Creating Educational iOS Games for Elementary School Students

A Thesis for Departmental Honors Submitted to the
University of Mary Washington

By
Heather Martin

Faculty Advisor
Dr. Jennifer Polack-Wahl

April 2012

Approval by Committee

This thesis has been submitted for approval by a select committee of the UMW department of computer science.

_____ _____

Dr. Jennifer Polack-Wahl Date

_____ _____

Dr. Ernest C. Ackermann Date

_____ _____

Dr. Karen Anewalt Date

Table of Contents

Abstract

The purpose of this honors project was to find evidence to support the hypothesis that students who use educational applications in conjunction with traditional classroom instruction improve their understanding of the subject at a faster rate than more traditional methods. It has been theorized that due to the rise of digital media in society, conventional classroom instruction is not as effective on younger generations; whereas educational games serve as a medium to both engage and teach the modern student. Due to the popularity of iOS applications, the educational games created for this research project were developed for that platform. Students involved in this research project became versed in iOS development; they learned how to program in Objective-C, navigate Xcode, use Interface Builder, and combined this knowledge to produce applications. The resulting applications were submitted to iTunes and released for free. Remaining time was spent conducting research and experiments to determine the effectiveness of educational games. Early testing results support the hypothesis. However, a larger sample size is required to reach a definitive answer.

1. Background Research

Smartphone sales are increasing dramatically; 491.4 million units were sold in 2011, which is up 61% from 2010 ("Global Mobile Statistics 2012"). Due to this influx in smartphone users, our applications are available to a global audience. iPads and iTouches are also becoming increasingly popular as classroom tools (Hu). The rise in the use of iPads/iTouches is due to their intuitive interface, which makes them easy for children to use and understand (Mallinson). The combined benefits of having intuitive technology, a platform for educational software, current use in the classroom and a wide global audience condensed into one technology make the iOS the practical choice for our research group to use for development.

The reason for the sudden rise in the use of iPads/iTouches is due to the theory that educational games are able to engage young students in ways conventional teaching tactics have been unable to do (Klopfer et al.). Studies have shown that the use of this technology prompts students to become active participants in learning the subject matter (Watson). There are many possible reasons that explain why games are so successful at educating students. They combine story, art, and software to create an interactive learning experience that has the potential to revolutionize the way we learn (Zyda).

2. General Overview of Applications

The next section describes the two applications I created for this project. It addresses the purpose, educational benefit, and overall concept of the applications.

2.1 Space Spell

Development on *Space Spell* started during the spring semester of 2011 and ended with the conclusion of the ensuing Summer Science Institute. *Space Spell* was a joint project undertaken by Stephen Jackson and myself. The original idea came from a teacher who requested an application to teach students to read by using words from the Dolch sight list. Children who are able to recognize words from the Dolch list improve their reading abilities ("Teaching Children Reading Begins with Dolch Sight Words"). During the later stages of development it was requested that "word families" be added in conjunction with the Dolch list.

The overall concept of the application is that the player is given a scrambled word which he must unscramble. This is done by dragging the letters into the correct screen position in order to spell the word [Figure 1]. When the game begins the player is able to choose which list of words he would like to use; either from the Dolch list or one of the "word families". After a list is chosen, the player is presented with a scrambled word from that list accompanied by an audio sample of the word. The player must then drag the rockets containing the letters to the correct docking station in order to spell the word. When the player believes the word is spelled correctly, he presses the submit button. If the player is correct, the game moves to the next word in the list. Otherwise, the letters placed in the incorrect positions are popped out of the docking station and the player must try again.

Figure 1: Screenshot of *Space Spell*

2.2 Rise of Pharaoh

The inspiration for *Rise of Pharaoh* began as a result of a longstanding request by an educator for an educational history application. We chose ancient Egypt as the subject matter because 2nd graders in Virginia are required to learn about it in order to pass their SOL's. Idea conceptualization of *Rise of Pharaoh* began during the Summer Science Institute of 2011 but no idea was definitively decided upon until the beginning of the fall 2011 semester.

The overall concept of the application is that the player starts on one of the lowest rungs on the ancient Egyptian social ladder, a pyramid builder, and his goal is to reach the highest rung, the pharaoh. The player accomplishes this goal by successfully completing the mini-game associated with his current social standing. There are two game modes; story mode and arcade mode. Story mode includes a narration as the player progressively beats each mini-game on his way to becoming pharaoh. Arcade mode has no narration but instead allows the player to pick and choose which mini-game he wants to play. There are four mini-games, one for each of the four social rankings, the player must complete before becoming pharaoh; pyramid builder, craftsman, scribe, and priest. To advance to the next social

ranking the player must fill up his experience bar. Selecting correct answers in succession allows the player to earn combo points which fills up his experience bar faster.

The player begins his quest to pharaoh with the pyramid builder game. The pyramid builder game is an interactive multiple choice quiz. The goal of the game is to build pyramids; bricks are added to the pyramid as the player correctly answers questions. The questions used were taken directly from 2nd grade SOL preparation tests created by educators. To make the game more dynamic a mummy feature was added; every five to ten seconds a mummy attacks one of the bricks on the pyramid and the player must tap that brick in order to defeat the mummy. If the player fails to defeat a mummy, or if he answers a question incorrectly, a brick is removed from the pyramid and his combo multiplier is reset.

The next game in story mode is the craftsman mini-game. The objective is to learn about different artifacts that were present in ancient Egypt and their importance to Egyptian society. The concept of the game is similar to the popular console game *Guitar Hero*. The player has two buttons, a purple circle and an orange circle, that he can interact with on the right hand side of the screen. The left side of the screen shows a random pattern of those symbols which light up one at a time. The player is required to hit the correct button when it lights up to build the artifact successfully [Figure 2.1]. As the player gets the pattern correct he improves the quality of the artifact. Once the artifact is crafted, the narrator explains the importance of that artifact in ancient Egyptian society [Figure 2.2].

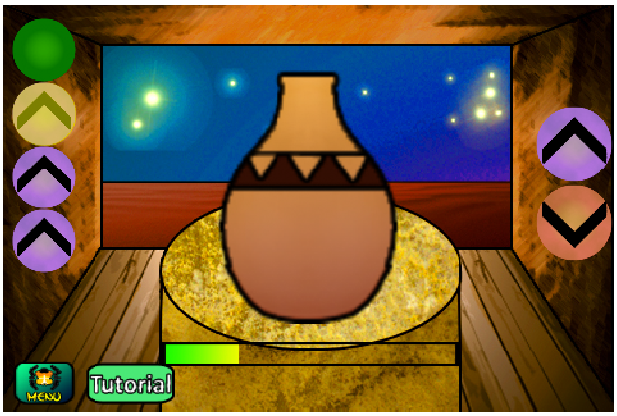

Figure 2.1: Incomplete artifact in craftsman mini-game

Figure 2.2: Completed artifact in craftsman mini-game

The scribe mini-game follows the craftsman game. The goal is for the player to learn about the importance of hieroglyphics. This mini-game resembles “Simon says”. It is a memory game where the player needs to remember the correct order of the hieroglyphs presented. The pattern the player must follow is highlighted in green, and as he mimics the pattern, the hieroglyphs he chooses are highlighted in blue. Every time the player successfully completes a pattern, an educational audio sample about scribes and the hieroglyphs is played. An additional hieroglyph is added to the pattern every time the player wins a round. This continues until the player has memorized the pattern for an entire board, at which point the game restarts, or moves onto the next level if the player’s experience bar is filled.

The final game is the priest mini-game, another memory based mini-game. The goal is to teach students about Egyptian gods. In the game there are five objects (positioned at the top of the screen) which are

hidden by clouds [Figure 3.1], shown for a couple of seconds [Figure 3.2], and then hidden again. Each object is a symbol that is associated with an Egyptian god. At the bottom of the screen is the Book of the Dead, which displays a god on the right and the symbol associated with that god on the left. The goal is to remember where that symbol is hidden and tap it. Once all the symbols are found the game moves to the next god. If the player is in story mode and fills his experience bar, he becomes pharaoh and wins the game.

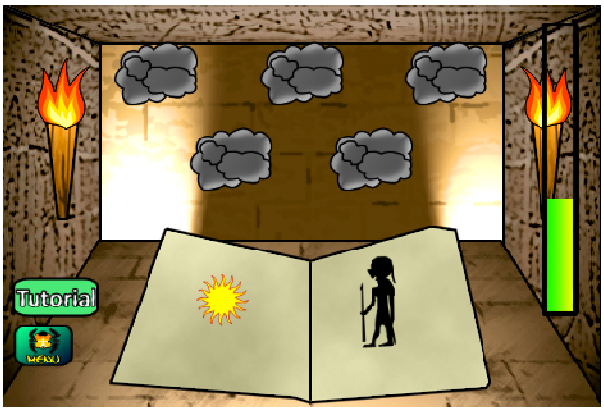

Figure 3.1: Symbols hidden in priest mini-game

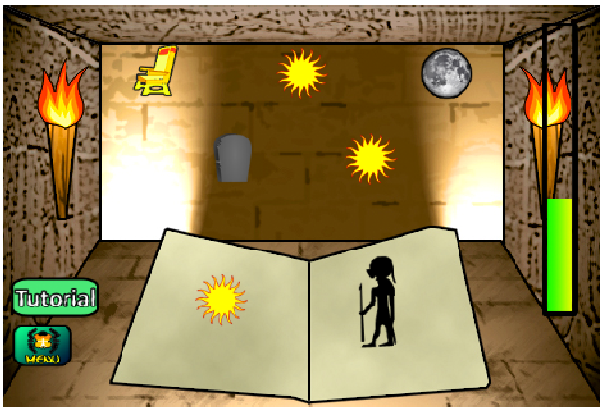

Figure 3.2: Symbols shown briefly in priest mini-game

3. Technical Overview of Applications

The following section delves deeper into the technical components of each application. It addresses the development process, design changes, and solutions to bugs discovered during testing.

3.1 Space Spell

Because *Space Spell* was my first attempt at creating an iOS application, the development process served as an introduction to Objective-C. The first half of the development process was spent learning the language and its connection to the hardware through Interface Builder. Interface Builder is the GUI that allows programmers to add objects (such as buttons) to the application and connect those objects to the code [Figure 4]. The second half of the development process was devoted to programming the logic and resolving issues that cropped up during testing.

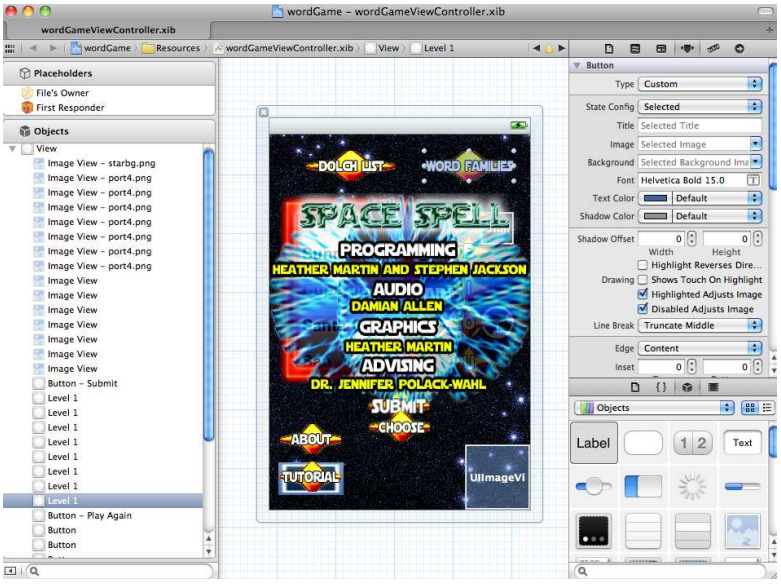

Figure 4: Using Interface Builder to add objects to the screen

The first issue encountered was brought to light after creating a feature for the GUI that allowed a rocket to automatically snap into a docking station. In the early stages of testing, it was discovered that a player could drag a rocket into an already occupied station resulting in the previous rocket being hidden from the player. The issue was resolved by creating a checking algorithm. Whenever the player dropped a rocket into a docking station, the algorithm checked for a rocket already existing in that docking station. If a rocket was found, the algorithm returned that rocket to its default position and allowed for the new rocket to be placed in the docking station. If no rocket was found in the docking station, the algorithm allowed for the placement of the new rocket.

As mentioned earlier in this report, towards the end of the development process, a request was made for the application to include additional words taken from "word families." This request served as the catalyst for changing the list selection from buttons to a scroll window. Buttons were a viable choice to display 4 lists but became an impractical option once the GUI was required to display 76 lists. Switching from buttons to a scroll window was a simple task because the functionality for scroll windows is built into xcode. However, the addition of an array specifying the titles of each list was required.

The next hurdle was adding audio to the game. Adding sound was the most time consuming process aside from programming the game logic. Each of the 1,000 words required a voice over clip. The recordings were created using Audacity, an audio editing software tool, and were voiced by Damian Allen. All the words were recorded into one file and then spliced into separate audio clips. The resulting clips were then remastered to normalize the volume level. They were exported using the .caf extension, an audio file type the iOS recognizes, and were named after the word they represented. This simplified the process of playing an audio clip because the name of the clip and the current word were identical.

Once sound was successfully implemented, an issue arose during testing. Whenever the player correctly spelled a word, a sound bite played to give them positive reinforcement before the game moved onto

the next word. To add variety, the sound bite that played was randomly chosen from three possible options; “good job”, “objective complete”, and “Houston we have a speller”. The delay between the sound bite playing and the game moving onto the next word was set to allow the longest clip to play uninterrupted. Unfortunately, this created an awkward delay after playing one of the shorter clips before the game moved onto the next word. The solution was to assign the length of the sound bite that was randomly chosen to a variable, which would determine the delay time before the game moved onto the next word [Figure 5].

```
}else{
    freeze = TRUE;
    NSString *pr;
    float wait;
    int randomNum = arc4random() % 3;
    if(randomNum == 0){ pr = @"houston"; wait = 2.5;}
    else if(randomNum == 1){ pr = @"objectivecomplete"; wait = 1.7;}
    else{ pr = @"goodjob"; wait = 1.5;}
    [self performSelector:@selector(playSound:) withObject:pr];
    [self performSelector:@selector(unfreeze:) withObject:nil afterDelay:wait];
    [self performSelector:@selector(loadNewWord) withObject:nil afterDelay:wait];
    [self performSelector:@selector(gameEnded) withObject:nil afterDelay:wait];
} //end else
```

Figure 5: Code snippet showing the solution to the wait delay problem

3.2 Rise of Pharaoh

When development on *Rise of Pharaoh* began I already had a grasp on the basics of Objective-C. Not needing to allocate time towards learning the language enabled me to attempt a more ambitious project. In terms of code and artwork, the scale of this project was much larger than what I had previously attempted with *Space Spell*. The first half of the development process was devoted to programming the game logic while the second half was spent creating artwork, sound, and debugging.

One new aspect I needed to learn for this project was view switching. Since there are four mini-games in total, trying to implement the application using only one view would have been cluttered and inefficient. A problem I ran into while switching views resulted from poor memory management; whenever the application received a memory level warning, it would deallocate one of its views. This turned into an issue when the program tried to switch to a view that it had deallocated; that view no longer existed, so the application would crash. The issue was resolved by changing the way views were switched. Before, there was a global instance of every view in the AppDelegate; a singleton class created by default for every application. In total there are eight views; one for the menu screen, arcade mode screen, level up screen, tutorial video screen, pyramid builder mini-game, craftsman mini-game, scribe mini-game, and priest mini-game. My solution was to use modal view controllers [Figure 6]. Every view in the application, from the mini-games to the menu screens, has a view controller associated with it. Each view controller has the capacity to present one other view controller, known as a modal view controller. Since each modal view controller removes itself from memory every time it is dismissed and instantiates itself every time it is presented, the memory issue was resolved.

```
arcadeMode *viewController=[[arcadeMode alloc] initWithNibName:@"arcadeMode" bundle:nil];
viewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
[self presentModalViewController:viewController animated:YES];
```

Figure 6: Code snippet showing the use of modal view controllers

Another challenge I encountered was creating a game loop for the mini-games. *Space Spell* did not require a game loop because it only listened and reacted to user input. On the other hand, most of the mini-games in *Rise of Pharaoh* did require a loop. The implementation of the game loop in this application does not follow the conventional method of updating game mechanics and graphics per frame. Instead, my update method indirectly calls itself after a specified timed delay. This is done by the update method calling some game function A_1, which then calls some game function $A_{2;}$ until it reaches the last game function it needs to call, $A_{n,}$ which finally calls the update method once again [Figure 7]. The problem I encountered with this implementation was when gameplay needed to be paused. The solution was to add a Boolean which prevented the loop from proceeding.

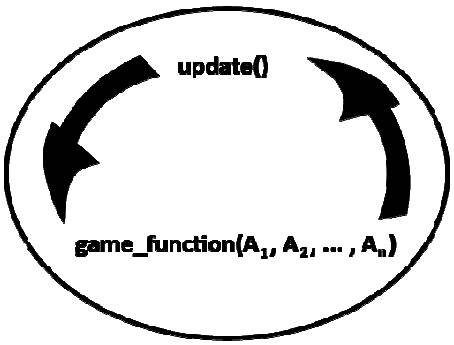

Figure 7: Diagram showing how the game loop works

After the code was finalized, artwork needed to be added. The issue with implementing multiple mini-games was that the amount of artwork required was increased to an unanticipated amount. The craftsman and priest mini-games required the most artwork. The craftsman game, apart from the background art, had 12 different artifacts in total. Each artifact needed to have 4 different images so that as the player got more of the pattern correct, the quality of the artifact improved. The priest game required artwork for each god and its corresponding symbol in addition to its own background art. Although completing the artwork for the priest and craftsman game took the most time, the other graphics for the mini-games, menu screens, and buttons were also time consuming.

The last feature implemented was audio. The scale of the audio in *Rise of Pharaoh* was much smaller than *Space Spell*. However, unlike *Space Spell*, the audio snippets required research; instead of voicing over words on a list, *Rise of Pharaoh* contains informative audio samples about various social and cultural aspects of ancient Egypt. This meant that although less time was spent recording the audio, more time was spent writing the script.

4. Data and Statistics

In order to test the hypothesis that students improve their understanding of a subject at a faster rate if they play educational games, experiments were run with the help of University of Mary Washington students. The goal of these experiments was to test knowledge of the subject material before and after

playing the application, in addition to gathering feedback pertaining to its usability. *Rise of Pharaoh* was selected for these experiments because of its greater complexity and level of difficulty.

The first test taken by the subjects was a pre-test consisting of 10 multiple-choice questions. This test was designed to determine their pre-existing knowledge about ancient Egypt. Having taken the pre-test, subjects read the "how to play" instructions and were then asked to play *Rise of Pharaoh*. The subjects were also given a paper copy of the instructions that could be referenced during testing. They were given 12 minutes to play the game, after which they were asked to take a post-test. The post-test was identical to the pre-test except for the ordering of the questions. Once the post-test was completed the subjects were then asked to take a usability test, which enabled them to rate different aspects of the application, offer constructive feedback, and report bugs.

The data gathered from the experiment was collected and inserted into an Excel spreadsheet where calculations were done to find the following: the average number of subjects who correctly answered each question in the pre/post-test, the score each subject received on the pre/post-test, the improvement percentage from the pre-test to the post-test, the average improvement of all subjects, and the average rating received for each question on the usability test. In total 36 students participated in this study.

The results showed that the average score received on the pre-test was 77% while the average score received on the post-test was 97%. This shows an average improvement rate of 20%. The following is a breakdown of how the subjects performed on each question:

- 97% were able to identify which continent Egypt was located on before playing the application and 100% were able to do so after.
- 100% of the subjects were able to identify the title given to the leaders of ancient Egypt before and after playing.
- 58% correctly identified the direction the Nile River flowed before playing and 91% were able to do so after.
- 58% were able to name the fertile black soil left by the Nile before playing and 100% were able to do so after.
- 38% were able to identify the amount of days the mummification process took before playing and 97% were able to do so after.
- 66% knew the name of the stone that helped the scientists understand the hieroglyphics before playing and 94% knew after.
- 94% were able to describe the climate of Egypt before playing and 100% were able to do so after.
- 80% knew the name of the ancient Egyptian god of the sun before playing and 86% knew after.
- 91% were able to name the embalmed body that was prepared for burial in ancient Egypt before playing and 94% were able to do so after.
- 80% were able to identify the largest desert in the world before playing and 97% were able to do so after.

A paired t-test was run using the aforementioned average scores per question; group 1 contained the averages for the pre-test and group 2 contained the averages for the post-test. The resulting two-tailed P-value was 0.01, which shows statistical significance in the data. This evidence supports the hypothesis that there is a higher rate of academic improvement in students who use educational applications.

The usability study results offered insight into how intuitive the application was. The test asked subjects to rate their comfort/agreement level from 1 to 5, with 5 being the most comfortable/agreeable. The following is a breakdown of the results:

- Average comfort level for operating iTouch devices was 4.4.
- Average agreement level that the game taught educationally appropriate material was 4.6.
- Average agreement level that the game was entertaining was 4.4.
- Average agreement level that the theme of the game was appealing to children was 4.4.
- Average agreement level that the game was visually appealing to children was 4.5.
- Average agreement level that the audio from the game enhanced learning was 4.5.
- Average agreement level that the images on the screen were easy to understand was 4.5.
- Average agreement level that the controls were easy to understand was 4.2.

The most common rating given for every question was 5, except for the last question where the mode was 4. This shows that *Rise of Pharaoh* was considered highly user friendly in the tests.

5. Conclusion

I have spent the past year creating educational games and testing the benefits of them. This culminated in the creation of two iOS applications, *Space Spell* and *Rise of Pharaoh*, which are both available to download for free on the App Store. Feedback from the usability test inspired ideas for possible updates in the future. The most requested features were the following: additional questions for the pyramid builder, more artifacts for the craftsman, more educational audio files for the scribe, and more gods for the priest mini-games. These could be addressed in a future update. The most common complaint was the volume of the audio. Complaints ranged from the audio being uncomfortably loud to inaudibly low. This stems from a difference in the devices rather than in the application itself, so there may be no quick fix for every user. Instead, the volume could be optimized for the newest generation of iPhones. Feedback from the usability test, although mostly positive, shows that *Rise of Pharaoh* has room for improvement.

The results from testing support the hypothesis that students who use educational applications improve their understanding of the subject material at a faster rate. However, the data would be more conclusive with a larger sample size. More testing is required to determine the full benefits of this project. These applications, along with those created by other students in the research group, show great potential as learning tools to aid teachers in the classroom.

Works Cited

"Global Mobile Statistics 2012." *mobiThinking*. 21 Apr. 2012. mobiThinking. February 2012 http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats#mobilebroadbandcountries.

Hu, Winnie. "Math That Moves: Schools Embrace the iPad." *The New York Times*. 21 Apr. 2012. The New York Times. 4 Jan. 2011 http://www.nytimes.com/2011/01/05/education/05tablets.html?_r=3&pagewanted=all.

Klopfer, Eric, Scot Osterweil, Jennifer Groff, and Jason Haas. "The Instructional Power of Digital Games and Social Networking Simulations, and How Teachers Can Leverage Them." *The Education Arcade*. 21 Apr. 2012. Massachusetts Institute of Technology. 2009 http://education.mit.edu/papers/GamesSimsSocNets_EdArcade.pdf.

Mallinson, Chris. "Apple's Touch Screens Can Bridge a Gap." *BC Hands & Voices*. 21 Apr. 2012. BC Hands & Voices. 23 Aug. 2010 http://www.bchandsandvoices.com/post/touch-screens.

"Teaching Children Reading Begins with Dolch Sight Words." *Teacher Support Force*. 21 Apr. 2012. Teacher Support Force. 2011 http://www.teacher-support-force.com/dolch-sight-words.html

Watson, William R., Christopher J. Mong, and Constance A. Harris. "A case study of the in-class use of a video game for teaching high school history." *Computers & Education* 56.2 (2011): 466-474. 21 Apr. 2012 [doi>10.1016/j.compedu.2010.09.007].

Zyda, Michael . "From Visual Simulation to Virtual Reality to Games." *Game Pipe*. 21 Apr. 2012. USC Information Sciences Institute. 2005 http://gamepipe.usc.edu/~zyda/pubs/zyda-ieee-computer-sept2005.pdf.