenarios of farm life played out through the game. This may not be possible all of the time depending on the question, so the ability to pose multiple choice questions has been integrated into the game design.

An Agile Development Process

An agile development process is one where features are delivered to the customer as they are developed. So the customer is evaluating software that is not yet fully functional. This process gives them an opportunity to test and provide feedback to developers. Each delivery to the customer is bigger than the last as features are added. The agile process includes the customer in the design process to insure the desired product created. Figure 5 below illustrates this iterative feature delivery/feedback process.

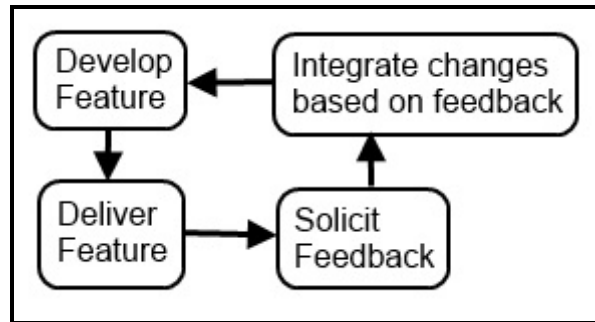


Figure 5: The Agile Development Process

The game project has been done as a continuous series of prototypes for NYCAMH to review. Prototypes can be small individual game elements, or larger working portions of the game. The goal here was to keep NYCAMH members involved in the design and development of the game. NYCAMH could optionally show the larger working prototypes to farm families and other groups to collect additional feedback. Another goal to keep the design under constant review to prevent large redesigns due to differences between the developer and NYCAMH expectations. NYCAMH Prototypes can be found in directories under the URL:

<http://www.cs.sunyit.edu/~begleyr/nycamh/>.

Theoretical Foundation for Game Design

A sequential online farm safety course where a student reads an excerpt from course material, or watches an instructional video, and then proceeds into an evaluation section of test questions would be an extremely straightforward and a relatively quick design. Additionally games that follow the same sequential process of teach and quiz would also make for a quick design. NYCAMH's Play It Safe board game follows this type of process.

However the review of literature has indicated that routine instruction can become boring causing the learning process fails, where Lin, et al (2011) discuss a game-like learning environment will make the activity enjoyable due to the interactivity, unpredictability, and generated curiosity. An RPG game becomes a powerful learning tool because it places the

player in the context of the environment he is learning about. Shaffer, et al (2005) discuss how these virtual, contextual worlds create situated understandings, and learning. The game must be entertaining to be an effective learning tool. Fernández-Manjón (2008) discusses how an educational game that fails to entertain loses the advantages of game based learning.

Based on these theories, the online Play It Safe game design has been done as an RPG game with the main character in a virtual world and the learning embedded within the entertainment. The farmer's task list is used to create situational or contextual learning experiences. Early NYCAMH feedback has also suggested adding random events to the game like fires which increases the unpredictability of the game play, the player's curiosity, the entertainment value, and therefore the educational advantages.

Van Dusen (2012), who worked directly with the target audience, found through a survey and data analysis that his game prototype stimulated motivation with the interviewees in the survey. He also believed that a virtual reality game would further increase the game experience. His work directly with farm families, surveys, prototyping, and data analysis has influenced the direction of the Play It Safe game design.

Development Tools

A server was needed that supported both PHP and MySQL and the Computer Science SUNYIT site worked fine for developing and delivering prototypes to NYCAMH.

Notepad++ was used for developing source code. It also has a built in FTP client allowing for quick edits and changes to be pushed up to the server quickly. The FileZilla FTP client was also used for file transfers that notepad++ was not well suited for, like graphic files, and large transfers of multiple files and directories.

Firebug is a Firefox plugin that facilitates the debugging of software running on the client

machine. Firebug allows for JavaScript debug messages to be sent to a console window while the game is being played. Firebug allows the developer to examine the structure of the HTML and CSS at anytime. Firebug greatly accelerates the debug of new code.

Adobe® Edge Animate is a GUI driven tool that allows anybody to create HTML 5 animations. Once an animation is published everything you need to drop into the game is generated including the animation’s own top level HTML file.

4. Game Design Framework

A major design goal of this project was to create a framework that can be built upon and added to. The game design involves several different web technologies which include HTML, CSS, JavaScript, JQuery, PHP, mySQL, and AJAX. A high level view of the game design is illustrated in Figure 6.

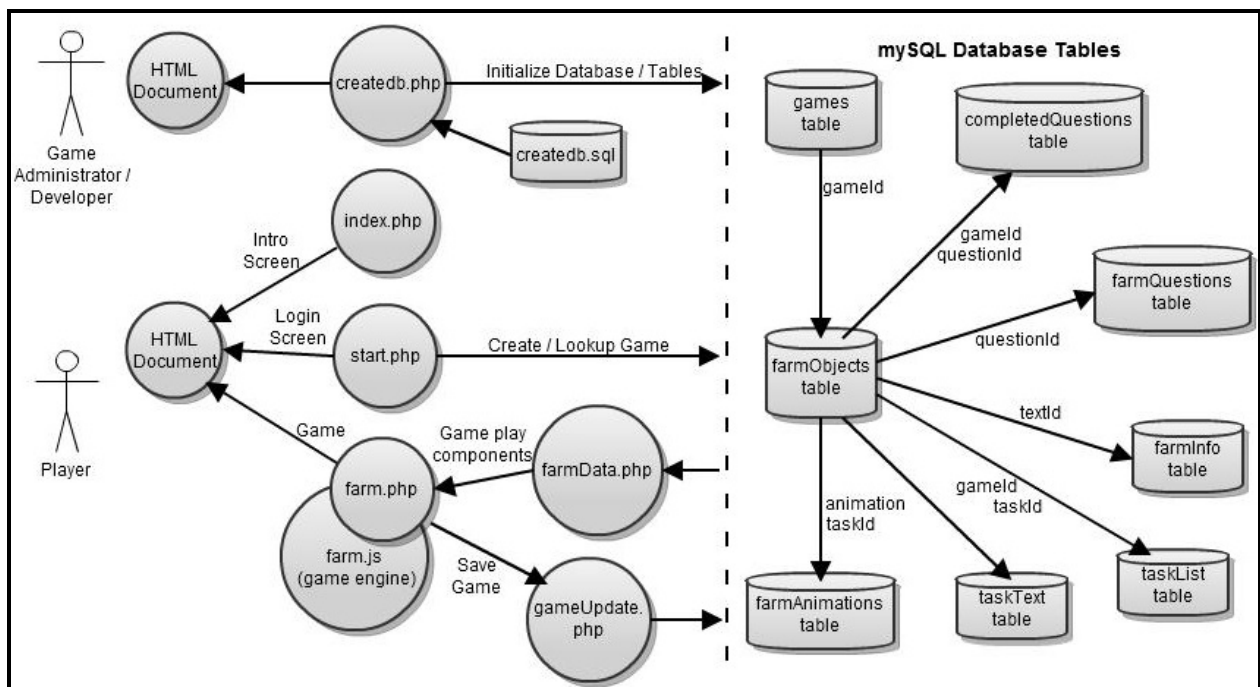


Figure 6: NYCAMH Play It Safe Game Design

Since the game runs in a browser, the game is an HTML document. All components of

the game are stored in a mySQL database on the web server. These components are read from the database by PHP scripts and feed them to the browser as an HTML page. A single JavaScript file (farm.js – refer to Appendix A.) is included as part of the HTML page and contains the game engine. The game engine is what allows the player to move the farmer around the farm, pick up items, and launch animations.

The Farm Canvas and Window Frame

The canvas refers to the entire 2 dimensional space allocated for the farm. This space is much larger than what can be viewed in the player's browser window. A window frame was created using CSS. This frame allows the player to view a portion of the canvas which lies below. As the player navigates the farmer around the farm the canvas will automatically scroll below this fixed frame (Figure 7).

The canvas itself is a single background image that repeats to completely fill the dimensions of the entire farm. All other pictures and graphics are added to the farm by the game engine.

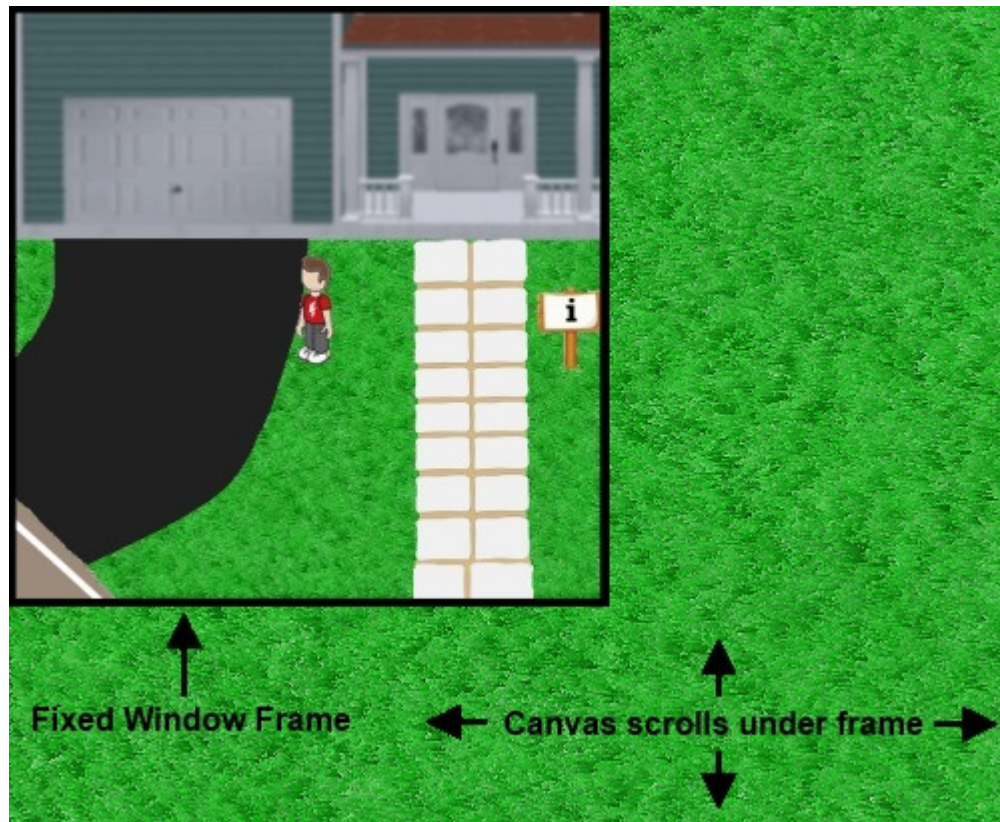


Figure 7: Canvas scrolls under CSS frame in browser window

Animating the Farmer

The most important graphic object placed on the canvas is the main character or farmer.

The farmer graphic is an image sprite that contains 12 different poses that are used for animation purposes. These poses are:

- Front facing standing
- Front facing walking left foot forward, right hand forward
- Front facing walking right foot forward, left hand forward
- Rear facing standing
- Rear facing walking left foot forward, right hand forward
- Rear facing walking right foot forward, left hand forward
- Left facing standing
- Left facing walking left foot forward, right hand forward
- Left facing walking right foot forward, left hand forward
- Right facing standing
- Right facing walking left foot forward, right hand forward
- Right facing walking right foot forward, left hand forward

Using CSS, JavaScript, and JQuery, the farmer can be animated to walk by specifying the coordinates within the image sprite file where each pose exists upon the use of arrow keys or mouse clicks. The image sprite is used so that the file is only loaded once into memory to prevent disk accesses as the animation moves from pose to pose in a walking sequence.

The current image sprite is a prototype. The Van Dusen (2012) research, and input from NYCAMH indicates the farmer character needs to be non-stereotypical farmer and “should stress all forms of diversity” (p. 38). A male farmer is currently being designed by professional graphic artist. As shown in Figure 8, he has proposed two different images to NYCAMH and they have accepted one of those images for the male farmer.



Figure 8: NYCAMH Approved Male Farmer Image

Once the 12 poses are delivered for the male farmer a proposal will be made for a female farmer. The game design will be modified to allow for player preference to select between male and female farmers. Optimally a selection of race, hair color, and wardrobe would be advantageous as Yucel, et al (2006), and other studies have found this type of user control (or modding) has been shown to contribute to learning. A single image of each farmer character will be designed and approved by NYCAMH before an entire sprite is designed and delivered.

Farm Object Types

In addition to the farmer there are many other types of objects placed on the farm canvas. The list of object types is expected to expand more as different types of farming scenarios are designed into the game. The current object types are as follows:

- Character (farmer)
- Static
- Inventory
- Animation
- Debris
- Other Static
- Information
- Question

The *static* object type is used for images to be placed on the canvas that do not have any interaction with the game. Since the canvas itself is a single color background image, the static objects help paint picture of the farm. The *inventory* object type is used for objects that can be picked up, dropped, and needed to complete farm tasks safely. When an *animation* object type is clicked on it can launch an HTML 5 animation provided the farmer has taken all of the appropriate safety precautions. *Debris* object types will be removed from the game if clicked on. The *other static* object types are static types that were previously an animation. This type reassignment is done to prevent the animation from being run again. The *information* object type is used to display safety information and the *question* object type for launching a multiple choice question.

Other Farm Object Attributes

In addition to an “object type”, farm objects have other important attributes. All objects have a unique name that is used for CSS purposes, a filename that contains the graphic for the object, and some have an associated text string to be displayed when the mouse hovers over the

object. All objects have coordinate information including X and Y coordinates of where the item is to be placed on the canvas. There are currently 17 different attributes for an object. However some are specific to the type of object being defined.

All of these objects that make up the farm are stored in the mySQL database table called “farmObjects”. Because players can move objects around, remove objects, and complete animations, each game has its own set of farm objects in the farmObjects table. This ensures that when a player comes back to resume a game already in progress, his farmer character is in the same place as where he left off and all other farm objects are in the same place and state they were when the game was saved.

Passing Game Components Down to the HTML Document

In addition to farm objects, there are many other tables stored in the mySQL database.

Here is the list of tables:

- games table
- farmObjects table
- completedQuestions table
- farmQuestions table
- farmInfo table
- taskList table
- taskText table
- farmAnimations table

The information in these tables must be loaded into the HTML document when a game is started or resumed so they can be utilized by the game engine. From a mySQL table, the data is read into a PHP object and then written into the HTML stream as a JavaScript multidimensional array. Each of these arrays (one for each table listed above) are predefined in the game engine (farm.js – refer to Appendix A.) and then initialized within HTML document sent to the browser (farm.php – refer to Appendix A.). This path that the data takes from mySQL table to the HTML

document is illustrated below in Figure 9.

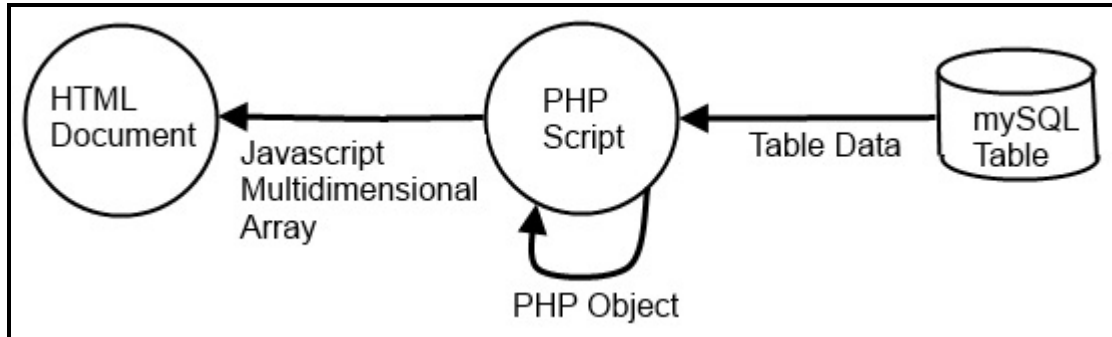


Figure 9: mySQL table data delivered to HTML Document

When the HTML page is loaded in the player's browser all game components are already on the client side and the game is ready to be played without any interactions with the server.

Saving Games Using AJAX

AJAX (Asynchronous JavaScript and XML) is a technology that allows the client side to send data to the server and retrieve data from the server without reloading the HTML document. It is with this technology the save game feature is implemented. During game play there are scoring changes, and state and location changes of farm objects. These changes are maintained by the game engine in the same multidimensional arrays that were loaded with the HTML page when the game was started.

When a player requests that the game be saved, all of the non static information for that game is updated on the server using AJAX calls to the updateGame.php script (refer to Figure 6, and Appendix A.). If a player does not save the game, the next time the game is loaded it will be loaded to last saved state. If the game was never previously saved, it would load to the initial state the game was in when it was created.

These AJAX calls can be made at anytime without the player's knowledge, so this technology can also be utilized to log game play information for research purposes.

Animations and Adobe® Edge Animate

Animations are integral to the game concept and design. Currently animations are used when a player successfully completes a task on the farm. There is a lot more that can be done with animations to enhance the entertainment and educational value of the game. The animations in the latest version of the project were done with Adobe® Edge Animate. In Figure 10 below you can see the silo animation being created with the design of a moving door.

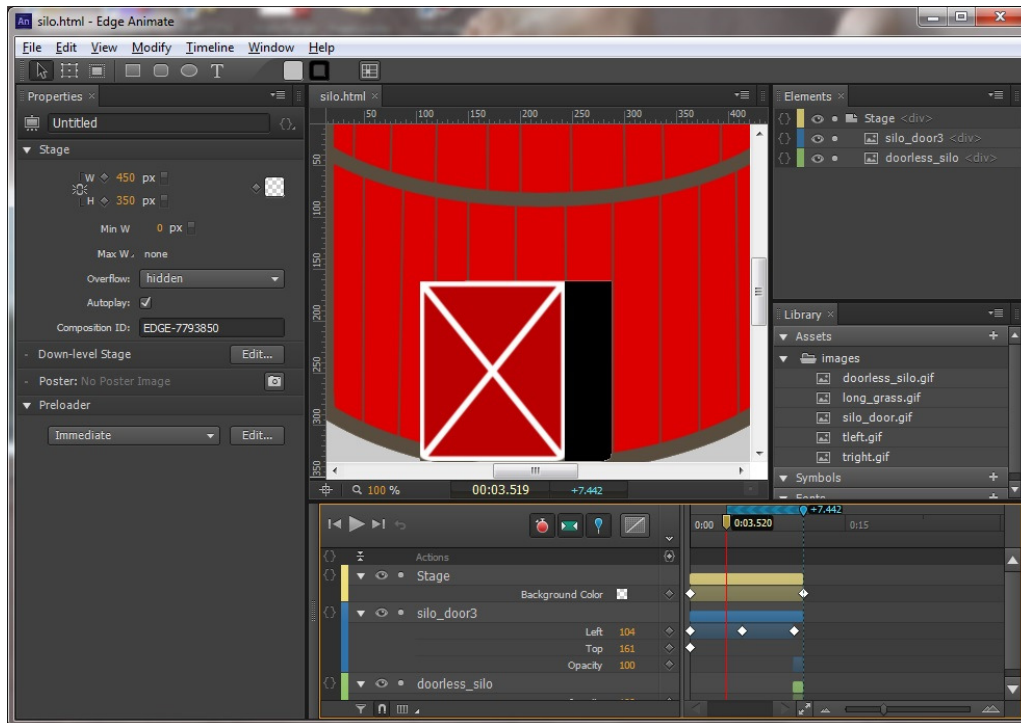


Figure 10: Creating Animations with Adobe® Edge Animate

Once an animation is completed, it can be integrated into the game solely with a database update. The location on the canvas, size, and name of the animation will be fed down to the game engine and the animation will be launched at the appropriate time without changing a line of code.

Initialization of Database Tables

The entire farm and all saved games are stored in a mySQL database. The creation and

initialization of these tables is done with a SQL script. The script is destructive in that it removes all tables and the contents of those tables from the database and recreates and repopulates those tables. All saved games are permanently removed.

The name of the script is `createdb.sql` (refer to Appendix A). The script can be run from the Linux command line by redirecting the contents of the file into the `mysql` executable:

```
mysql -h db -u begleyr -p < create_database.sql
```

The script can also be utilized using a PHP script called `createdb.php` (refer to Figure 6, and Appendix A.). The `createdb.php` runs in a browser window. It opens the `createdb.sql` file and executes each command individually. This is helpful for development on Linux servers where you are not granted command line access.

5. Anticipations and Limitations

Small prototypes that demonstrated player movement, and the ability to pick up inventory items were delivered to NYCAMH fairly early in the process, however only the last prototype could be considered a fully functioning game and therefore received the most feedback. The anticipation was that there would more than one delivery of this nature that would incorporate some of the suggestions into successive deliveries before the writing of this paper. Because of the lateness of the deliver of this last prototype it has reached a limited audience.

Only two items on the farmer's task list have been completed thus far, and I had anticipated having five or more done. The limiting factor at this point would be a lack of skill in graphic arts and animation.

Currently the game is still using a temporary image sprite for the farmer. I had anticipated having the male farmer fully implemented but I am still waiting on the delivery of the 12 poses needed. I had also anticipated that a prototype of the female farmer would have been

delivered for NYCAMH review. Communication problems with the first graphics artist made it impossible to continue and a professional artist was brought on board late in the process.

A big limiting factor has been time. An early goal was to complete a framework that could be built on after this initial project was completed. While the framework is in a working state, there will be the need for expansion, adjustments, and bug fixes.

Graphics availability has been a limited progress. This is truly a multidiscipline project blending information design, software design, and graphic arts and therefore requires a team of people to keep the process moving forward. Many graphics used thus far in the development are crude and in need of replacement but have been sufficient for demonstration purposes.

Since this was a project to solely focus on the design of a game, there has not been any study to measure the effectiveness of the game as a learning tool. If studies are done on the effectiveness of the game, the results of those studies could be integrated into the agile process as feedback.

6. Findings and Results

Games are an excellent way to facilitate learning. A constructivist approach creating a conceptual learning environment with an RPG game blends learning and entertainment. Play It Safe is designed to have the player live the life of a farmer, performing tasks, and taking safety precautions. Following Gee's (2005) risk taking principle the player cannot die during game play which encourages the player to tackle the farm tasks without the fear of the game ending prematurely.

Play It Safe is a virtual world where players can take any path through the game they desire following principles described by Kafai (2006) and others which state that allowing players to create or design their own path through a game engages players and enhances learning.

Agile Development and Educational Game Development

The agile process is the preferred process for the development for educational games. There are typically multiple areas of expertise needed for any software development process and educational games are no different. The development of Play It Safe required knowledge of web development technologies and languages, graphics and animation creation, farm safety educational content, and the various principles of developing educational games discussed in the literature review. NYCAMH's Play It Safe Feedback spreadsheet was an excellent resource for educational content requirements, but without their constant feedback development could have easily moved in the wrong direction. The reason for this is that NYCAMH has familiarity with the farmers and their families and without that insight and feedback on prototypes the project may engage and entertain the target audience, farm families.

Test and Evaluation

NYCAMH has provided continuous feedback as prototypes were delivered. The last prototype delivered was the largest set of features and almost a fully functional game. Their comments were as follows:

- Would this be a full screen image? The small screen is annoying to navigate if I don't have a full picture (bird's eye view). I also think there needs to be more explanations. As an introduction of what needs to be done before you start. Or have the task list pop up rather than it just being in the menu tab.
- More direction, a welcome screen or explain what we're supposed to be doing/ make the task list more visible.
- Add a map/view of the entire layout in the corner of your game screen, so you don't just travel blindly in areas.
- Maybe rather than the "I" signs, add a person to talk to (parent/grandparent). We had a pre-teen play with this, and a comment that was made was she will have conversations with people to get additional information, but wouldn't just read the sign.
- It's difficult to pick things up, we think you have to be directly in front of the object, but tried 4-5 times before getting it.
- Another idea that came up would be adding catastrophic events that could happen on

the farm, for example a flood, or a fire on the farm. It just makes things more exciting from a younger child's perspective.

Some of this feedback can be addressed quickly and other feedback will require more design work and follow on projects. It is important though that an agile development process continues with future work and constant feedback is channeled back into the design work.

The effectiveness of Play It Safe as a learning tool is unknown at this point since it has not been published on NYCAMH's website and tested with the target audience. As an ongoing agile process, future iterations of the game design should be tested with a sample group and have feedback channeled back to generate design changes. After playing the game, the sample group would be asked to answer several questions directly from the Play It Back feedback sheet. It is this data from the feedback sheet that has been designed into the contextual aspects of the virtual world of the Play It Safe game.

A comparison between sample groups who have played the board game, groups who have played the online game, and groups that have played both would yield interesting data to qualitatively measure the effectiveness of the game. It would be beneficial to have these tests done sooner rather than later to stay true to an agile development process and create the most effective educational game possible.

Further Study and Game Enhancements

The *Play It Safe* game is a large project. There are over 180 items in NYCAMH's Play It Safe Feedback sheet that need to be worked into the game.

HTML 5 animations for this game were created with Adobe® Edge Animate. The animations can be dropped into the game without code changes. This area of the project would suit a student who has a strong interest in animation, particularly HTML 5 animations and has skills in graphic arts.

Database Administration at this point in the project the only way to add elements to the game is through a mySQL script that initializes the database. Each time a new element is added to the game, or repositioned, the database needs to be regenerated. Additional PHP code could be written to help with adding items to the database and adjusting their locations.

Expansion of **personal preferences** would further increase player engagement. The current design has a single fixed farmer character. There are currently plans to have a selection for a female farmer character. However further character preferences like hair color and clothing choices would further increase player engagement.

Game expansion into different canvases would make the game multidimensional. The current game design has a single canvas which is the farm. An example of another canvas would be the inside of a barn with different rooms. The game is already architected to support different canvases so the majority of the work would be the creation of graphics.

Facebook integration would increase the visibility of the game to more players. Competition increases player motivation and the integration into a social media site will create a community of players.

A **wayfinding** project would make an interesting project to address early NYCAMH feedback that has indicated that an area map of the player's current location would be highly desirable. The map would show a miniature view of the entire canvas. The project would require some additional code as the current game architecture is not setup for this type of feature.

A project to accommodate different **hardware platforms** would insure the entertainment value would carry over to tablets and mobile devices. While the game is playable on smaller devices, enhancements to dynamically adapt to different platforms to maximize the game playing experience will increase the accessibility of the game to more players.

A **qualitative research** project would measure the games effectiveness as a learning tool. This type of project could couple data collected from interviews with people who have played the game with data collected from game activity logs. Logs could be generated in files stored on the server, or in mySQL tables. Log generation would require the creation of additional software that would utilize the AJAX technology which allows the client side JavaScript code to write data to the server in the background while the game is being played. This study should also address how the results of the study would effect changes to the on going agile development process.

7. Conclusion

NYCAMH's Play It Safe board game was created to raise awareness in farm safety. The game is a question and answer game with multiple choice answers. The game needs to be setup and have multiple players. Information retention is questionable in this type of format, as is its ability to engage players in the subject matter. The online Play It Safe game takes a more effective approach. The game is online and does not require any setup or software installation other than an operating system and a browser. The game is a single player game and can be played asynchronously using save and resume game features. Information retention is maximized as the player lives the life of a farmer in a virtual world taking true to life safety precautions. The RPG style, animations, and given challenges found on the farm will engage and entertain players as they perform farm tasks and discover the needed safety precautions on their own.

In order to create an educational game that follows a constructivist approach with a contextual learning style requires a full range of disciplines and hard work. The internet and web technologies allow for the game to be accessible. The tools and technologies are out there to

make this type of project successful. Newer tools, like Construct 2, are making game development easier and virtually eliminating the need for writing code.

The alternative approach of a game that sequentially teaches, and quizzes would be much easier to implement, but the review of literature would suggest that it would not be as entertaining, engaging, and therefore ineffective as a learning tool. The goal is to build a game that people want to play not because it teaches them about farm safety, but because it is entertaining. The learning is built in to the entertainment.

The NYCAMH project has been an aggressive project to make a virtual world, a true contextual learning environment where the player has complete control of the sequence of events. Although the framework of the game is complete, early feedback suggests that enhancements will be made for an area map and spontaneous farm events.

Once NYCAMH's immediate concerns are addressed, and 3 or 4 more farm tasks are added to the game, an evaluation phase can begin to test the game with the target demographic in a qualitative study to assess the game's entertainment, and educational value. Results and comments feedback through the study can be incorporated into the iterative agile process to initiate design enhancements.

Working with NYCAMH organization has been a pleasure as they are pleasant and responsive to requests. All of the educational aspects of farm safety are nicely summarized in NYCAMH's Play It Safe Feedback Sheet. This is extremely helpful for the game developers because all of the requirements for the game are contained in a single spreadsheet. There is the possibility of multiple projects being started from the work done on this project and to have a single document that contains the game requirements that all people can work off of is invaluable.

This project has highlighted the need for game developers to know their audience, and develop in an agile process where feedback can be incorporate back into the design. There were many emails and meetings with NYCAMH during this project all of which help guide the design to a desired outcome. This is especially true for developers who are attempting to incorporate learning into a game as they would not necessarily be the subject matter expert on the educational topic. NYCAMH has made this process much easier by developing a spread sheet that outlines all of the educational topics. The design of a contextual learning environment, in this case a farm, requires the supervision and guidance of people who understand this space and culture. While a game developer may succeed in creating an entertaining game, without this type of team approach, and iterative/agile process, a game design is likely to miss its mark from an educational standpoint.

References

- Brom, C., Šisler, V., & Slavík, R. (2010). Implementing digital game-based learning in schools: augmented learning environment of 'Europe 2045'. *Multimedia Systems*, 16(1), 23-41.
doi:10.1007/s00530-009-0174-0
- Gee, J. (2005). Good Video Games and Good Learning. *Phi Kappa Phi Forum*, 85(2), 33-37.
- Juracz, L. (2010). DEVELOPING COURSES WITH HOLORENA, A FRAMEWORK FOR SCENARIO- AND GAME BASED E-LEARNING ENVIRONMENTS. *International Journal Of Software Engineering & Applications*, 1(4), 1-17.
doi:10.5121/ijsea.2010.1401
- Kafai, Y. B. (2006). Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and Culture*, 1(1), 36-40.
- Lin, K., Wu, T., & Wang, Y. (2011). Developing a Web-based and Competition-based Quiz Game Environment to Improve Student Motivation. *Journal Of Networks*, 6(5), 736-742.
doi:10.4304/jnw.6.5.736-742
- Markey, K., Swanson, F., Jenkins, A., Jennings, B. J., Jean, B. t., Rosenberg, V., & ... Frost, R. L. (2008). Designing and testing a web-based board game for teaching information literacy skills and concepts. *Library Hi Tech*, 26(4), 663-681.
- Moreno-Ger, P., Burgos, D., Martínez-Ortiz, I., Sierra, J., & Fernández-Manjón, B. (2008). Educational game design for online education. *Computers In Human Behavior*, 24(6), 2530-2540. doi:10.1016/j.chb.2008.03.012
- Nousiainen, T., & Kankaanranta, M. (2009). Exploring Children's Requirements for Game-Based Learning Environments. *Advances In Human-Computer Interaction*, 1-7.
doi:10.1155/2008/284056

- Papastergiou, M. (2008). Online Computer Games as Collaborative Learning Environments: Prospects and Challenges for Tertiary Education. *Journal Of Educational Technology Systems*, 37(1), 19-38. doi:10.2190/ET.37.1.c
- Rankin, J. R., Vargas, S., & Taylor, P. F. (2009). Testing Metaphorical Educational FPS Games. *International Journal Of Computer Games Technology*, 1-5. doi:10.1155/2009/456763
- Shaffer, D., Squire, K. R., Halverson, R., & Gee, J. P. (2005). Video Games and The Future of Learning. (cover story). *Phi Delta Kappan*, 87(2), 105-111.
- Squire, K.D., Jan, M., Matthews, J., Wagler, M., Martin, J., Devane, B. & Holden, C. (2007). Wherever You Go, There You Are: Placed-Based Augmented Reality Games for Learning. In B. E. Sheldon & D. Wiley (Eds). *The design and use of simulation computer games in education*, (265-294). Rotterdam, Netherlands: Sense Publishing.
- Súilleabháin, G. Ó., Walsh, P., & Sime, J. (2009). The Role of Games in Facilitating Preparation for Future Learning. *Proceedings Of The European Conference On Games Based Learning*, 294-302.
- Torrente, J., Moreno-Ger, P., Martínez-Ortiz, I., & Fernandez-Manjon, B. (2009). Integration and Deployment of Educational Games in e-Learning Environments: The Learning Object Model Meets Educational Gaming. *Journal Of Educational Technology & Society*, 12(4), 359-371.
- Van Dusen, M. (2012). *GAMES THAT CAN SAVE LIVES*, SUNYIT Thesis Paper.
- Wideman, H. H., Owston, R. D., Brown, C., Kushniruk, A., Ho, F., & Pitts, K. C. (2007). Unpacking the potential of educational gaming: A new tool for gaming research. *Simulation & Gaming*, 38(1), 9-29. doi:10.1177/1046878106297650

Yucel, I., Zupko, J., Seif El-Nasr, M., (2006) "IT education, girls and game modding",
Interactive Technology and Smart Education, Vol. 3 Iss: 2, pp.143 - 156

Appendix A. Play It Safe Screen Shots



Figure 11: Attempting task without necessary inventory



Figure 12: Reading information sign post by silo

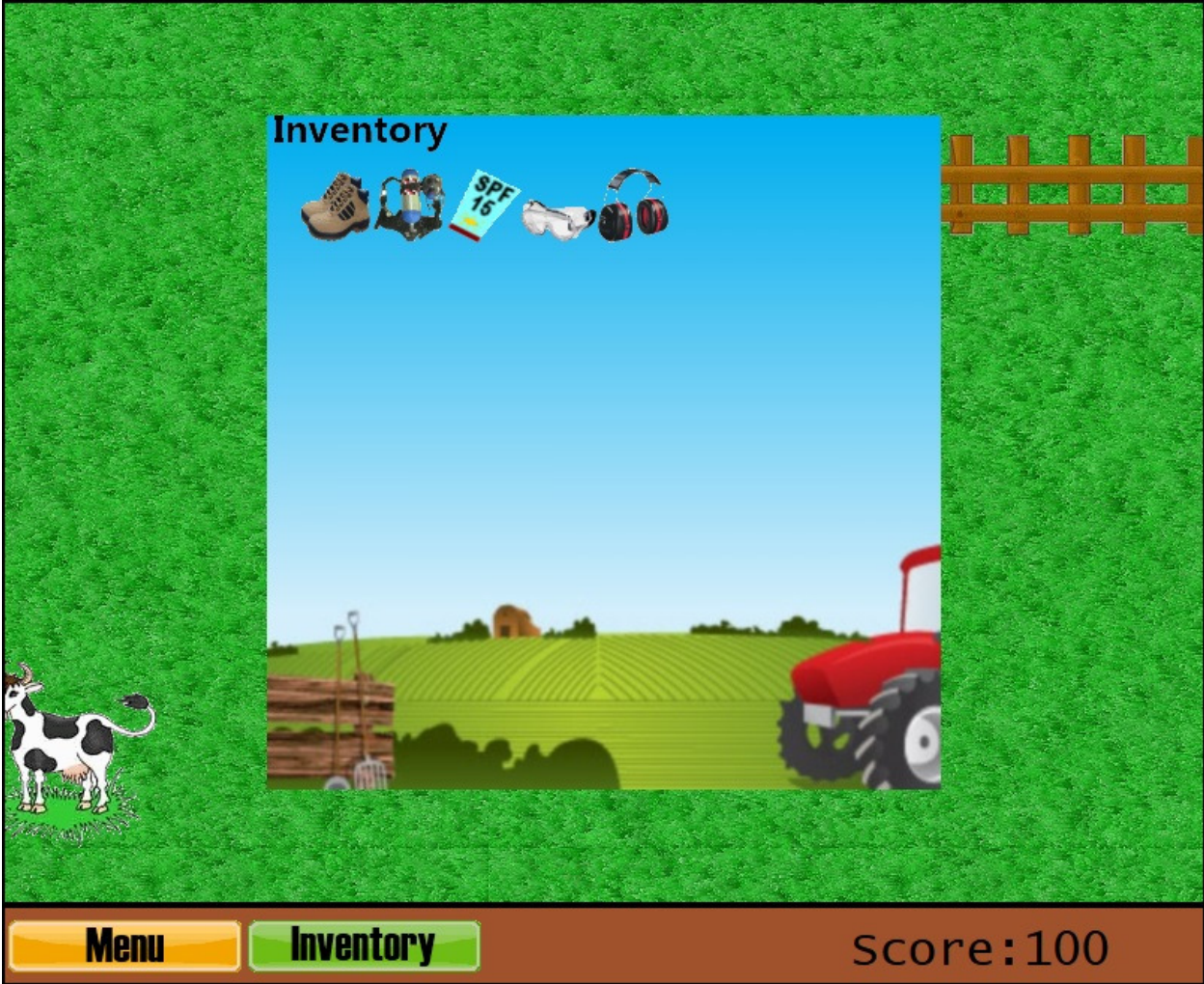


Figure 13: Checking inventory list

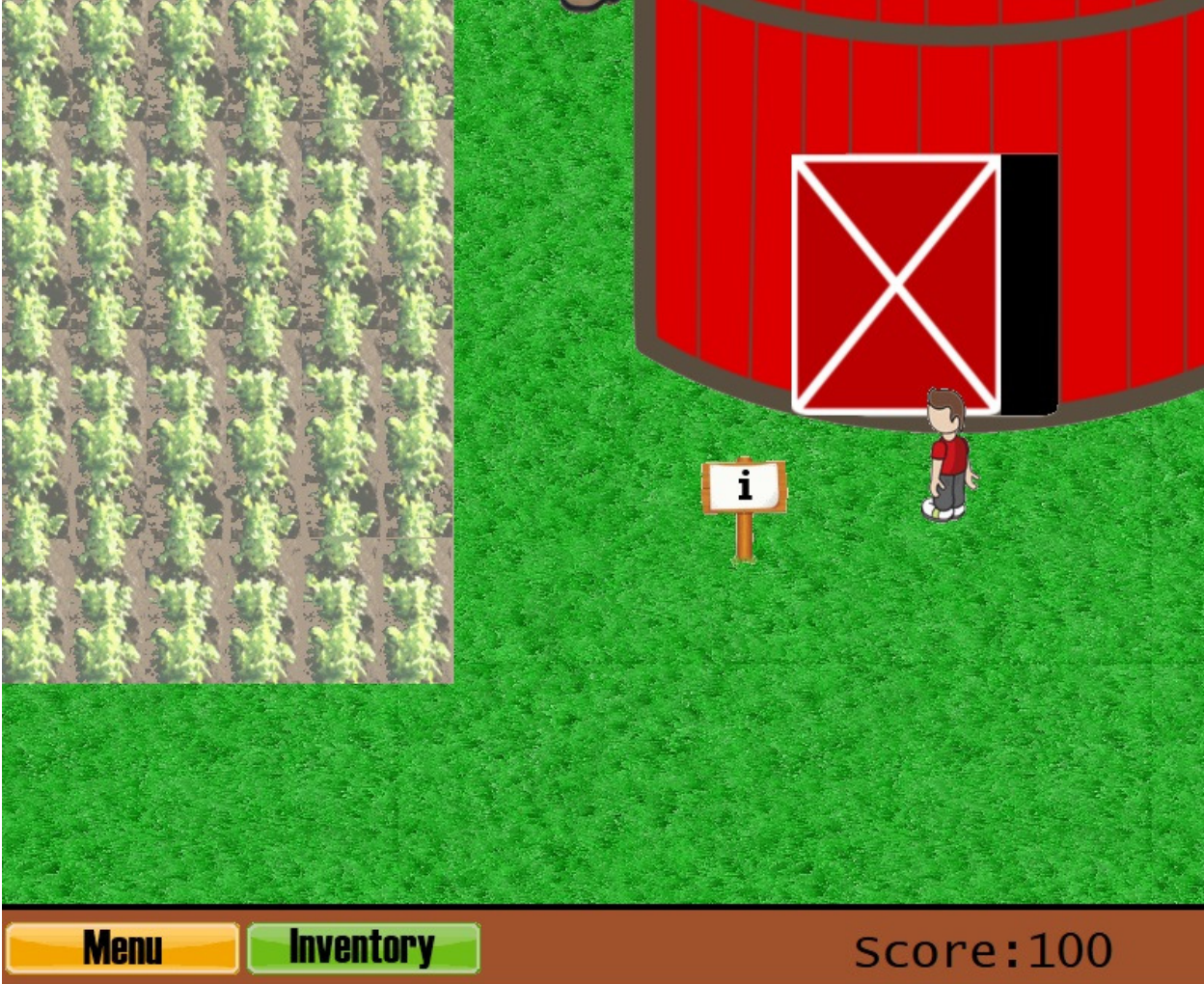


Figure 14: Door opens during silo animation

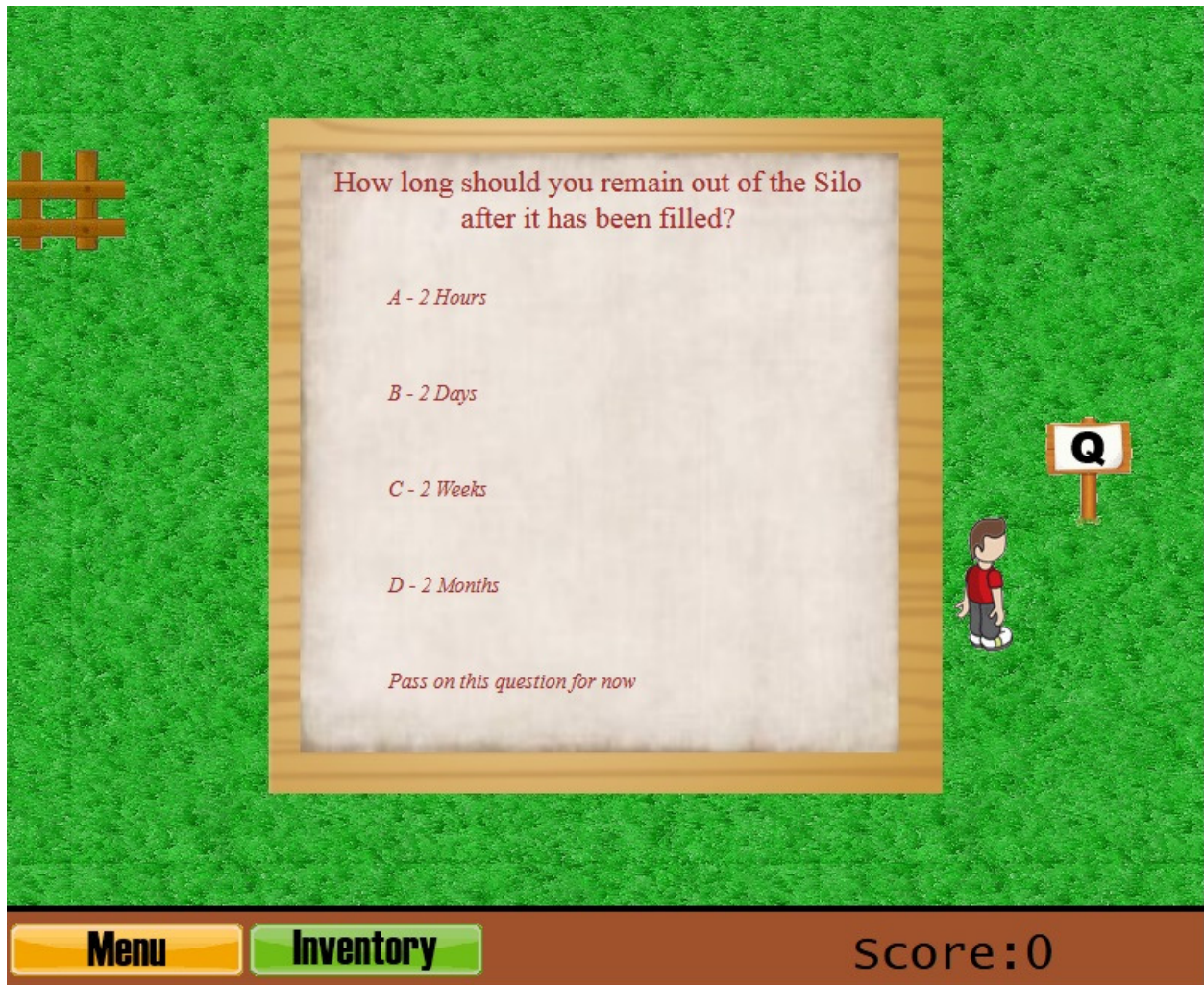


Figure 15: Player attempts question about silo



Figure 16: High score lists

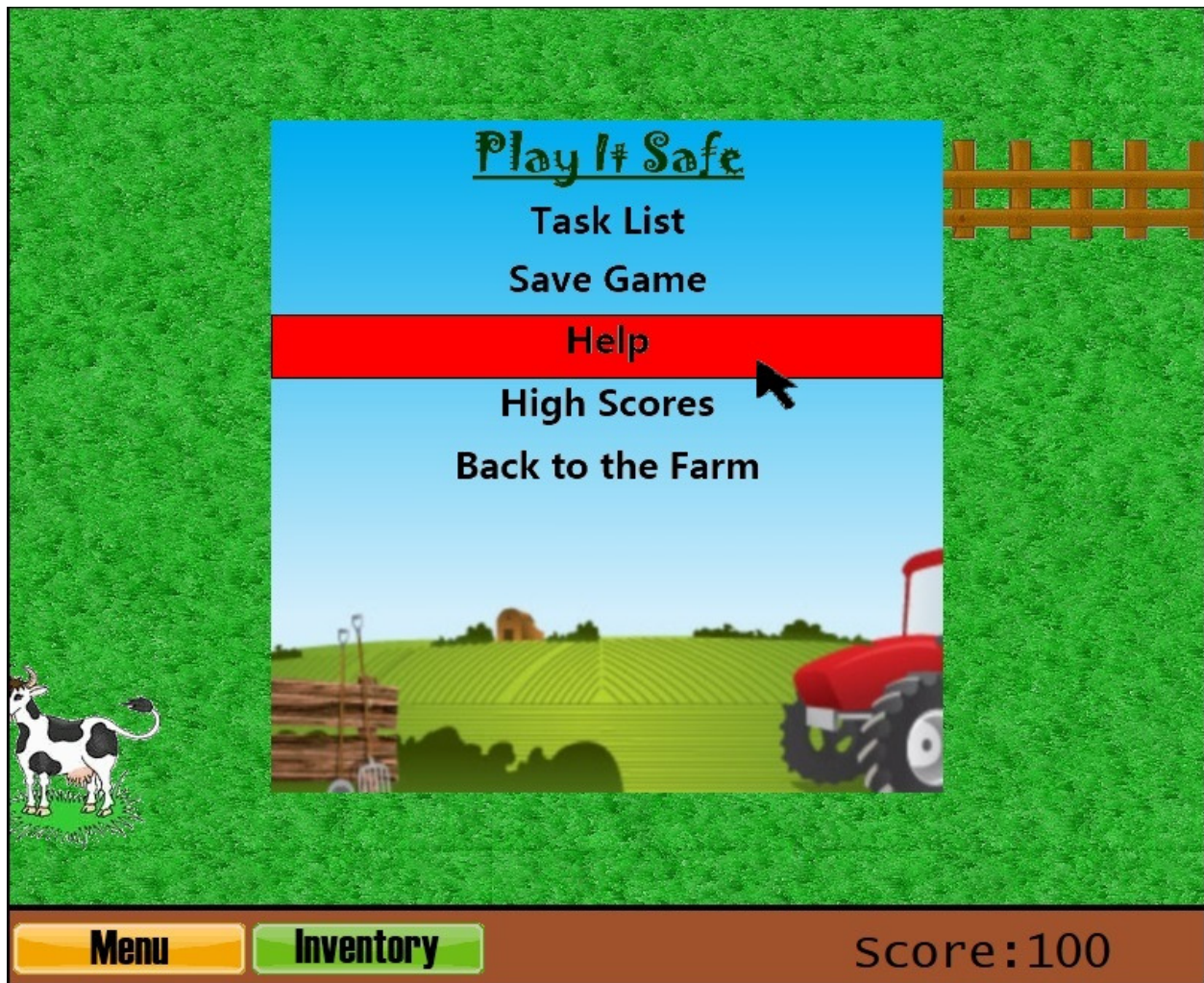


Figure 17: Play It Safe game menu

Appendix B. Source Files

The pages that follow contain source code listings for the NYCAMH Play It Safe online game.

```
<?php
// -----
// FILE: index.php
//
// DESCRIPTION:
//     This file displays the very first screen the user sees when he plays
// the game. He will be given the choice of starting a new game or resuming
// an old game.
// -----

include_once("databaseConnect.php");

class TitleScreen
{
    public function DisplayTitleContent()
    {
        echo '<!doctype html>' . "\n";
        echo '<html>' . "\n";
        echo ' <head>' . "\n";
        echo '     <meta content="text/html; charset=utf-8" http-equiv="Content-Type">' . "\n";
        echo '     <title>NYCAMH - Play It Safe</title>' . "\n";
        echo '     <link rel="stylesheet" href="farm.css" />' . "\n";
        echo ' </head>' . "\n";

        echo ' <body>' . "\n";
        echo '     <div id="farm_frame">' . "\n";
        echo '         <div id="farm">' . "\n";
        echo '             </div>' . "\n";
        echo '             <div id="title_window">' . "\n";
        echo '                 <div id="title_frame_top">' . "\n";
        echo '                     </img>' . "\n";
        echo '                     <a href="start.php?new=1">' . "\n";
        echo '                         </img>' . "\n";
        echo '                     </a>' . "\n";
        echo '                     <a href="start.php?new=0">' . "\n";
        echo '                         </img>' . "\n";
        echo '                     </a>' . "\n";
        echo '                 </div>' . "\n";
        echo '             </div>' . "\n";
        echo '         </div>' . "\n";
        echo ' </body>' . "\n";
        echo '</html>' . "\n";
    }
}

$title = new TitleScreen();
$title->displayTitleContent();

?>
```

```
<?php
// -----
// FILE: start.php
//
// DESCRIPTION:
//     This file is used by index.php to launch a new (or existing) game.
// -----

include_once("databaseConnect.php");

class StartScreen extends DatabaseConnect
{
    private $gameId = -1;

    // -----
    // FUNCTION:    displayHead
    //
    // DESCRIPTION:
    //     This function generates the HTML header info.
    // -----
    public function displayHead()
    {
        echo ' <head>' . "\n";
        echo '     <meta content="text/html; charset=utf-8" http-equiv="Content-Type">' . "\n";
        echo '     <title>NYCAMH - Play It Safe</title>' . "\n";
        echo '     <link rel="stylesheet" href="farm.css" />' . "\n";
        echo ' </head>' . "\n";
    }

    // -----
    // FUNCTION:    displayBodyStart
    //
    // DESCRIPTION:
    //     This function generates the HTML for the first half of the body
    //     tag.
    // -----
    public function displayBodyStart()
    {
        echo ' <body>' . "\n";
        echo '     <div id="farm_frame">' . "\n";
        echo '         <div id="farm">' . "\n";
        echo '             </div>' . "\n";
        echo '         <div id="title_window">' . "\n";
        echo '             <div id="title_frame_top">' . "\n";

    }

    // -----
    // FUNCTION:    displayBodyEnd
    //
    // DESCRIPTION:
    //     This function generates the HTML for the last part of the body
    //     tag.
    // -----
    public function displayBodyEnd()
```

```

{
    echo '                </div>' . "\n";
    echo '                </div>' . "\n";
    echo '            </div>' . "\n";
    echo ' </body>' . "\n";
}

// -----
// FUNCTION:    displayNewGameScreen
//
// DESCRIPTION:
//     This function displays a registration form for creating a new game.
// -----
public function displayStartScreen($newGame)
{
    echo '<!doctype html>' . "\n";
    echo '<html>' . "\n";
    $this->displayHead();
    $this->displayBodyStart();
    if ($newGame)
    {
        $this->displayNewGameScreen();
    }
    else
    {
        $this->displayLoginScreen();
    }
    $this->displayBodyEnd();
    echo '</html>' . "\n";
}

// -----
// FUNCTION:    displayNewGameScreen
//
// DESCRIPTION:
//     This function displays a registration form for creating a new game.
// -----
private function displayNewGameScreen()
{
    echo '<br><form action="start.php?startnew=1" method="post">';
    echo '<p>Username: <input name="username" size="40" value="" type="text"/></p>';
    echo '<p>Password: <input name="password" size="20" value="" type="password"/></p>';
    echo '<p>Reenter Password: <input name="password2" size="20" value=""
type="password"/></p>';
    echo '<input type="submit" name="submitButton" value="Start Game"/>';
    echo '<input type="hidden" name="submitted" value="1"/> ';
    echo '</form>';
}

// -----
// FUNCTION:    displayLoginScreen
//
// DESCRIPTION:
//     This function displays a login form when the user selects resume

```

```

// game.
// -----
private function displayLoginScreen()
{
    echo '<br><form action="start.php?startnew=0" method="post">';
    echo '<p>Username: <input name="username" size="40" value="" type="text"/></p>';
    echo '<p>Password: <input name="password" size="20" value="" type="password"/></p>';
    echo '<input type="submit" name="submitButton" value="Resume Game"/>';
    echo '<input type="hidden" name="submitted" value="1"/> ';
    echo '</form>';
}

// -----
// FUNCTION:   __construct
//
// DESCRIPTION:
//     This function is the constructor for the BaseApi class.
// -----
public function __construct()
{
    $this->databaseConnect();
}

// -----
// FUNCTION:   __destruct
//
// DESCRIPTION:
// -----
public function __destruct()
{
    $this->databaseDisconnect();
}

// -----
// FUNCTION:   createGame
//
// DESCRIPTION:
//     This function will create a new game in the games table, and
//     also update the farmObjects table with the new game info.
// -----
public function createGame($name)
{
    $error = "";
    $q = "SELECT gameId, username FROM games WHERE username='$name'";
    $result = mysql_query($q, $this->mysqlConnection);
    if ($result)
    {
        $numRows = mysql_num_rows($result);
        if ( $numRows == 1 )
        {
            // -----
            // Create farm objects entries for the new game
            // -----
            $row = mysql_fetch_array($result);

```

```

$this->gameId = $row['gameId'];
//print_r($row);
$q = "SELECT * FROM farmObjects WHERE gameId=0";
$result = mysql_query($q, $this->mysqlConnection);
if ($result)
{
    $numRows = mysql_num_rows($result);
    if ( $numRows > 0 )
    {
        while ($row = mysql_fetch_array($result))
        {
            $n = $row['name'];
            $f = $row['filename'];
            $t = $row['type'];
            $w = $row['width'];
            $h = $row['height'];
            $x = $row['xPos'];
            $y = $row['yPos'];
            $p = $row['prxTol'];
            $c = $row['clkTol'];
            $s = $row['screenNum'];
            $i = $row['inventoryKey'];
            $a = $row['animation'];
            $o = $row['obstructs'];
            $txt = $row['textId'];
            $hvr = $row['hoverTxt'];
            $q = "INSERT INTO farmObjects " .
                "(gameId, name, filename, type, width, height, xPos, yPos,
prxTol, clkTol, " .
                "screenNum, inventoryKey, animation, obstructs, textId,
hoverTxt) " .
                "VALUES ($this->gameId, '$n', '$f', $t, $w, $h, $x, $y, $p, $c
, $s, $i, $a, $o, $txt, '$hvr')";
            $writeResult = mysql_query($q, $this->mysqlConnection);
            if (! $writeResult)
            {
                $error = 'Error writing new game farm object data <br>';
            }
        }
    }
    else
    {
        $error = 'Default game farm object data was empty. <br>';
    }
}
else
{
    $error = 'Database access error - error reading default game farm objects
<br>';
}
// -----
// Create a task list for the new game
// -----
$q = "SELECT * FROM taskText";

```

```

        $result = mysql_query($q, $this->mySqlConnection);
        if ($result)
        {
            $numTasks = mysql_num_rows($result);
            for ($task=1; $task<=$numTasks; $task++)
            {
                $q = "INSERT INTO taskList (gameId, taskId, complete) VALUES ($this
->gameId, $task, 0)";
                $writeResult = mysql_query($q, $this->mySqlConnection);
                if (!$writeResult)
                {
                    $error = 'Error writing new game tasks <br>';
                }
            }
        }
        else
        {
            $error = 'Database access error - reading task list text table <br>';
        }
    }
    else
    {
        $error = 'Duplicate name and id found in games table<br>';
    }
}
else
{
    $error = 'Database access error - reading games table <br>';
}

return $error;
}

// -----
// FUNCTION:   checkUserLogin
//
// DESCRIPTION:
//     This function checks the users registration information for an
//     brand new game.
// -----
public function checkRegistrationInfo($name, $password, $password2)
{
    $error = "";

    if (strlen($name) == 0)
    {
        $error = 'A username is required.<br>';
    }
    else if (strlen($password) == 0)
    {
        $error = 'Password is required.<br>';
    }
    else if ($password != $password2)
    {

```

```
        $error = 'Password mismatch. <br>';
    }
    else
    {
        $q = "SELECT gameId FROM games WHERE username='$name'";
        $result = mysql_query($q, $this->mySqlConnection);
        if ($result)
        {
            $numItems = mysql_num_rows($result);
            if ($numItems > 0)
            {
                $error = 'That username is not available. <br>';
            }
            else
            {
                $q = "INSERT INTO games (username, password, score) VALUES ('$name', '$password', 0)";
                $result = mysql_query($q, $this->mySqlConnection);
                if ($result)
                {
                    $error = $this->createGame($name);
                    if (strlen($error) == 0)
                    {
                        //echo "Attempting redirect ...<br>";
                        header("Location: farm.php?id=$this->gameId");
                        exit;
                    }
                }
                else
                {
                    $error = 'Database access error. <br>';
                }
            }
        }
        else
        {
            $error = 'Database access error. <br>';
        }
    }

    echo '<!doctype html>' . "\n";
    echo '<html>' . "\n";
    $this->displayHead();
    $this->displayBodyStart();
    echo $error;
    $this->displayBodyEnd();
    echo '</html>' . "\n";
}

// -----
// FUNCTION:    checkUserLogin
//
// DESCRIPTION:
//     This function checks the users login information for an existing
//     game.
```

```

// -----
public function checkUserLogin($name, $password)
{
    $error = "";

    if (strlen($password) == 0)
    {
        $error = 'Password is required.<br>';
    }
    else
    {
        $q = "SELECT gameId FROM games WHERE username='$name' AND password='$password'";
        $result = mysql_query($q, $this->mysqlConnection);
        if ($result)
        {
            $numItems = mysql_num_rows($result);
            if ($numItems > 0)
            {
                $row = mysql_fetch_array($result);
                $this->gameId = $row[gameId];
                //echo "Attempting redirect ...<br>";
                header("Location: farm.php?id=$this->gameId");
                exit;
            }
            else
            {
                $error = 'Username and password information not found. <br>';
            }
        }
        else
        {
            $error = 'Database access error. <br>';
        }
    }
    echo '<!doctype html>' . "\n";
    echo '<html>' . "\n";
    $this->displayHead();
    $this->displayBodyStart();
    echo $error;
    $this->displayBodyEnd();
    echo '</html>' . "\n";
}
}

// -----
// Parse parameters passed
// 'new' - set to 1 if it is a new game, 0 if it is an old game. This
// is passed from index.php when the user makes his initial selection.
// 'startnew' - Is passed here from this very file when the user either
// enters his registration info, or his existing username and password.
// -----
if ( isset($_GET['new']) )
{
    $newGame = $_GET['new'];
}

```

```
$startScreen = new StartScreen();
$startScreen->displayStartScreen($newGame);
}
else if ( isset($_GET['startnew']) )
{
    $newGame = $_GET['startnew'];
    $startScreen = new StartScreen();
    if ($newGame)
    {
        $registrationFailed = $startScreen->checkRegistrationInfo($_POST[username], $_POST[
password], $_POST[password2]);
    }
    else
    {
        $loginFailed = $startScreen->checkUserLogin($_POST[username], $_POST[password]);
    }
}
else
{
    echo 'An error occurred - no info passed to start.php';
}

```

?>

```
<?php
// -----
// FILE: farm.php
//
// DESCRIPTION:
//     This file is used to generate the HTML document that is the game.
// Javascript data arrays will be included in the document header with the
// specifics of the current game being played.
// -----

include_once("databaseConnect.php");
include_once("farmData.php");

// const is only supported in PHP 5.3.0 and higher
define("SCN_REMOVED", -1);
define("SCN_INVENTORY", 0);
define("SCN_FARM_1", 1);

define("FOT_CHARACTER", 0);
define("FOT_STATIC", 1);
define("FOT_INVENTORY", 2);
define("FOT_ANIMATION", 3);
define("FOT_DEBRIS", 4);
define("FOT_OTHER", 5);
define("FOT_INFO", 6);
define("FOT_QUESTION", 7);

// Required Inventory
define("REQ_INV_SAFETY_GLASSES", 0x00000001);
define("REQ_INV_BOOTS", 0x00000002);
define("REQ_INV_GLOVES", 0x00000004);

// Data structure defining a single farm object that will appear on the farm
class FarmObject
{
    public $name;
    public $filename;
    public $type;
    public $width;
    public $height;
    public $xPos;
    public $yPos;
    public $xCtr;
    public $yCtr;
    public $proximityTolerance;
    public $clickTolerance;
    public $screenNumber;
    public $inventoryKey;
    public $animation;
    public $obstructs;
    public $textId;
    public $hoverTxt;
}
```

```
// Data structure defining a single animation object.
class AnimationObject
{
    public $targetObjectName;
    public $targetObjectIndex;
    public $locationObjectName;
    public $locationObjectIndex;
    public $removeObjectName;
    public $removeObjectIndex;
    public $debrisObjectName;
    public $debrisObjectIndex;
    public $animationName;
    public $taskId;
}

// Data structure defining a task to be completed on the farmer's task list.
class Task
{
    public $taskId;
    public $taskDescription;
    public $complete;
}

// Not implemented ... yet. All play is on screen 1 now.
class FarmScreen
{
    public $screenNumber;
    public $adjacentScreens = array();
}

// Data structure for text found on an "i" information sign
class FarmInfo
{
    public $textId;
    public $text;
}

// Data structure for text found on an "Q" question sign
class FarmQuestion
{
    public $questionId;
    public $text;
    public $answerA;
    public $answerB;
    public $answerC;
    public $answerD;
    public $correctAns;
}

// Data structure for creating a high scores array
class HighScore
{
    public $name;
    public $score;
}
```

```

}

// -----
// Class for generating the farm html document.
// -----
class FarmGrid
{
    // public $farmScreens = array();
    private $farmObjects = array();
    private $farmAnimations = array();
    private $taskList = array();
    private $farmInfo = array();
    private $farmQuestions = array();
    private $highScores = array();
    private $completedQuestions = array();
    private $currentScreen = 1;
    private $gameId = -1;
    private $score = 0;

    // -----
    // FUNCTION:    constructor
    //
    // DESCRIPTION:
    //     This constructor will read all of the necessary elements from the
    //     databases for the game sepecified by $id.
    // -----
    public function __construct($id)
    {
        $this->gameId = $id;
        $farmData = new FarmData();
        $this->farmObjects = $farmData->readFarmObjects($id);
        $this->farmAnimations = $farmData->readAnimations();
        $this->taskList = $farmData->readTaskList($id);
        $this->farmInfo = $farmData->readFarmInfo();
        $this->farmQuestions = $farmData->readFarmQuestions();
        $this->completedQuestions = $farmData->readCompletedQuestions($id);
        $this->score = $farmData->readScore($id);
        $this->highScores = $farmData->getHighScores();

        for ($i=0; $i < count($this->farmAnimations); $i++)
        {
            // Remove and Debris objects are optional ... so initialize them to an invalid index.
            $this->farmAnimations[$i]->removeObjectIndex = -1;
            $this->farmAnimations[$i]->debrisObjectIndex = -1;

            for ($j=0; $j < count($this->farmObjects); $j++)
            {
                if ($this->farmObjects[$j]->name == $this->farmAnimations[$i]->targetObjectName)
                {
                    $this->farmAnimations[$i]->targetObjectIndex = $j;
                }
                if ($this->farmObjects[$j]->name == $this->farmAnimations[$i]->
locationObjectName)
                {

```

```

        $this->farmAnimations[$i]->locationObjectIndex = $j;
    }
    if ($this->farmObjects[$j]->name == $this->farmAnimations[$i]->removeObjectName)
    {
        $this->farmAnimations[$i]->removeObjectIndex = $j;
    }
    if ($this->farmObjects[$j]->name == $this->farmAnimations[$i]->debrisObjectName)
    {
        $this->farmAnimations[$i]->debrisObjectIndex = $j;
    }
}
}

// -----
// FUNCTION:   displayScreenGrid
//
// DESCRIPTION:
//     This function generates the HTML for the farm.
// -----
public function displayScreenGrid()
{
    echo '<!doctype html>' . "\n";
    echo '<html>' . "\n";
    echo '  <head>' . "\n";
    echo '    <meta content="text/html; charset=utf-8" http-equiv="Content-Type">' . "\n";
    //echo '    <meta content="IE=Edge" http-equiv="X-UA-Compatible">' . "\n";
    echo '    <title>NYCAMH - Play It Safe</title>' . "\n";
    echo '    <script src="jquery.js"></script>' . "\n";
    echo '    <script src="farm.js" type="text/javascript"></script>' . "\n";
    echo '    <link rel="stylesheet" href="farm.css" />' . "\n";
    echo '    <script type="text/javascript">' . "\n";
    echo '      currentScreen = ' . $this->currentScreen . ';' . "\n";
    echo '      gameId = ' . $this->gameId . ';' . "\n";
    echo '      score = ' . $this->score . ';' . "\n";
    for ($i=0; $i<count($this->farmObjects); $i++)
    {
        echo '          farmObjects[' . $i . '] = [ "' . $this->farmObjects[$i]->
name          . '", "' .
                                $this->farmObjects[$i]->
filename      . '", ' .
                                $this->farmObjects[$i]->
type          . '", ' .
                                $this->farmObjects[$i]->
xPos          . '", ' .
                                $this->farmObjects[$i]->
yPos          . '", ' .
                                $this->farmObjects[$i]->
width         . '", ' .
                                $this->farmObjects[$i]->
height        . '", ' .
                                $this->farmObjects[$i]->
xCtr          . '", ' .
                                $this->farmObjects[$i]->

```

```

yCtr . ', ' . $this->farmObjects[$i]->
proximityTolerance . ', ' . $this->farmObjects[$i]->
clickTolerance . ', ' . $this->farmObjects[$i]->
screenNumber . ', ' . $this->farmObjects[$i]->
inventoryKey . ', ' . $this->farmObjects[$i]->
animation . ', ' . $this->farmObjects[$i]->
obstructs . ', ' . $this->farmObjects[$i]->
textId . ', "' . $this->farmObjects[$i]->
hoverTxt . '" ];' . "\n";
    }
    for ($i=0; $i<count($this->farmAnimations); $i++)
    {
        echo '          animations[' . $i . '] = [ ' . $this->farmAnimations[$i]->
targetObjectIndex . ', ' . $this->farmAnimations[$i]->
locationObjectIndex . ', ' . $this->farmAnimations[$i]->
removeObjectIndex . ', ' . $this->farmAnimations[$i]->
debrisObjectIndex . ', "' . $this->farmAnimations[$i]->
animationName . '" , ' . $this->farmAnimations[$i]->taskId.
' ];' . "\n";
    }
    for ($i=0; $i<count($this->taskList); $i++)
    {
        echo '          tasks[' . $i . '] = [ ' . $this->taskList[$i]->taskId . ', "' .
$this->taskList[$i]->taskDescription .
' , ' . $this->taskList[$i]->complete . ', ' .
$this->taskList[$i]->taskId . ' ];' .
"\n";
    }
    for ($i=0; $i<count($this->farmInfo); $i++)
    {
        echo '          info[' . $i . '] = [ ' . $this->farmInfo[$i]->textId . ', "' .
$this->farmInfo[$i]->text . '" ];' . "\n";
    }
    for ($i=0; $i<count($this->farmQuestions); $i++)
    {
        $questionDone = in_array($this->farmQuestions[$i]->questionId, $this->
completedQuestions) ? 1 : 0;
        echo '          questions[' . $i . '] = [ ' . $this->farmQuestions[$i]->
questionId . ', "' . $this->farmQuestions[$i]->text .

```

```

", " .
$this->farmQuestions[$i]->answerA .
", " .
$this->farmQuestions[$i]->answerB .
", " .
$this->farmQuestions[$i]->answerC .
", " .
$this->farmQuestions[$i]->answerD .
", ' .
$this->farmQuestions[$i]->
correctAns . ', ' .
$questionDone . '];' . "\n";
}
for ($i=0; $i<count($this->highScores); $i++)
{
    echo '          highScores[' . $i . '] = [ "' . $this->highScores[$i]->name . '", ' .
        $this->highScores[$i]->score . '];'
    . "\n";
}
echo '    </script>' . "\n";
echo '    <style type="text/css">' . "\n";
for ($i=0; $i<count($this->farmObjects); $i++)
{
    switch ($this->farmObjects[$i]->type)
    {
        case FOT_DEBRIS:
        case FOT_INVENTORY:
        case FOT_ANIMATION:
        case FOT_INFO:
        case FOT_QUESTION:
            echo '# ' . $this->farmObjects[$i]->name .
                ' { position: absolute; background: url(' . $this->farmObjects[$i]->
filename .
                    ') no-repeat; width: ' . $this->farmObjects[$i]->width . 'px;
height: ' . $this->farmObjects[$i]->height .
                    'px; z-index: -1 }' . "\n";
            break;
        case FOT_OTHER:
        case FOT_STATIC:
            echo '# ' . $this->farmObjects[$i]->name .
                ' { position: absolute; background: url(' . $this->farmObjects[$i]->
filename .
                    ') no-repeat; width: ' . $this->farmObjects[$i]->width . 'px;
height: ' . $this->farmObjects[$i]->height .
                    'px; z-index: -5 }' . "\n";
            break;
    }
}
echo '    </style>' . "\n";

//echo '    <!--Adobe Edge Runtime-->' . "\n";
//echo '    <script type="text/javascript" charset="utf-8"
src="mowing_edgePreload.js"></script>' . "\n";
//echo '    <style>' . "\n";

```

```
//echo '          .edgeLoad-EdgeContent { visibility:hidden; }' . "\n";  
//echo '          </style>' . "\n";  
//echo '          <!--Adobe Edge Runtime End-->' . "\n";
```

```
echo ' </head>' . "\n";
```

```
echo ' <body>' . "\n";
```

```
echo ' </body>' . "\n";
```

```
echo '</html>' . "\n";
```

```
}
```

```
}
```

```
if ( isset($_GET['id']) )
```

```
{
```

```
    $farm = new FarmGrid($_GET['id']);
```

```
    $farm->displayScreenGrid();
```

```
}
```

```
?>
```

```
// -----  
// NYCAMH Play It Safe Farm Safety Game  
// Javascript include file  
//  
// SUNYIT  
// rdb - 2013  
// -----  
  
var debugOn = false;  
  
// -----  
// Constants  
// -----  
  
var MOVE_INCREMENT = 15;           // Number of pixels farmer will move  
var CHAR_SPEED     = 150;          // how fast the character will move  
var FARM_FRAME_WIDTH = 800;        // Size of the window the user is viewing. These must  
var FARM_FRAME_HEIGHT = 600;      // be equal to width and height of farm_frame css div  
var SCROLL_TOLERANCE = 100;       // Distance to the edge of a frame before a scroll occurs  
var ANIMATION_START_TIME = 5000;  
var WINDOW_START_DELAY = 150;  
  
// Point values  
POINTS_QUESTION     = 10;  
POINTS_TASK         = 100;  
  
// Window Active constants  
var WA_NONE         = 0x00;  
var WA_INVENTORY    = 0x01;  
var WA_MENU         = 0x02;  
var WA_TASKLIST     = 0x04;  
var WA_MESSAGE      = 0x08;  
var WA_INFO         = 0x10;  
var WA_HELP         = 0x20;  
var WA_HIGHScores   = 0x40;  
var WA_QUESTION     = 0x80;  
  
// Farm Object Array Definitions  
var FO_NAME         = 0;           // Name  
var FO_FILE         = 1;           // Filename  
var FO_TYPE         = 2;           // Farm Item Type  
var FO_X_POS        = 3;  
var FO_Y_POS        = 4;  
var FO_WIDTH        = 5;  
var FO_HEIGHT       = 6;  
var FO_X_CTR        = 7;           // X Position (center of object)  
var FO_Y_CTR        = 8;           // Y Position (center of object)  
var FO_PT           = 9;           // Proximity Tolerance  
var FO_CT           = 10;          // Click Tolerance  
var FO_LOC          = 11;          // Location (0=Farmers Inventory, 1-n screen number)  
var FO_REQ_INV      = 12;          // Required Inventory for animation objects  
var FO_ANIMATE      = 13;  
var FO_OBSTRUCT     = 14;          // Cannot walk through object  
var FO_TEXT         = 15;          // Text id for FOT_INFO, and FO_QUESTION  
var FO_HOVERTXT    = 16;
```

```
// Farm screen locations
var SCN_REMOVED      = -1;
var SCN_INVENTORY    = 0;
var SCN_FARM_1       = 1;

// Farm Object types
var FOT_CHARACTER    = 0;
var FOT_STATIC       = 1;
var FOT_INVENTORY    = 2;
var FOT_ANIMATION    = 3;
var FOT_DEBRIS       = 4;
var FOT_OTHER        = 5;
var FOT_INFO         = 6;
var FOT_QUESTION     = 7;

// Task List indices
var TSK_ID           = 0;
var TSK_DESC         = 1;
var TSK_DONE         = 2;

// Farm Info indices
var FI_ID            = 0;
var FI_TEXT          = 1;

// Farm Question indices
var FQ_ID            = 0;
var FQ_TEXT          = 1;
var FQ_ANS_A         = 2;
var FQ_ANS_B         = 3;
var FQ_ANS_C         = 4;
var FQ_ANS_D         = 5;
var FQ_CA            = 6;
var FQ_DONE          = 7;

// High Score indices
var HS_NAME          = 0;
var HS_SCORE         = 1;

// Required Inventory
var REQ_INV_SAFETY_GLASSES = 0x00000001;
var REQ_INV_BOOTS     = 0x00000002;
var REQ_INV_GLOVES    = 0x00000004;
var REQ_INV_EAR_PROTECTION = 0x00000008;
var REQ_INV_DUST_MASK = 0x00000010;
var REQ_INV_RUBBER_BOOTS = 0x00000020;
var REQ_INV_BREATHING_APP = 0x00000040;
var REQ_INV_SUNSCREEN = 0x00000080;

var A_TARGET_OBJ     = 0;
var A_LOCATION_OBJ   = 1;
var A_REMOVE_OBJ     = 2;
var A_DEBRIS_OBJ     = 3;
var A_ANIMATION      = 4;
```

```

var A_TASK_ID          = 5;

var REQUIRED_INVENTORY = "You do not have the required inventory to complete the task";
var DEBRIS_REMOVAL    = "Debris removal is required before you can proceed";
var TIME_MESSAGE      = 3500;
var TIME_WRONG_ANSWER = 7000;
// -----
// Global variables that will be accessed in the functions below.
// -----
var currentKey;           // records the current key pressed
var TimerWalk;           // timer handle
var charStep = 2;        // 1=1st foot, 2=stand, 3=2nd foot, 4=stand
var windowActive = WA_NONE; // Indicates a popup (menu, inventory) is active.
var currentScreen;      // The screen the farmer character is located.
var farmObjects = new Array(); // Array storing Objects on the farm
var animations = new Array(); // Array storing animation info
var tasks = new Array(); // Array holding the tasks to be completed on the farm.
var info = new Array(); // Array holding info sign text
var questions = new Array(); // Array holding questions.
var highScores = new Array(); // Array holding the current high scores
var gameId = -1;
var score = 0;
var controlKeyDown = false;
var currentQuestionObj;

// -----
// logDebug
// -----
function logDebug(debugString)
{
    if (debugOn)
    {
        console.debug(debugString);
    }
}

// -----
// displayMessageWindow
// -----
function displayMessageWindow(displayString, timeout, inventory)
{
    var msgWindow = '<div id="message_window" ></div>';
    $("#farm_frame").append(msgWindow);
    var msgFrame = '<div id="message_frame" ></div>';
    $("#message_window").append(msgFrame);
    var msgText = '<div id="message_text_center">' + displayString + '</div>';

    // Put the specified inventory pictures in the message window
    if (inventory !=0)
    {
        var msgText = '<br><div id="message_multiple_items">' + displayString;
        msgText += '<br><table width="100%"><tr>';
        for (i=0; i<farmObjects.length; i++)
        {

```

```

        if ((farmObjects[i][FO_TYPE] == FOT_INVENTORY) &&
            ((farmObjects[i][FO_REQ_INV] & inventory) != 0))
        {
            msgText += '<td></td>';
        }
    }
    msgText += '</tr></table></div>';
}

// If the user clicks on the window ... remove it
$('#message_frame').click(
    function(e)
    {
        killMessageWindow();
        return false;
    }
);

$("#message_frame").append(msgText);
windowActive |= WA_MESSAGE;

// If the caller specified a timeout create a function to remove the window
if (timeout > 0)
{
    setTimeout( function()
        {
            var msgWndId = document.getElementById('message_window');
            if (msgWndId)
            {
                msgWndId.parentNode.removeChild(msgWndId);
                windowActive &= ~WA_MESSAGE;
            }
        }, timeout);
}
}

// -----
// killMessageWindow
// -----
function killMessageWindow()
{
    var messageWindowId = document.getElementById('message_window');
    messageWindowId.parentNode.removeChild(messageWindowId);
    windowActive &= ~WA_MESSAGE;
}

// -----
// updateScore
// -----
function updateScore(adjustment)
{
    logDebug("updateScore");
    score += adjustment;
    logDebug("updateScore - new score = " + score);
}

```

```

    $('#score').text(score);
}

// -----
// addItemToFarm
// -----
function addItemToFarm(index)
{
    logDebug("appending " + farmObjects[index][FO_NAME] + " to the farm.");
    var frmObj = '<div id="' + farmObjects[index][FO_NAME] + '" class="' +
                farmObjects[index][FO_NAME] + '" style="left:' +
                farmObjects[index][FO_X_POS] + 'px; top:' +
                farmObjects[index][FO_Y_POS] + 'px";
    if (farmObjects[index][FO_HOVERTXT].length > 0)
    {
        frmObj += ' title="' + farmObjects[index][FO_HOVERTXT] + '"';
    }
    frmObj += '</div>';

    logDebug("frmObj: " + frmObj);
    $('#farm').append(frmObj);
    farmObjects[index][FO_LOC] = currentScreen;
}

// -----
// Initialization
// -----
$(function()
{
    logDebug("Initialization starting ... ");

    $(document.body).append('<div id="farm_frame"></div>');
    $('#farm_frame').click(
        function(e)
        {
            processFarmFrameClick(e);
        }
    );

    $('#farm_frame').append('<div id="farm"></div>');

    logDebug("Processing " + farmObjects.length + " farm objects");
    for (i=0; i < farmObjects.length; i++)
    {
        logDebug("farm object " + farmObjects[i]);
        if (farmObjects[i][FO_LOC] == currentScreen)
        {
            addItemToFarm(i);
        }
    }

    $(document.body).append('<div id="buttons_window"></div>');
    $('#buttons_window').append('<div id="buttons_frame"></div>');
}

```

```
$("#buttons_frame").append('<div id="menu_button"></div>');
$('#menu_button').click(
    function(e)
    {
        processMenuButtonClick();
    }
);

$("#buttons_frame").append('<div id="inventory_button"></div>');
$('#inventory_button').click(
    function(e)
    {
        processInventoryButtonClick();
    }
);

$("#buttons_frame").append('<div id="score">' + score + '</div>');
$("#buttons_frame").append('<div id="score_text">Score: </div>');

scrollFarm('left');

logDebug("Initialization complete.");
}
);

// -----
// Once the DOM is ready, set the character to facing forward (default position)
// -----
$(document).ready(
    function()
    {
        //add character state class
        $('#farmer').addClass('front-stand');
    }
);

// -----
// KeyDown Function
// if there is no currentKey down, execute charWalk
// -----
$(document).keydown(
    function(e)
    {
        if (e.keyCode == 17)
        {
            controlKeyDown = true;
        }

        if (!currentKey)
        {
            //set the currentKey to the key that is down
            currentKey = e.keyCode;

            //execute character movement function charWalk('direction')
```

```

        switch(e.keyCode)
        {
            case 38: charWalk('up');    break;
            case 39: charWalk('right'); break;
            case 40: charWalk('down');  break;
            case 37: charWalk('left');  break;
        }
    }
}
);

// -----
//  KeyUp - User released a key
// -----

$(document).keyup(
    function(e)
    {
        if (controlKeyDown)
        {
            if (e.keyCode == 190) // '.' (period)
            {
                logDebug("Debug is " + !debugOn);
                debugOn = !debugOn;
                logDebug("Debug is " + debugOn);
            }
            else if (e.keyCode == 17)
            {
                controlKeyDown = false;
            }
        }

        //don't stop the walk if the player is pushing other buttons
        //only stop the walk if the key that started the walk is released
        if (e.keyCode == currentKey)
        {
            //set the currentKey to false, this will enable a new key to be pressed
            currentKey = false;

            //clear the walk timer
            clearInterval(TimerWalk);

            //finish the character's movement
            $('#farmer').stop(true, true);
        }
    }
);

// -----
//  charWalk - Character Walk Function
// -----

function charWalk(dir)
{
    //adjust from lang to code
    if (dir == 'up') dir = 'back';
}

```

```

if (dir == 'down') dir = 'front';

//move the character
processWalk(dir);

//set the interval timer to continually move the character
TimerWalk = setInterval(function() { processWalk(dir); }, CHAR_SPEED);
}

// -----
// processWalk - Process Character Walk Function
// -----
function processWalk(dir)
{
    // -----
    // Determine the end position for character to pass to checkForObstruction
    // -----
    var farmerPosition = new FarmerPositionObject();
    var endX = farmerPosition.feetX;
    var endY = farmerPosition.feetY;
    switch (dir)
    {
        case 'back': endY -= MOVE_INCREMENT; break;
        case 'front': endY += MOVE_INCREMENT; break;
        case 'left': endX -= MOVE_INCREMENT; break;
        case 'right': endX += MOVE_INCREMENT; break;
    }

    // -----
    // If there is an obstruction ... don't proceed.
    // -----
    if ( ! checkForObstruction(endX, endY) )
    {
        //increment the charStep as we will want to use the next stance in the animation
        //if the character is at the end of the animation, go back to the beginning //increment
the charStep as we will want to use the next stance in the animation
        charStep++;
        if (charStep == 5) charStep = 1;

        //remove the current class
        $('#farmer').removeAttr('class');

        //add the new class
        switch(charStep)
        {
            case 1: $('#farmer').addClass(dir+'-stand'); break;
            case 2: $('#farmer').addClass(dir+'-right'); break;
            case 3: $('#farmer').addClass(dir+'-stand'); break;
            case 4: $('#farmer').addClass(dir+'-left'); break;
        }

        var position = $("#farmer").position();

        //move the char

```

```

//we will only want to move the character 32px (which is 1 unit) in any direction
switch(dir)
{
  case'front':
    var farmHeight = $("#farm").height();
    var farmerHeight = $("#farmer").height();

    if (position.top <= (farmHeight-farmerHeight-MOVE_INCREMENT))
    {
      $('#farmer').stop().animate({top: '+=' + MOVE_INCREMENT}, CHAR_SPEED);
    }
    break;
  case'back':
    //don't let the character move any further up if they are already at the top of
the screen
    if (position.top > 0)
    {
      $('#farmer').stop().animate({top: '-=' + MOVE_INCREMENT}, CHAR_SPEED);
    }
    break;
  case'left':
    //don't let the character move any further left if they are already at the left
side of the screen
    if (position.left > 0)
    {
      $('#farmer').stop().animate({left: '-=' + MOVE_INCREMENT}, CHAR_SPEED);
    }
    break;
  case'right':
    var farmWidth = $("#farm").width();
    var farmerWidth = $("#farmer").width();
    if (position.left <= (farmWidth-farmerWidth-MOVE_INCREMENT))
    {
      $('#farmer').stop().animate({left: '+=' + MOVE_INCREMENT}, CHAR_SPEED);
    }
    break;
}

scrollFarm(dir);
}
}

// -----
// moveCharacterToMousePosition - Moves the character position based on mouse
// click.
// -----
function moveCharacterToMousePosition(click)
{
  //remove the current class
  $('#farmer').removeAttr('class');

  var farmerPosition = new FarmerPositionObject();
  // northwest of current pos
  if ((click.xPos <= farmerPosition.xPos) && (click.yPos <= farmerPosition.yPos))

```

```

{
    $('#farmer').addClass('left-stand');
}
// northeast of current pos
if ((click.xPos >= farmerPosition.xPos) && (click.yPos <= farmerPosition.yPos))
{
    $('#farmer').addClass('back-stand');
}
// southwest of current pos
if ((click.xPos <= farmerPosition.xPos) && (click.yPos >= farmerPosition.yPos))
{
    $('#farmer').addClass('front-stand');
}
// southeast of current pos
if ((click.xPos >= farmerPosition.xPos) && (click.yPos >= farmerPosition.yPos))
{
    $('#farmer').addClass('right-stand');
}

if ( ! checkForObstruction(click.xPos, click.yPos))
{
    var animateLeft = click.xPos - (farmerPosition.width/2);
    var animateTop = click.yPos - (farmerPosition.height/2);
    $('#farmer').stop().animate({
        left:animateLeft,
        top:animateTop
    },
    {
        queue: false,
        complete: function()
        {
            scrollFarm('all');
        }
    } );
}
}

// -----
// checkForObstruction
// -----
function checkForObstruction(endX, endY)
{
    var pathObstructed = false;

    var i=0;
    while ( (i < farmObjects.length) && ! pathObstructed )
    {
        if ((farmObjects[i][FO_LOC] == currentScreen) && (farmObjects[i][FO_OBSTRUCT] == 1))
        {
            var x1 = farmObjects[i][FO_X_POS];
            var y1 = farmObjects[i][FO_Y_POS];
            var x2 = x1 + farmObjects[i][FO_WIDTH];
            var y2 = y1 + farmObjects[i][FO_HEIGHT];

```

```

    if ( (endX > x1) && (endX < x2) && (endY > y1) && (endY < y2))
    {
        logDebug('PATH OBSTRUCTED BY ' + farmObjects[i][FO_NAME]);
        logDebug('endX: ' + endX + ' endY: ' + endY );
        logDebug('x1: ' + x1 + ' y1: ' + y1 );
        logDebug('x2: ' + x2 + ' y2: ' + y2 );
        pathObstructed = true;
    }
}
i++;
}

return pathObstructed;
}

// -----
// scrollFarm
// -----
function scrollFarm(dir)
{
    logDebug('ScrollFarm enter (dir = ' + dir + ')');
    var scrollNeeded = false;
    var farmerPosition = new FarmerPositionObject();
    var farmPosition = new FarmPositionObject();

    if ( (dir == 'front') || (dir == 'back') || (dir == 'all') )
    {
        if ( ((Math.abs(farmerPosition.yPos - farmPosition.yPos) > (FARM_FRAME_HEIGHT -
SCROLL_TOLERANCE)) ||
            (farmerPosition.yPos < (farmPosition.yPos + SCROLL_TOLERANCE))) &&
            ((farmPosition.yPos + FARM_FRAME_HEIGHT) <= $('#farm').height()) )
        {
            scrollNeeded = true;
        }
    }

    if ( (dir == 'left') || (dir == 'right') || (dir == 'all') )
    {
        if ( ((Math.abs(farmerPosition.xPos - farmPosition.xPos) > (FARM_FRAME_WIDTH -
SCROLL_TOLERANCE)) ||
            (farmerPosition.xPos < (farmPosition.xPos + SCROLL_TOLERANCE))) &&
            ((farmPosition.xPos + FARM_FRAME_WIDTH) <= $("#farm").width()) )
        {
            scrollNeeded = true;
        }
    }

    if (scrollNeeded)
    {
        var scrollX = farmerPosition.xPos - (FARM_FRAME_WIDTH/2);
        if (scrollX < 1)
        {
            scrollX = 1;
        }
    }
}

```

```

    if (scrollX > ($("#farm").width() - FARM_FRAME_WIDTH))
    {
        scrollX = ($("#farm").width() - FARM_FRAME_WIDTH);
    }
    var scrollY = farmerPosition.yPos - (FARM_FRAME_HEIGHT/2);
    if (scrollY < 1)
    {
        scrollY = 1;
    }
    if (scrollY > ($("#farm").height() - FARM_FRAME_HEIGHT))
    {
        scrollY = ($("#farm").height() - FARM_FRAME_HEIGHT);
    }
    logDebug('SCROLLING farm to: ' + scrollX + ", " + scrollY);
    $("#farm").stop().animate({left:'-'+scrollX, top:'-'+scrollY});
}

logDebug('ScrollFarm exit!');
}

// -----
// Farm Position Object
// -----
function FarmPositionObject ()
{
    // Determine the center pixel location of the farmer
    this.position = ($("#farm").position());
    this.xPos = Math.abs(this.position.left);
    this.yPos = Math.abs(this.position.top);

    logDebug('Farm: xPos:'+this.xPos+' yPos:'+this.yPos);
}

// -----
// Farmer Position Object
// -----
function FarmerPositionObject ()
{
    // Determine the center pixel location of the farmer
    this.position = ($("#farmer").position());
    this.height = ($("#farmer").height());
    this.width = ($("#farmer").width());
    this.xPos = this.position.left + (this.width/2); // position center x
    this.yPos = this.position.top + (this.height/2); // position center y
    this.feetX = this.xPos; // position center at farmers feet x
    this.feetY = this.position.top + this.height; // position center at farmers feet y

    logDebug('Farmer: xPos:'+this.xPos+' yPos:'+this.yPos);
}

// -----
// Farm Mouse Click Object
// -----
function MouseClickObject (e)

```

```

{
    this.absX = e.pageX;
    this.absY = e.pageY;
    var framePosition = $("#farm_frame").position();
    logDebug('Farm Frame is at: '+framePosition.left+', '+framePosition.top);

    var farmPosition = new FarmPositionObject();

    var farmerPosition = new FarmerPositionObject();

    this.xPos = (this.absX - framePosition.left) + farmPosition.xPos;
    this.yPos = (this.absY - framePosition.top) + farmPosition.yPos;

    logDebug('MouseClickedObject Exit - xPos:'+this.xPos+ ' yPos:'+this.yPos+ ' absX:'+this.absX+
absY:'+this.absY);
}

// -----
// Pick Up Object
// -----
function pickUpItem(farmObj)
{
    logDebug("Picking up object: " + farmObjects[farmObj.index]);
    $('.' + farmObj.name).remove();
    farmObjects[farmObj.index][FO_LOC] = SCN_INVENTORY;
}

// -----
// Remove Debris
// -----
function removeDebris(farmObj)
{
    logDebug('Removing Debris ...');
    $('.' + farmObj.name).remove();
    farmObjects[farmObj.index][FO_LOC] = SCN_REMOVED;
}

// -----
// ProcessAnimation
// -----
function processAnimation(farmObj)
{
    logDebug("processAnimation enter ...");

    // Create a list of the items currently in inventory
    var itemsInInventory = 0;
    for (i=0; i<farmObjects.length; i++)
    {
        logDebug(i + " " + farmObjects[i][FO_NAME] + " " + farmObjects[i][FO_LOC]);
        if (farmObjects[i][FO_LOC] == SCN_INVENTORY)
        {
            itemsInInventory += farmObjects[i][FO_REQ_INV];
        }
    }
}

```

```

    logDebug("current inventory: " + itemsInInventory + " required inventory: " + farmObj.
requiredInventory);

// Check to see if player has required inventory for the task/animation
if ((itemsInInventory & farmObj.requiredInventory) == farmObj.requiredInventory)
{
    // Check to see if debris removal was done.
    if ((farmObj.animation.debrisObjectIndex >= 0) &&
        (farmObjects[farmObj.animation.debrisObjectIndex][FO_LOC] != SCN_REMOVED))
    {
        displayMessageWindow(DEBRIS_REMOVAL, TIME_MESSAGE, 0);
    }
    else
    {
        // In order for this load of the animation to work, the
<animation-name>_edgePreload.js file needs to be
        // updated. The following line: "window.addEventListener("load", onDocLoaded,
false);" needs to be
        // replaced with "onDocLoaded();"

        // This is an attempt to get the edge animation to run within the current html
document
        //$('#' + farmObjects[farmObj.animation.locationObjectIndex][FO_NAME]).append(
'<div id=stage class="EDGE-3207839"></div>');
        //onDocLoaded();

        // This code loads the edge generated html document for the animation into the
appropriate div
        $('#' + farmObjects[farmObj.animation.locationObjectIndex][FO_NAME]).load( farmObj.
animation.animation + ".html" );

        // If there is an object to remove after the animation completes, set a timeout for
it to be removed.
        logDebug("Remove object index is " + farmObj.animation.removeObjectIndex);
        if (farmObj.animation.removeObjectIndex >= 0)
        {
            setTimeout( function()
            {
                logDebug("Removing " + farmObjects[farmObj.animation.
removeObjectIndex][FO_NAME]);
                $('#' + farmObjects[farmObj.animation.removeObjectIndex][FO_NAME
]).css("background", "transparent");
            },
            ANIMATION_START_TIME);

            farmObjects[farmObj.animation.removeObjectIndex][FO_LOC] = SCN_REMOVED;
        }

        // Convert the animation object to a static object ... so the animation cannot be
run again.
        farmObjects[farmObj.index][FO_TYPE] = FOT_OTHER;

        // Find the task and mark it done.

```

```

    var i=0;
    while ((tasks[i][TSK_ID] != farmObj.animation.taskId) && (i<tasks.length)) i++;
    if ((i < tasks.length) && (tasks[i][TSK_ID] == farmObj.animation.taskId))
    {
        tasks[i][TSK_DONE] = 1;
        // add ajax update here.
    }

    updateScore(POINTS_TASK);
}
}
else
{
    displayMessageWindow(REQUIRED_INVENTORY, TIME_MESSAGE, farmObj.requiredInventory);
}

logDebug("processAnimation exit.");
}

// -----
// Animation Object
// -----
function AnimationObject(index)
{
    logDebug("AnimationObject enter with index=" + index);
    if (index >= 0)
    {
        this.targetObjectIndex = animations[index][A_TARGET_OBJ];
        this.locationObjectIndex = animations[index][A_LOCATION_OBJ];
        this.debrisObjectIndex = animations[index][A_DEBRIS_OBJ];
        this.removeObjectIndex = animations[index][A_REMOVE_OBJ];
        this.animation = animations[index][A_ANIMATION];
        this.taskId = animations[index][A_TASK_ID];
    }
}

// -----
// Farm Object
// -----
function FarmObject(click)
{
    logDebug("FarmObject enter");
    this.inProximity = false;
    var i = 0;
    while ((i < farmObjects.length) && ( ! this.clicked ))
    {
        if (farmObjects[i][FO_LOC] == currentScreen)
        {
            logDebug("FarmObject checking " + farmObjects[i][FO_NAME] + " at " +
                farmObjects[i][FO_X_CTR] + ", " + farmObjects[i][FO_Y_CTR]);
            if ( (Math.abs(click.xPos-farmObjects[i][FO_X_CTR]) < farmObjects[i][FO_CT]) &&
                (Math.abs(click.yPos-farmObjects[i][FO_Y_CTR]) < farmObjects[i][FO_CT]) )
            {
                this.clicked = true;
            }
        }
    }
}

```

```

    this.index = i;

    var farmerPos = new FarmerPositionObject ();

    // Check to see if the character is within proximity of the edu component
    if ((Math.abs(farmerPos.xPos - click.xPos) <= farmObjects[i][FO_PT]) &&
        (Math.abs(farmerPos.yPos - click.yPos) <= farmObjects[i][FO_PT]))
    {
        this.inProximity = true;
        this.name = farmObjects[i][FO_NAME];
        this.type = farmObjects[i][FO_TYPE];
        this.requiredInventory = farmObjects[i][FO_REQ_INV];
        this.animation = new AnimationObject (farmObjects[i][FO_ANIMATE]);
        this.textId = farmObjects[i][FO_TEXT];
    }
}
}
i++;
}
logDebug("FarmObject exit");
}

// -----
// processSaveGame
// -----
function processSaveGame ()
{
    xmlhttp = new XMLHttpRequest ();

    // Remove the menu first
    killMenu();

    displayMessageWindow("Saving Game ...", -1, 0);
    setTimeout( function()
    {
        // Update the score
        var ajaxString = "gameUpdate.php?u=score&g=" + gameId + "&pts=" + score;
        if (debugOn)
        {
            ajaxString += "&debug=1";
        }
        xmlhttp.open("GET", ajaxString, false);
        xmlhttp.send();

        // Update location of farm objects
        for (i=0; i < farmObjects.length; i++)
        {
            if (farmObjects[i][FO_TYPE] != FOT_STATIC)
            {
                if (farmObjects[i][FO_TYPE] == FOT_CHARACTER)
                {
                    farmerPos = new FarmerPositionObject ();
                    farmObjects[i][FO_X_POS] = farmerPos.position.left;
                    farmObjects[i][FO_Y_POS] = farmerPos.position.top;
                }
            }
        }
    }
    );
}

```

```

    }

    ajaxString = "gameUpdate.php?u=pos&g=" + gameId +
        "&n=" + farmObjects[i][FO_NAME] +
        "&s=" + farmObjects[i][FO_LOC] +
        "&x=" + farmObjects[i][FO_X_POS] +
        "&y=" + farmObjects[i][FO_Y_POS] +
        "&t=" + farmObjects[i][FO_TYPE];

    if (debugOn)
    {
        ajaxString += "&debug=1";
    }
    xmlhttp.open("GET", ajaxString, false);
    xmlhttp.send();
}

// Update the task list
for (i=0; i < tasks.length; i++)
{
    if (tasks[i][TSK_DONE] != 0)
    {
        ajaxString = "gameUpdate.php?u=task&g=" + gameId + "&tid=" + tasks[i][TSK_ID];
        if (debugOn)
        {
            ajaxString += "&debug=1";
        }
        xmlhttp.open("GET", ajaxString, false);
        xmlhttp.send();
    }
}

killMessageWindow();
displayMessageWindow("Game Saved", 2000, 0);
}, 500);
}

// -----
// displayTaskList
// -----
function displayTaskList ()
{
    logDebug("displayTaskList enter ...");

    // Remove the menu first
    killMenu();

    var taskListWindow = '<div id="task_list_window" ></div>';
    $("#farm_frame").append(taskListWindow);
    var taskListFrame = '<div id="task_list_frame" ></div>';
    $("#task_list_window").append(taskListFrame);
    setTimeout( function()
    {
        $("#task_list_frame").append("<img src='../farm_buttons/task_list_title.gif' />");
    }

```

```

    for (i=0; i<tasks.length; i++)
    {
        if (tasks[i][TSK_DONE])
        {
            $("#task_list_frame").append("<br><span class='task_done_text'>" + tasks[i][
TSK_DESC] + "</span>");
        }
        else
        {
            $("#task_list_frame").append("<br><span class='task_item_text'>" + tasks[i][
TSK_DESC] + "</span>");
        }
    }
}, WINDOW_START_DELAY);

windowActive |= WA_TASKLIST;
logDebug("windowActive = " + windowActive);

$('#task_list_window').click(
    function(e)
    {
        var taskListWindowId = document.getElementById('task_list_window');
        taskListWindowId.parentNode.removeChild(taskListWindowId);
        logDebug("windowActive = " + windowActive);
        windowActive &= ~WA_TASKLIST;
        logDebug("windowActive = " + windowActive);
        return false;
    }
);
}

// -----
// displayHelp
// -----
function displayHelp()
{
    logDebug("displayHelp enter ...");
    killMenu();
    var help_window = '<div id="help_window" ></div>';
    $("#farm_frame").append(help_window);
    //var help_frame = '<div id="help_frame" ></div>';
    //$("#help_window").append(help_frame);
    $("#help_window").load("help.html");
    windowActive |= WA_HELP;
    $('#help_window').click(
        function(e)
        {
            var helpWindowId = document.getElementById('help_window');
            helpWindowId.parentNode.removeChild(helpWindowId);
            windowActive &= ~WA_HELP;
            return false;
        }
    );
};

```

```

    logDebug("displayHelp exit.");
}

// -----
//  displayHighScores
// -----
function displayHighScores()
{
    logDebug("displayHighScores enter ...");
    killMenu();
    var highscores_window = '<div id="highscores_window" ></div>';
    $("#farm_frame").append(highscores_window);
    highScoreHtml = '<div id="highscore_frame"><h3>High Scores</h3><table align="center"
width="60%">';
    for (i=0; i<highScores.length; i++)
    {
        highScoreHtml += '<tr><td align="left">' + highScores[i][HS_NAME] + '</td><td
align="right">' + highScores[i][HS_SCORE] + '</td></tr>';
    }
    highScoreHtml += '</table></div>';
    $("#highscores_window").append(highScoreHtml);
    windowActive |= WA_HIGHSCORES;
    $('#highscores_window').click(
        function(e)
        {
            var highscoresWindowId = document.getElementById('highscores_window');
            highscoresWindowId.parentNode.removeChild(highscoresWindowId);
            windowActive &= ~WA_HIGHSCORES;
            return false;
        }
    );
    logDebug("displayHighScores exit.");
}

// -----
//  displayMenu
// -----
function displayMenu()
{
    logDebug("displayMenu enter ...");
    var menu_window = '<div id="title_window" ></div>';
    $("#farm_frame").append(menu_window);
    var menu_frame = '<div id="title_frame_top" ></div>';
    $("#title_window").append(menu_frame);
    var menu = '<div id="menu" ></div>';
    $("#title_frame_top").append(menu);
    setTimeout( function()
    {
        $("#menu").append("<img src='../farm_buttons/game_title_text.gif'></img>");
        $("#menu").append("<div id='menu_task_list' class='menusel'></div>");
        $("#menu_task_list").append("<img src='../farm_buttons/task_list_text.gif'></img>");
        $('#menu_task_list').click(
            function(e)
            {

```

```

        displayTaskList ();
    }
};
$("#menu").append('<div id="save_game" class="menusel"></div>');
$("#save_game").append("<img src='../farm_buttons/save_game_text.gif'></img>");
$('#save_game').click(
    function(e)
    {
        processSaveGame ();
    }
);
$("#menu").append('<div id="menu_help" class="menusel"></div>');
$("#menu_help").append("<img src='../farm_buttons/help_text.gif'></img>");
$('#menu_help').click(
    function(e)
    {
        displayHelp ();
    }
);
$("#menu").append('<div id="menu_high_scores" class="menusel"></div>');
$("#menu_high_scores").append("<img src='../farm_buttons/high_scores_text.gif'></img>");
$('#menu_high_scores').click(
    function(e)
    {
        displayHighScores ();
    }
);
$("#menu").append('<div id="menu_back_to_farm" class="menusel"></div>');
$("#menu_back_to_farm").append("<img src='../farm_buttons/back_to_farm_text.gif'></img>");
$('#menu_back_to_farm').click(
    function(e)
    {
        killMenu ();
    }
);
}, WINDOW_START_DELAY);

windowActive |= WA_MENU;

logDebug("displayMenu exit.");
}

// -----
// killMenu
// -----
function killMenu()
{
    if (windowActive & WA_MENU)
    {
        var popUpId = document.getElementById('title_window');
        popUpId.parentNode.removeChild(popUpId);
        windowActive &= ~WA_MENU;
    }
}
}

```

```

// -----
// processMenuButtonClick
// -----
function processMenuButtonClick ()
{
    logDebug("Menu Button has been clicked, windowActive=" + windowActive);
    if ( windowActive == WA_NONE )
    {
        displayMenu();
    }
    else if ( windowActive & WA_INVENTORY )
    {
        killInventory();
        displayMenu();
    }
    else
    {
        killMenu();
    }
    return false;
}

// -----
// displayInformationSign
// -----
function displayInformationSign(textId)
{
    logDebug("displayInformationSign enter textId=" + textId);

    var infoWindow = '<div id="info_window" ></div>';
    $("#farm_frame").append(infoWindow);
    var infoFrame = '<div id="info_frame" ></div>';
    $("#info_window").append(infoFrame);
    setTimeout( function()
    {
        for (i=0; (i<info.length); i++)
        {
            //logDebug("displayInformationSign searching ID:" + info[i][FI_ID] + " text:" +
            info[i][FI_TEXT]);
            if (info[i][FI_ID] == textId)
            {
                $("#info_frame").append('<div id="info_text" >' + info[i][FI_TEXT] + '</div>');
                i = info.length;
            }
        }
    }, WINDOW_START_DELAY);

    windowActive |= WA_INFO;
    logDebug("windowActive = " + windowActive);

    $('#info_window').click(
        function(e)
        {

```

```

    var infoWindowId = document.getElementById('info_window');
    infoWindowId.parentNode.removeChild(infoWindowId);
    logDebug("windowActive = " + windowActive);
    windowActive &= ~WA_INFO;
    logDebug("windowActive = " + windowActive);
    return false;
  }
);
}

// -----
// questionObj
// -----
function questionObj(questionId)
{
  this.questionId = questionId;
  this.index = -1;
  this.correctAnswer = -1;

  for (i=0; (i<questions.length); i++)
  {
    if (questions[i][FQ_ID] == questionId)
    {
      this.index = i;
      this.correctAnswer = questions[i][FQ_CA];
      this.completed = questions[i][FQ_DONE];
      logDebug("questionObj - question completed is " + this.completed);

      // Adjust i to exit loop
      i = questions.length;
    }
  }
}

// -----
// killQuestion
// -----
function killQuestion()
{
  var questionWindowId = document.getElementById('question_window');
  questionWindowId.parentNode.removeChild(questionWindowId);
  windowActive &= ~WA_QUESTION;
}

// -----
// processAnswer
// -----
function processAnswer(userAnswer)
{
  if (currentQuestionObj.completed == 0)
  {
    logDebug("Question is being marked completed ...");
  }
}

```

```

questions[currentQuestionObj.index][FQ_DONE] = 1;

xmlhttp=new XMLHttpRequest();
xmlhttp.open("GET","gameUpdate.php?u=qtn&g=" + gameId +
              "&qid=" + currentQuestionObj.questionId, true);
xmlhttp.send();

if (userAnswer == currentQuestionObj.correctAnswer)
{
    updateScore(POINTS_QUESTION);
}

if (userAnswer == currentQuestionObj.correctAnswer)
{
    displayMessageWindow("Correct!", TIME_MESSAGE, 0);
}
else
{
    correction = 'Sorry, but the correct answer was \'';
    switch (currentQuestionObj.correctAnswer)
    {
        case 1: correction += 'A - ' + questions[currentQuestionObj.index][FQ_ANS_A] + '\'';
break;
        case 2: correction += 'B - ' + questions[currentQuestionObj.index][FQ_ANS_B] + '\'';
break;
        case 3: correction += 'C - ' + questions[currentQuestionObj.index][FQ_ANS_C] + '\'';
break;
        case 4: correction += 'D - ' + questions[currentQuestionObj.index][FQ_ANS_D] + '\'';
break;
    }
    displayMessageWindow(correction, TIME_WRONG_ANSWER, 0);
}
killQuestion();
}

// -----
// displayQuestionSign
// -----
function displayQuestionSign(textId)
{
    logDebug("displayQuestionSign enter textId=" + textId);

    currentQuestionObj = new questionObj(textId);

    var questionWindow = '<div id="question_window" ></div>';
    $("#farm_frame").append(questionWindow);
    var questionFrame = '<div id="question_frame" ></div>';
    $("#question_window").append(questionFrame);
    var answersFrame = '<div id="answers_frame" ></div>';
    $("#question_window").append(answersFrame);
    setTimeout( function()
    {

```

```

    $("#question_frame").append('<div id="question_text">' + questions[currentQuestionObj.
index][FQ_TEXT] + '</div>');
    $("#answers_frame").append('<div id="ansA" class="answer_text" style="top:0px;">A - ' +
questions[currentQuestionObj.index][FQ_ANS_A] + '</div>');
    $("#answers_frame").append('<div id="ansB" class="answer_text" style="top:64px;">B - ' +
questions[currentQuestionObj.index][FQ_ANS_B] + '</div>');
    $("#answers_frame").append('<div id="ansC" class="answer_text" style="top:128px;">C - '
+ questions[currentQuestionObj.index][FQ_ANS_C] + '</div>');
    $("#answers_frame").append('<div id="ansD" class="answer_text" style="top:192px;">D - '
+ questions[currentQuestionObj.index][FQ_ANS_D] + '</div>');
    $("#answers_frame").append('<div id="pass" class="answer_text" style="top:256px;">Pass
on this question for now</div>');

    // Mouse events for question
    $('#ansA').click( function(e) { processAnswer( 1 ); } );
    $('#ansB').click( function(e) { processAnswer( 2 ); } );
    $('#ansC').click( function(e) { processAnswer( 3 ); } );
    $('#ansD').click( function(e) { processAnswer( 4 ); } );
    $('#pass').click( function(e) { killQuestion(); return false; } );

    // Adjust i to exit loop
    i = questions.length;
}, WINDOW_START_DELAY);

windowActive |= WA_QUESTION;
logDebug("windowActive = " + windowActive);
}

// -----
// Inventory Object
// -----
function InventoryObject(click)
{
    logDebug("InventoryObject enter");
    this.inProximity = false;
    var i = 0;
    while ((i < farmObjects.length) && ( ! this.clicked ))
    {
        if (farmObjects[i][FO_LOC] == SCN_INVENTORY)
        {
            var inventoryItemIdName = "#inv_itm_" + i;
            var itemCenterX = $(inventoryItemIdName).offset().left + 25;
            var itemCenterY = $(inventoryItemIdName).offset().top + 25;
            logDebug("InventoryObject found inventory object: " + farmObjects[i]);
            logDebug("InventoryObject xtol: " + Math.abs(click.absX-itemCenterX));
            logDebug("InventoryObject ytol: " + Math.abs(click.absY-itemCenterY));
            if ( (Math.abs(click.absX-itemCenterX) < farmObjects[i][FO_CT]) &&
                (Math.abs(click.absY-itemCenterY) < farmObjects[i][FO_CT]) )
            {
                logDebug("InventoryObject was clicked");
                this.clicked = true;
                this.index = i;
            }
        }
    }
}

```

```

        i++;
    }
    logDebug("InventoryObject exit");
}

// -----
//  displayInventory
// -----
function displayInventory()
{
    var popup = '<div id="popup" >' + '</div>';
    $("#farm_frame").append(popup);
    setTimeout( function()
    {
        $("#popup").append("<img src='../farm_buttons/inventory_text.gif'></img>");
        var inventory = '<div id="inventory" >' + '</div>';
        $("#popup").append(inventory);
        for (i=0; i < farmObjects.length; i++)
        {
            if (farmObjects[i][FO_LOC] == SCN_INVENTORY)
            {
                var inventoryItemId = "inv_itm_" + i;
                //var inventoryItemId = farmObjects[i][FO_NAME];
                var inventoryItemIdName = "#" + inventoryItemId;
                var inventoryItem = '<div id="' + inventoryItemId + '" class="inventory_item"
title="' +
                                                                    farmObjects[i][
FO_HOVERTXT] + '>' + '</div>';
                logDebug("inventoryItem: " + inventoryItem);
                $("#inventory").append(inventoryItem);
                $(inventoryItemIdName).append("<img src='" + farmObjects[i][FO_FILE] +
''></img>");
                farmObjects[i][FO_X_CTR] = $(inventoryItemIdName).offset().left + 25;
                farmObjects[i][FO_Y_CTR] = $(inventoryItemIdName).offset().top + 25;
            }
        }
    }, WINDOW_START_DELAY);

    windowActive |= WA_INVENTORY;

    $('#popup').click(
        function(e)
        {
            var click = new MouseEvent(e);
            processInventoryClick(click);
            return false;
        }
    )
}

// -----
//  killInventory
// -----

```

```

function killInventory ()
{
    if (windowActive & WA_INVENTORY)
    {
        var popUpId = document.getElementById('popup');
        popUpId.parentNode.removeChild(popUpId);
        windowActive &= ~WA_INVENTORY;
    }
}

// -----
// processInventoryClick
// -----
function processInventoryClick (click)
{
    logDebug('processInventoryClick enter');
    var status = true;

    if (document.getElementById('inventory'))
    {
        var inventoryObject = new InventoryObject (click);
        if (inventoryObject.clicked)
        {
            logDebug("inventory object was clicked");
            var farmerPos = new FarmerPositionObject ();
            farmObjects[inventoryObject.index][FO_LOC] = currentScreen;
            farmObjects[inventoryObject.index][FO_X_POS] = farmerPos.xPos;
            farmObjects[inventoryObject.index][FO_Y_POS] = farmerPos.yPos;
            farmObjects[inventoryObject.index][FO_X_CTR] = farmerPos.xPos + 25;
            farmObjects[inventoryObject.index][FO_Y_CTR] = farmerPos.yPos + 25;
            //logDebug("Removing " + farmObjects[inventoryObject.index][FO_NAME]);
            //$('#' + farmObjects[inventoryObject.index][FO_NAME]).remove();
            var inventoryItemId = "inv_itm_" + inventoryObject.index;
            logDebug("Removing " + inventoryItemId);
            $('#' + inventoryItemId).remove();

            addItemToFarm(inventoryObject.index);
        }
        else
        {
            killInventory ();
        }
    }

    logDebug('processInventoryClick exit');
    return status;
}

// -----
// processInventoryButtonClick
// -----
function processInventoryButtonClick ()
{
    logDebug("Inventory Button has been clicked.");
}

```

```
if ( windowActive == WA_NONE )
{
    displayInventory();
}
else if ( windowActive & WA_MENU)
{
    killMenu();
    displayInventory();
}
else if ( windowActive & WA_INVENTORY)
{
    killInventory();
}
return false;
}

// -----
// processFarmFrameClick
// -----
function processFarmFrameClick (e)
{
    logDebug("Farm Frame has been clicked.");
    var click = new MouseClickObject(e);
    if ( windowActive == WA_NONE )
    {
        var farmObj = new FarmObject(click);
        if (farmObj.inProximity)
        {
            logDebug("click function processing Farm Item Type: " + farmObj.type);
            switch (farmObj.type)
            {
                case FOT_INVENTORY:
                    pickUpItem(farmObj);
                    break;
                case FOT_ANIMATION:
                    processAnimation(farmObj);
                    break;
                case FOT_DEBRIS:
                    removeDebris(farmObj);
                    break;
                case FOT_INFO:
                    displayInformationSign(farmObj.textId);
                    break;
                case FOT_QUESTION:
                    displayQuestionSign(farmObj.textId);
                    break;
            }
        }
        else
        {
            moveCharacterToMousePosition(click);
        }
    }
    else

```

```
{  
  logDebug("ERROR Missing click handler with windowActive is " + windowActive);  
}  
}
```

```
/* ----- */
/* NYCAMH Farm Safety Game
/* CSS File
/* SUNYIT
/* rdb - 2013
/* ----- */

#farmer
{
    position: absolute;
    background: url(./farm_images/farmer.gif) no-repeat;
    /* border: 1px solid red; */
    width: 60px;
    height: 95px;
    z-index: 1;
}

/* Farmer positions to animate walking and standing */

#farmer.front-right { background-position: -65px -295px; } /* lower left in gif */
#farmer.front-stand { background-position: -130px -295px; }
#farmer.front-left { background-position: -1px -295px; }

#farmer.left-right { background-position: -65px -194px; } /* upper left in gif */
#farmer.left-stand { background-position: -130px -194px; }
#farmer.left-left { background-position: -1px -194px; }

#farmer.back-right { background-position: -65px -1px; } /* upper right in gif */
#farmer.back-stand { background-position: -130px -1px; }
#farmer.back-left { background-position: -1px -1px; }

#farmer.right-right { background-position: -65px -99px; } /* lower right in gif */
#farmer.right-stand { background-position: -130px -99px; }
#farmer.right-left { background-position: -1px -99px; }

#farm_frame
{
    position: relative;
    border: 2px solid black;
    height: 600px;
    width: 800px;
    overflow: hidden;
    z-index: 0; /* negative z-index was disabling click function */
}

#farm
{
    position: absolute;
    /*background: url(./farm_images/grass.jpg) repeat; */
    background: url(./farm_images/grass_500.jpg) repeat;
    /* height: 4096px;
    width: 4096px; */
    height: 1536px;
    width: 2560px;
```

```
    z-index: -10;
}

#buttons_window
{
    position: relative;
    display: table;
    overflow: hidden;
    border: 2px solid black;
    background: sienna;
    height: 50px;
    width: 800px;
    z-index: 5;
}

#buttons_frame
{
    position: relative;
    display: table-cell;
    vertical-align: middle;
    width: 100%;
}

#inventory_button
{
    position: relative;
    top: -50%;
    float:left;
    background: url(./farm_buttons/inventory_button.gif) no-repeat;
    height: 40px;
    width: 160px;
    z-index: 10;
}

#menu_button
{
    position: relative;
    top: -50%;
    float:left;
    background: url(./farm_buttons/menu_button.gif) no-repeat;
    height: 40px;
    width: 160px;
    z-index: 10;
}

#score_text
{
    position: relative;
    top: -50%;
    float:right;
    /*border: 1px solid black; */
    /*background: url(./farm_buttons/score_text.gif) no-repeat;
    height: 40px;
    width: 100px; */
}
```

```
    color: black;
    margin-top: 7px;
    font-style: bold;
    font-size: 32px;
    font-family: "Lucida Console";
    z-index: 10;
}
```

```
#score
{
    position: relative;
    top: -50%;
    float:right;
    width: 120px;
    height: 100%;
    color: black;
    margin-top: 7px;
    font-style: bold;
    font-size: 32px;
    font-family: "Lucida Console";
    /* text-align:left; */
    /* border: 1px solid black; */
    z-index: 10;
}
```

```
#inventory
{
    position: absolute;
    left: 20px;
    top: 35px;
    width: 400px;
    height: 250px;
    z-index: -1;

    /* For Debug
    border-style:solid;
    border-width:1px; */
}
```

```
.inventory_item
{
    position: relative;
    width: 50px;
    height: 50px;
    z-index: -1;
    display:inline-block;

    /* For Debug
    border-style:solid;
    border-width:1px; */
}
```

```
#popup
{
```

```
position: absolute;
background: url(./farm_buttons/FarmLandscape_450px.jpg) no-repeat;
width: 450px;
height: 450px;
left: 50%;
top: 50%;
margin-left: -225px;
margin-top: -225px;
z-index: 10;
}

#message_window
{
    position: absolute;
background: url(./farm_buttons/small_message_window.jpg) no-repeat;
display: table;
overflow: hidden;
width: 400px;
height: 125px;
left: 50%;
top: 50%;
margin-left: -200px;
margin-top: -62px;
z-index: 10;
}

#message_frame
{
    position: absolute;
display: table-cell;
vertical-align: middle;
width: 100%;
margin: 0 auto;
text-align: center;
}

#message_text_center
{
    color: brown;
font-style: italic;
width: 100%;
height: 100%;
line-height: 125px;
}

#message_multiple_items
{
    color: brown;
font-style: italic;
height: 80%;
}

#title_window
```

```
{
    position: relative;
    background: url(./farm_buttons/FarmLandscape_450px.jpg) no-repeat;
    display: table;
    overflow: hidden;
    width: 450px;
    height: 450px;
    left: 50%;
    top: 50%;
    margin-left: -225px;
    margin-top: -225px;
    z-index: 10;
}

#title_frame_center
{
    position: absolute;
    display: table-cell;
    vertical-align: middle;
    width: 100%;
    margin: 0 auto;
    text-align:center;
}

#title_frame_top
{
    position: absolute;
    display: table-cell;
    vertical-align: top;
    width: 100%;
    margin: 0 auto;
    text-align:center;
}

.menusel
{
    padding: 5px;
}

.menusel:hover
{
    border-style: solid;
    border-width: 1px;
    border-color: black;
    background:red;
}

#task_list_window
{
    position: absolute;
    background: url(./farm_buttons/task_paper.gif) no-repeat;
    display: table;
    overflow: hidden;
    width: 300px;
}
```

```
    height: 400px;
    left: 50%;
    top: 50%;
    margin-left: -150px;
    margin-top: -200px;
    z-index: 12;
}
#task_list_frame
{
    position: absolute;
    display: table-cell;
    vertical-align: top;
    width: 100%;
    margin: 0 auto;
    text-align:center;
}

.task_item_text
{
    font-style: italic;
    font-size: 75%;
}

.task_done_text
{
    font-style: italic;
    text-decoration: line-through;
    font-size: 75%;
}

.button_links
{
    text-decoration: none;
    border: 0px;
}

#info_window
{
    position: absolute;
    background: url(../farm_buttons/sign450px.jpg) no-repeat;
    display: table;
    overflow: hidden;
    width: 450px;
    height: 450px;
    left: 50%;
    top: 50%;
    margin-left: -225px;
    margin-top: -225px;
    z-index: 12;
}
#info_frame
{
    position: absolute;
    display: table-cell;
    vertical-align: top;
```

```
width: 85%;
margin: 0 auto;
margin-left: 25px;
margin-top: 30px;
text-align:center;
}

#info_text
{
    color: brown;
    font-style: italic;
    font-size: 18px;
}

#help_window
{
    position: absolute;
    overflow: auto;
    background: tan;
    width: 450px;
    height: 450px;
    left: 50%;
    top: 50%;
    margin-left: -225px;
    margin-top: -225px;
    border-style:solid;
    border-width:2px;
    z-index: 12;
}

#help_frame
{
    position: absolute;
    vertical-align: top;
    width: 95%;
    margin: 0 auto;
    margin-left: 5px;
    margin-top: 5px;
    text-align:left;
}

#highscores_window
{
    position: absolute;
    display: table;
    overflow: hidden;
    background: tan;
    width: 450px;
    height: 450px;
    left: 50%;
    top: 50%;
    margin-left: -225px;
    margin-top: -225px;
    border-style:solid;
    border-width:2px;
```

```
    z-index: 12;
}
#highscore_frame
{
    position: absolute;
    display: table-cell;
    vertical-align: top;
    width: 95%;
    margin: 0 auto;
    margin-left: 5px;
    margin-top: 5px;
    text-align:center;
    font-style: bold;
    font-size: 24px;
}

#question_window
{
    position: relative;
    background: url(./farm_buttons/sign450px.jpg) no-repeat;
    overflow: hidden;
    width: 450px;
    height: 450px;
    left: 50%;
    top: 50%;
    margin-left: -225px;
    margin-top: -225px;
    z-index: 12;
}

#question_frame
{
    position: absolute;
    width: 400px;
    height: 80px;
    left: 20px;
    top: 20px;
    text-align: center;
/* border-style: solid;
border-width: 1px;
border-color: black; */
}

#question_text
{
    color: brown;
    font-style: italic, bold;
    font-size: 20px;
    padding: 10px;
}

#answers_frame
{
    position: absolute;
    left:20px;
```

```
    top:100px;
    width: 400px;
    height: 320px;
/* border-style: solid;
border-width: 1px;
border-color: red; */
}

.answer_text
{
    position: absolute;
    width: 450px;
    height: 90px;
    margin-left: 50px;
    color: brown;
    font-style: italic;
    font-size: 14px;
    text-align:left;
    padding: 10px;
}

.answer_text:hover
{
    font-style: bold;
    font-size: 18px;
    color: green;
    cursor: pointer;
/* border-style:solid;
border-width:1px; */
}
```

```
<?php
// -----
// FILE: farmData.php
//
// DESCRIPTION:
// This file is used to read farm information from the mySQL database.
// -----

include_once("databaseConnect.php");

// -----
// FarmData class
// -----

class FarmData extends DatabaseConnect
{
    private $imageDirectory = "./farm_images/";

    // -----
    // FUNCTION: readFarmObjects
    //
    // DESCRIPTION:
    // Reads the farm objects into an array for the specified game id.
    // -----

    public function readFarmObjects($id)
    {
        $farmObjects = array();

        $q = "SELECT * FROM farmObjects WHERE gameId=$id";
        $result = mysql_query($q, $this->mysqlConnection);
        if ($result)
        {
            //$numItems = mysqli_num_rows($result);
            $numItems = mysql_num_rows($result);
            if ($numItems > 0)
            {
                while ($row = mysql_fetch_array($result))
                {
                    $frmObj = new FarmObject();
                    $frmObj->name = $row[name];
                    $frmObj->filename = $this->imageDirectory . $row[filename];
                    $frmObj->type = $row[type];
                    $frmObj->width = $row[width];
                    $frmObj->height = $row[height];
                    $frmObj->xPos = $row[xPos];
                    $frmObj->yPos = $row[yPos];
                    $frmObj->xCtr = $frmObj->xPos + ($frmObj->width/2);
                    $frmObj->yCtr = $frmObj->yPos + ($frmObj->height/2);
                    $frmObj->proximityTolerance = $row[prxTol];
                    $frmObj->clickTolerance = $row[clkTol];
                    $frmObj->screenNumber = $row[screenNum];
                    $frmObj->inventoryKey = $row[inventoryKey];
                    $frmObj->animation = $row[animation];
                    $frmObj->obstructs = $row[obstructs];
                    $frmObj->textId = $row[textId];
                }
            }
        }
    }
}
```

```
        $frmObj->hoverTxt = $row[hoverTxt];
        $farmObjects[] = $frmObj;
    }
}
else
{
    echo "Query failure reading farmObjects list<br>";
}

return $farmObjects;
}

// -----
// FUNCTION:   readAnimations
//
// DESCRIPTION:
//     Reads the animations from the database into an array.
// -----
public function readAnimations()
{
    $animations = array();

    $q = "SELECT * FROM farmAnimations";
    $result = mysql_query($q, $this->mySqlConnection);
    if ($result)
    {
        // $numItems = mysqli_num_rows($result);
        $numItems = mysql_num_rows($result);
        if ($numItems > 0)
        {
            // while ($row = mysqli_fetch_array($result))
            while ($row = mysql_fetch_array($result))
            {
                $aniObj = new AnimationObject();
                $aniObj->targetObjectName = $row[targetObj];
                $aniObj->locationObjectName = $row[locationObj];
                $aniObj->removeObjectName = $row[removeObj];
                $aniObj->debrisObjectName = $row[debrisObj];
                $aniObj->animationName = $row[animationName];
                $aniObj->taskId = $row[taskId];
                $animations[] = $aniObj;
            }
        }
    }
    else
    {
        echo "Query failure reading animations list<br>";
    }

    return $animations;
}

// -----
```

```
// FUNCTION:   readFarmInfo
//
// DESCRIPTION:
//   Reads the farm info text from the database into an array.
// -----
public function readFarmInfo()
{
    $info = array();

    $q = "SELECT textId, text FROM farmInfo";
    $result = mysql_query($q, $this->mySqlConnection);
    if ($result)
    {
        $numItems = mysql_num_rows($result);
        if ($numItems > 0)
        {
            while ($row = mysql_fetch_array($result))
            {
                $infoObj = new FarmInfo();
                $infoObj->text = $row[text];
                $infoObj->textId = $row[textId];
                $info[] = $infoObj;
            }
        }
    }
    else
    {
        echo "Query failure reading animations list<br>";
    }

    return $info;
}

// -----
// FUNCTION:   readFarmQuestions
//
// DESCRIPTION:
//   Reads the questions from the database into an array.
// -----
public function readFarmQuestions()
{
    $questions = array();

    $q = "SELECT * FROM farmQuestions";
    $result = mysql_query($q, $this->mySqlConnection);
    if ($result)
    {
        $numItems = mysql_num_rows($result);
        if ($numItems > 0)
        {
            while ($row = mysql_fetch_array($result))
            {
                $qtn = new FarmQuestion();
                $qtn->questionId = $row[questionId];
            }
        }
    }
}
```

```

        $qtn->text = $row[text];
        $qtn->answerA = $row[answerA];
        $qtn->answerB = $row[answerB];
        $qtn->answerC = $row[answerC];
        $qtn->answerD = $row[answerD];
        $qtn->correctAns = $row[correctAns];
        $questions[] = $qtn;
    }
}
}
else
{
    echo "Query failure reading farm questions<br>";
}

return $questions;
}

// -----
// FUNCTION:    readCompletedQuestions
//
// DESCRIPTION:
//     Reads the list of completed questions from the database into an
//     array.
// -----
public function readCompletedQuestions($id)
{
    $completed = array();

    $q = "SELECT * FROM completedQuestions WHERE gameId=$id";
    $result = mysql_query($q, $this->mysqlConnection);
    if ($result)
    {
        $numItems = mysql_num_rows($result);
        if ($numItems > 0)
        {
            while ($row = mysql_fetch_array($result))
            {
                $completed[] = $row[questionId];
            }
        }
    }
    else
    {
        echo "Query failure reading completed questions<br>";
    }

    return $completed;
}

// -----
// FUNCTION:    readTaskList
//
// DESCRIPTION:

```

```
// Reads the task list from the database into an array.
// -----
public function readTaskList($id)
{
    $taskList = array();

    $q = "SELECT complete, taskId FROM taskList WHERE gameId=$id";
    $result = mysql_query($q, $this->mySqlConnection);
    if ($result)
    {
        $numItems = mysql_num_rows($result);
        if ($numItems > 0)
        {
            while ($row = mysql_fetch_array($result))
            {
                $task = new Task();
                $task->complete = $row[complete];
                $task->taskId = $row[taskId];
                $q = "SELECT taskDescription FROM taskText WHERE taskId=$task->taskId";
                $result2 = mysql_query($q, $this->mySqlConnection);
                if ($result2)
                {
                    $numItems = mysql_num_rows($result2);
                    if ($numItems == 1)
                    {
                        $row2 = mysql_fetch_array($result2);
                        $task->taskDescription = $row2[taskDescription];
                    }
                    else
                    {
                        echo "Invalid number of entries found: " . $numItems . "<br>";
                    }
                }
                else
                {
                    echo "Error reading task text table.<br>";
                }
                $taskList[] = $task;
            }
        }
        else
        {
            echo "Task List Table was empty.<br>";
        }
    }
    else
    {
        echo "Query failure reading taskList table<br>";
    }

    return $taskList;
}

// -----
```

```
// FUNCTION:   readScore
//
// DESCRIPTION:
//     Gets the score for the specified game.
// -----
public function readScore($id)
{
    $score = 0;
    $q = "SELECT score FROM games WHERE gameId=$id";
    $result = mysql_query($q, $this->mySqlConnection);
    if ($result)
    {
        $numItems = mysql_num_rows($result);
        if ($numItems == 1)
        {
            $row = mysql_fetch_array($result);
            $score = $row['score'];
        }
    }
    else
    {
        echo "Query failure reading score<br>";
    }

    return $score;
}

// -----
// FUNCTION:   getHighScores
//
// DESCRIPTION:
//     Retrieves the 10 top scores from the database.
// -----
public function getHighScores()
{
    $highScores = array();
    $q = "SELECT score, username FROM games ORDER BY score DESC";
    $result = mysql_query($q, $this->mySqlConnection);
    if ($result)
    {
        $i = 0;
        while (($row = mysql_fetch_array($result)) && (i < 10))
        {
            $score = new HighScore();
            $score->name = $row['username'];
            $score->score = $row['score'];
            $highScores[] = $score;
            $i++;
        }
    }
    else
    {
        echo "Query failure reading score<br>";
    }
}
```

```
    return $highScores;
}

// -----
// FUNCTION:   __construct
//
// DESCRIPTION:
//     This function is the constructor for the BaseApi class.
// -----
public function __construct()
{
    $this->databaseConnect();
}

// -----
// FUNCTION:   __destruct
//
// DESCRIPTION:
//     The destructor
// -----
public function __destruct()
{
    $this->databaseDisconnect();
}
}
```

```
?>
```

```
<?php
// -----
// FILE: GameUpdate.php
//
// DESCRIPTION:
//     This file is used for AJAX calls to save information into the mySQL
//     database without the need of re-loading the HTML document.
// -----

include_once("databaseConnect.php");

// -----
// GameUpdate class
// -----

class GameUpdate extends DatabaseConnect
{
    private $debugOn = false;

    // -----
    // FUNCTION:    logAjaxDebug
    //
    // DESCRIPTION:
    //     Used for debugging purposes. If debug is turned on, it will write
    //     the specified debug strings to a file called "ajax_debug.log" on the
    //     web server.
    // -----

    private function logAjaxDebug($debugStr)
    {
        if ($this->debugOn)
        {
            $handle = fopen('ajax_debug.log', 'a');
            $dateString = date(DATE_ATOM, time());
            $writeString = $dateString . " => " . $debugStr . "\n";
            fwrite($handle, $writeString );
            fclose($handle);
        }
    }

    // -----
    // FUNCTION:    updateObject
    //
    // DESCRIPTION:
    //     This function is used to update a farm object for a specified game.
    //     It may have changed position on the farm, or is in inventory, or it has
    //     been removed from the farm (like the rock in the long grass).
    // -----

    public function updateObject($id, $name, $x, $y, $s, $t)
    {
        $q = "UPDATE farmObjects SET xPos=$x, yPos=$y, screenNum=$s, type=$t WHERE gameId=$id
AND name='$name'";
        $this->logAjaxDebug($q);
        $result = mysql_query($q, $this->mySqlConnection);
        if ( ! $result)
        {
```

```
        $this->logAjaxDebug("Error performing object position update.");
    }
}

// -----
// FUNCTION:    updateTaskList
//
// DESCRIPTION:
//     This function will update a task in the task list for a specified
//     game.  It basically is used to mark a task complete.
// -----
public function updateTaskList($gid, $tid)
{
    $q = "UPDATE taskList SET complete=1 WHERE taskId=$tid AND gameId=$gid";
    $this->logAjaxDebug($q);
    $result = mysql_query($q, $this->mySqlConnection);
    if ( ! $result)
    {
        $this->logAjaxDebug("Error performing task list update.");
    }
}

// -----
// FUNCTION:    updateCompletedQuestions
//
// DESCRIPTION:
//     This function is used to mark a question completed (right or
//     wrong).  Users are only given one chance to answer the question for
//     points.  They can answer the question a second time but will not
//     receive points if the question is initially answered wrong.
// -----
public function updateCompletedQuestions($gid, $qid)
{
    $q = "INSERT INTO completedQuestions (gameId, questionId) VALUES ($gid, $qid)";
    $this->logAjaxDebug($q);
    $result = mysql_query($q, $this->mySqlConnection);
    if ( ! $result)
    {
        $this->logAjaxDebug("Error inserting a new recorded into the completed questions
table.");
    }
}

// -----
// FUNCTION:    updateScore
//
// DESCRIPTION:
//     Updates the score for the specified game.
// -----
public function updateScore($id, $pts)
{
    $q = "UPDATE games SET score=$pts WHERE gameId=$id";
    $this->logAjaxDebug($q);
    $result = mysql_query($q, $this->mySqlConnection);
}
```

```

    if ( ! $result)
    {
        $this->logAjaxDebug("Error performing object position update.");
    }
}

// -----
// FUNCTION:   __construct
//
// DESCRIPTION:
//     This function is the constructor for the BaseApi class.
// -----
public function __construct($debug)
{
    $this->debugOn = $debug;
    $this->databaseConnect();
}

// -----
// FUNCTION:   __destruct
//
// DESCRIPTION:
// -----
public function __destruct()
{
    $this->databaseDisconnect();
}
}

$debug = false;
if (isset($_GET['debug']))
{
    $debug = true;
}

$game=$_GET["g"];
$update=$_GET["u"];

switch($update)
{
    case "pos":
        $n = $_GET["n"];    // name
        $x = $_GET["x"];    // x pos
        $y = $_GET["y"];    // y pos
        $s = $_GET["s"];    // screen location
        $t = $_GET["t"];    // type
        $update = new GameUpdate($debug);
        $update->updateObject($game, $n, $x, $y, $s, $t);
        break;
    case "state":
        break;
    case "score":
        $pts = $_GET["pts"];
        $update = new GameUpdate($debug);

```

```
$update->updateScore($game, $pts);  
break;  
case "task":  
    $tid = $_GET["tid"];  
    $update = new GameUpdate($debug);  
    $update->updateTaskList($game, $tid);  
    break;  
case "qtn":  
    $qid = $_GET["qid"];  
    $update = new GameUpdate($debug);  
    $update->updateCompletedQuestions($game, $qid);  
    break;  
  
default:  
    break;
```

}

?>

```
<?php

// -----
// FILE: databaseConnect.php
//
// DESCRIPTION:
//     This file contains the base class for any class that requiring
// database access. Parent classes will just "extend" this class. This
// class is not intended to be instantiated directly.
// -----

class DatabaseConnect
{
    protected $mySqlConnection;

    // -----
    // FUNCTION:  databaseConnect
    //
    // DESCRIPTION:
    //     This function is responsible for opening a connection to the
    // cleancutterdata mySQL database.
    // -----

    protected function databaseConnect()
    {
        $this->mySqlConnection = mysql_connect(
            'localhost',           // host
            'xxxxxxxxxxxxxxxx',    // user
            'xxxxxxx',            // password
            false,                 // newlink
            0);                    // clientFlag

        if (!$this->mySqlConnection)
        {
            echo "mySQL database connect failed!<BR>";
        }
        else if ( ! mysql_select_db('gbabycre_farmsafety' ) )
        //else if ( ! mysql_select_db('begleyr' ) )
        {
            echo "mySQL error selecting database!<BR>";
        }
    }

    protected function databaseDisconnect()
    {
        if ($this->mySqlConnection)
        {
            mysql_close($this->mySqlConnection);
        }
    }
}

?>
```

```
<?php
// -----
// FILE: createdb.php
//
// DESCRIPTION:
//     This file initializes the Play It Safe farm safety database. It opens
// the createdb.sql file and reads each line out of the file and executes
// each mySQL command in sequence.
// -----

include_once("databaseConnect.php");

class CreateDB extends DatabaseConnect
{

    // -----
    // FUNCTION:   __construct
    //
    // DESCRIPTION:
    //     This function is the constructor for the BaseApi class.
    // -----

    public function __construct()
    {
        $this->databaseConnect();
    }

    public function runSQLFile($location)
    {
        //load file
        $commands = file_get_contents($location);

        //delete comments
        $lines = explode("\n", $commands);
        $commands = '';
        foreach($lines as $line){
            $line = trim($line);
            //
            // != is not a typo. We can't use != because the
            // function will return false if the string does not exist
            // and (false == 0) after type juggling. So === and !=
            // mean "identical" values and types.
            //
            if( $line && (strpos($line, '--')!==0) )
            {
                $commands .= $line . "\n";
            }
        }

        //convert to array
        $commands = explode(";", $commands);

        //run commands
        $total = $success = 0;
        foreach($commands as $command){
```

```
        if(trim($command)){
            $success += (@mysql_query($command)==false ? 0 : 1);
            $total += 1;
        }
    }

    //return number of successful queries and total number of queries found
    return array(
        "success" => $success,
        "total" => $total
    );
}
}
```

```
echo "Creating Farm Safety Game database ... <br>";
$farmDB = new CreateDB();
$results = $farmDB->runSQLFile("createdb.sql");
echo "Total commands: " . $results["total"] . "<br>";
echo "Successful commands: " . $results["success"] . "<br>";
```

```
?>
```

```

--
-- Database Creation Script
--
-- Run by doing: mysql -h db -u begleyr -p < create_database.sql
-- Or by running: createdb.php
--
-- use begleyr;
use gbabycre_farmsafety;
drop table if exists farmObjects;
create table farmObjects ( gameId INT, name VARCHAR(20), filename VARCHAR(25), type INT, width INT, height INT,
xPos INT, yPos INT, prxTol INT, clkTol INT, screenNum INT, inventoryKey INT,
animation INT, obstructs INT, textId INT, hoverTxt VARCHAR(20));
--
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "farmer", "", 0, 0, 0, 355, 185, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "gloves", "gloves_50.gif", 2, 50, 50, 1600, 450, 40, 30, -1, 0x04, -1, 0, -1, "Gloves");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "cow", "cow.gif", 1, 128, 125, 650, 1375, 0, 0, 1, 0x00, -1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "boots", "boots_50.gif", 2, 50, 50, 1600, 20, 40, 30, 1, 0x02, -1, 0, -1, "Leather
Boots");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "dustmask", "dustmask.gif", 2, 50, 50, 300, 600, 40, 30, -1, 0x10, -1, 0, -1, "Dust
Mask");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "breathingapp", "breathingapp.gif", 2, 50, 50, 950, 10, 40, 30, 1, 0x40, -1, 0, -1,
"Breathing Apparatus");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "sunscreen", "sunscreen.gif", 2, 50, 50, 980, 660, 40, 30, 1, 0x80, -1, 0, -1, "Sun
Screen");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "rubberboots", "rubberboots.gif", 2, 50, 50, 900, 200, 40, 30, -1, 0x20, -1, 0, -1,

```

```
"Rubber Boots");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "goggles", "goggles_50.gif", 2, 50, 50, 2150, 10, 40, 30, 1, 0x01, -1, 0, -1, "Eye
Protection");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "earprot", "earprot.gif", 2, 50, 50, 1200, 1200, 40, 30, 1, 0x08, -1, 0, -1, "Ear
Protection");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "grass", "long_grass.gif", 5, 200, 200, 800, 101, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "rock", "rock.gif", 4, 30, 20, 850, 150, 20, 10, 1, 0x00, -1, 0, -1, "Rock");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "tractor", "tractor.gif", 3, 100, 73, 800, 25, 80, 25, 1, 0x8b, 0, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "fence", "fence.gif", 1, 512, 67, 1024, 1024, 0, 0, 1, 0x00, -1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "rooster", "rooster.gif", 1, 50, 43, 700, 600, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "chicken", "chicken.gif", 1, 50, 48, 730, 700, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "henhouse", "henhouse.gif", 1, 170, 136, 800, 600, 0, 0, 1, 0x00, -1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "farmhouse", "farmhouse.gif", 1, 783, 200, 1, 1, 0, 0, 1, 0x00, -1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "crops", "crops.gif", 1, 512, 512, 1025, 1, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "shrub", "shrub.gif", 1, 100, 75, 530, 230, 0, 0, 1, 0x00, -1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
```

```

VALUES (0, "vertwalk", "vertwalk.gif", 1, 100, 276, 345, 200, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "curvewalk", "curvewalk.gif", 1, 240, 240, 210, 476, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "mailbox", "mailbox.gif", 1, 50, 75, 180, 532, 0, 0, 1, 0x00, -1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "road", "road.gif", 1, 200, 600, 1, 380, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "driveway", "driveway.gif", 1, 260, 282, 1, 200, 0, 0, 1, 0x00, -1, 0, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "inf01", "infosign.gif", 6, 60, 75, 450, 240, 100, 50, 1, 0x00, -1, 1, 1, "Safety Info"
);
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "info2", "infosign.gif", 6, 60, 75, 735, 230, 100, 50, 1, 0x00, -1, 1, 2, "Safety Info"
);
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "silo", "silo.gif", 3, 450, 350, 1650, 1, 175, 150, 1, 0x40, 1, 1, -1, "");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "info3", "infosign.gif", 6, 60, 75, 1700, 360, 100, 50, 1, 0x00, -1, 1, 3, "Safety
Info");
INSERT INTO farmObjects ( gameId, name, filename, type, width, height, xPos, yPos, prxTol, clkTol, screenNum,
inventoryKey, animation, obstructs, textId, hoverTxt)
VALUES (0, "question1", "questionsign.gif", 7, 60, 75, 2150, 1200, 100, 50, 1, 0x00, -1, 1, 1,
"Question");
--
drop table if exists farmAnimations;
create table farmAnimations ( targetObj VARCHAR(20), locationObj VARCHAR(20), removeObj VARCHAR(20), debrisObj VARCHAR(20),
animationName VARCHAR(20), taskId INT );
INSERT INTO farmAnimations ( targetObj, locationObj, removeObj, debrisObj, animationName, taskId )
VALUES ("tractor", "grass", "grass", "rock", "mowing", 1);
INSERT INTO farmAnimations ( targetObj, locationObj, removeObj, debrisObj, animationName, taskId )
VALUES ("silo", "silo", "", "", "silo", 2);
--

```

```
drop table if exists taskText;
create table taskText (taskId INT, taskDescription VARCHAR(120), PRIMARY KEY (taskId));
INSERT INTO taskText ( taskId, taskDescription ) VALUES (1, "Mow patch of grass on side of the house.");
INSERT INTO taskText ( taskId, taskDescription ) VALUES (2, "Check the grain in silo.");
--
drop table if exists taskList;
create table taskList (gameId INT, taskId INT, complete INT);
--
drop table if exists games;
create table games (gameId INT NOT NULL AUTO_INCREMENT, username VARCHAR(40), password VARCHAR(15),
                    score INT, PRIMARY KEY (gameId));
--
drop table if exists farmInfo;
create table farmInfo (textId INT, text VARCHAR(1024), PRIMARY KEY (textId));
INSERT INTO farmInfo ( textId, text )
VALUES (1, "Measures should be taken to protect your skin from UV radiation whether it's "
"sunny or cloudy as clouds do not stop burn producing UV radiation.<br><br>"
"Wearing a hat with a wide brim, sunscreen with an SPF (sun protection factor) "
"of 15 or higher, sunglasses with 100% UVA and UVB (Ultraviolet radiation A and B) "
"protection, and a shirt with long sleeves is highly recommended. <br><br>"
"Find the sunscreen and add it to your inventory.");
INSERT INTO farmInfo ( textId, text )
VALUES (2, "Mowing the lawn? <br><br>Do a quick survey of the area to be mowed. Check for "
"objects like garden hoses, wire, tree branches, rocks etc. and move them to a safe "
"location. It's also important to ask children (and adults) to leave the area where "
"you are going to be mowing. You do not want to risk injuring them by something thrown "
"from the mower.<br><br>This task will require sturdy leather shoes/boots. Leather "
"sneakers are not acceptable. Sturdy leather shoes will help to prevent or lessen "
"the severity of injury from objects flung out from under the mower. You also need "
"safety glasses, hearing protection (plugs or muffs), and long pants");
INSERT INTO farmInfo ( textId, text )
VALUES (3, "Nothing except a self contained breathing apparatus (like what firemen wear) will protect "
"you in the silo. You must stay out after filling the silo for at least 2 weeks. Silo "
"gas can build up to dangerous levels within 4-6 hours after filling the last load so "
"the silage should be leveled immediately after putting in the last load. Breathing silo "
"gas, in high concentrations, can kill immediately or it can cause serious burn-like "
"injury to the lungs. Silo gas can cause breathing difficulties (like shortness of breath) "
"6-8 weeks after exposure as the lungs heal. These later breathing difficulties can be life "
"threatening. It is very important to see a physician if you think you may have breathed in "
"some silo gas so you can be monitored and treated for possible future breathing problems ");
--
```

```
drop table if exists farmQuestions;
create table farmQuestions (questionId INT, text VARCHAR(512), answerA VARCHAR(80), answerB VARCHAR(80),
                           answerC VARCHAR(80), answerD VARCHAR(80), correctAns INT, PRIMARY KEY (questionId));
INSERT INTO farmQuestions ( questionId, text, answerA, answerB, answerC, answerD, correctAns )
  VALUES (1, "How long should you remain out of the Silo after it has been filled?",
          "2 Hours", "2 Days", "2 Weeks", "2 Months", 3);
--
drop table if exists completedQuestions;
create table completedQuestions (idx INT NOT NULL AUTO_INCREMENT, gameId INT, questionId INT, PRIMARY KEY (idx));
```

VITA

Robert D. Begley (b. 1960) graduated from Potsdam State University in 1983 with a Bachelor of Arts in Computer Science. He worked for Sperry Univac in Salt Lake City, Utah as an Associate Programmer from 1983 to 1985 writing Assembly Language for smart terminals and DOS device drivers for a port of the then popular MS-DOS operating system to Sperry terminal devices. He then moved back to New York and worked for Eastman Kodak Company as a Service Engineer and a Software Engineer from 1985 to 1998 where the bulk of his time was spent writing embedded real time motor control software for various Kodak film handling devices. He worked for Gretag Imaging from 1998 to 2000 as a Senior Software Engineer where he continued to support embedded software for a large Kodak photofinishing printer. At Gretag, he also began working as a Windows developer. He worked for Xerox Corporation from 2001 to 2011 as a Software Engineer. In 2011 he began working for HCL America where he continues to support Xerox products as a Senior Software Engineer and currently works on a Solaris based digital front end for large Xerox print engines.