Saving Princess Sera
Game Design, Usability, and Interface

Hyun Lee Jordan
Department of Computer Science
Lynchburg College

Abstract

Game2Learn group at University of North Carolina at Charlotte (UNCC), whose ultimate goal is to create a programming-based MMORPG(Massive Multiplayer Online Role Playing Game) to assist in teaching intro-level computer science. The members of Game2Learn and its sponsor Dr. Tiffany Barnes have been working on an enormous project called, StormHaven. StormHaven is a game which is currently at a developing process. The purpose of this game will be to ease the steep learning curve normally associated with computer science and make computer science a more approachable discipline by using a more hands-on approach and using entertainment as a means of keeping students more inclined to focus (Game2Learn.com). The main purpose of the research for this summer was to develop a game to teach introductory level of Computer Science. We've determined on working in two different teams to create 2 diverse games to perform a usability test at the end of the summer.

Introduction

The principle of this paper is to present the works that have been performed this summer in depth. This paper will present details about the production of our game *Saving Princess Sera.* This paper will also talk about the different types of interfaces and the ways to learn programming.

Background

While there is a lot of speculation about the benefits of multimedia exploration, research on learning and technology suggests that the *creation* of media by students has even greater benefit for learning (Guzdial, 2001). This summer study was all about creating educational games to teach computer science and

study compatibility between the user and the interfaces. According to Dr. Myers' research, in today's applications, an average of 48% of the code is devoted to the user interface portion. The average time spent on the user interface portion is 45% during the design phase, 50% during the implementation phase, and 37% during the maintenance phase. While the programmers concentrate on designing interfaces, there are multiple different programs/tools that were developed to enhance human-computer interaction. Below are varieties of tools that are currently available to us.

Alice
Developed at Carnegie Mellon University, Alice is a programming environment where objects are manipulated in 3D worlds. Alice gives students the opportunity to learn about object-oriented programming concepts without the syntax frustrations imposed by text-based programming languages. With Alice, a programmer, in a GUI environment, selects program constructs and method invocations from lists of available choices (Gordon, 2006).

Drag and Drop Interfaces
Karel Universe is a drag and drop editor integrated with the Karel J Robot [1] simulator system. It is intended for those students who wish to learn Java with the absolute minimum of syntax. The editor permits the student to create classes, objects, and programs by dragging syntactically correct program fragments from one pane to another. The resulting programs may be then executed in the Karel J Robot simulator (Bergin, 2006).

DrJava
DrJava is a pedagogic programming environment for Java that enables students to focus on designing programs, rather than learning how to use the environment. The environment provides a simple interface based on a "read-eval-print loop" that enables a programmer to develop, test, and debug Java programs in an interactive, incremental fashion (Allen, 2002). DrJava is a friendly, highly interactive IDE targeted at teaching Java to beginners. DrJava has a simple interface consisting of a Definitions pane for entering and editing program text and an Interactions pane for evaluating arbitrary Java statements and expressions given the program in the Definitions pane. This interface frees students from the complication of defining main methods for their programs and encourages them to explore the Java language by conducting simple experiments (Reis, 2003).

Eclipse
Eclipse is a powerful integrated development environment (IDE) for Java targeted at professional software developers. However, Eclipse is poorly suited for use in introductory computing education because the complexity of its interface and the associated computing environment can overwhelm beginners (Reis, 2003).

Kid's Programming Language
KPL's language is modeled on the simplicity and readability of BASIC, but it is a structured rather than linear programming language. KPL lets kids see eye-catching and immediate results from their programs, while teaching them fundamental concepts like variables, data types, loops, decision structures, methods and functions. KPL's data types include integers, decimals, strings, booleans, arrays, and user-defined structures (Microsoft). The program is kid-usable but functionally complete integrated development environment. It also contains a growing collection of fun sample programs and games as well as an experience designed to make it fun for kids learning to code (Microsoft).

Unified Modeling Language (UML)
The Unified Modelling Language (UML) is a diagrammatic notation widely used in the computing industry and often taught in universities as a way to represent software requirements specifications and design descriptions (Flint, 2004). However, its features are not independent which is the source of numerous inconsistencies. Present consistency checking techniques are limited either to certain UML features or to certain kinds of inconsistencies.

Visual Programming
Visual programming language, VPL, is a method of creating codes by using visual representations such as icons, drawings, graphics, and even animations. Peridot uses visual programming, programming by example, constraints, and plausible inferencing to allow nonprogrammers to create menus, buttons, scroll bars, and many other interaction techniques easily and quickly. Peridot created its own interface and can create almost all of the interaction techniques in the Macintosh Toolbox (Myers, 1990). Due to the fact that this tool is so easy to use, even non-programmers can create professional-like programs.

jGRASP
jGRASP is an integrated development environment that provides automatic generation of visualizations to improve the comprehensibility of software. These visualizations, which are particularly well suited for CS1 and CS2 students learning Java, include Control Structure Diagrams, UML Class Diagrams, and dynamic Object Views (including arrays ArrayList, LinkedList, HashMap, and TreeMap) (Cross, 2006).

In *Saving Princess Sera,* the interfaces that we've used are somewhat similar to the methods that I've mentioned above. Since this literature review was done after the completion of the study, I was not aware of all the different ways to design interface. Surprisingly, the interfaces that we've created are found somewhat effective according to above techniques.

Methods

The study for this summer was consisted of developing two different games created by two different game engines. *Saving Princess Sera* was created by

using Role Playing Game Maker (RPG Maker), which gives you the power to create your own original role-playing games. Its highly user-friendly editor interface satisfies beginners and experts alike in developing and playing games created by it. The other game *Catacombs,* is developed by using Never Winter Nights (NWN), produced by BioWare and published by Infogrames (now Atari), is a third-person perspective computer role-playing game that is based on third edition *Dungeons & Dragons* and *Forgotten Realms* rules (Wikipedia).

There were two main developing stages in the creation of *Saving Princess Sera*:

Brainstorming

Before we have decided on which game engines we wanted to use, we had to think of the quest ideas of the game. All the team members have gathered together and thought of the way to implement the principles of computer science into enjoyable game ideas. At first, we had to decide on what programming concepts to use. Our group was planning to use if/then statements, nested for loops, while loops, and switch statements. Since the game is going to supplement the introductory leveled Computer Science courses, we thought these concepts would be appropriate. Once we have settled on the concepts, we then had to decide on what kind of interfaces we are going to use to implement these concepts. After a long thoughtful brainstorming we thought fill in the blank, multiple choices, flow charts, dialog trees, and Russian Keyboard were most appropriate for the game. Once all the other decisions were made, we came up with the main concept of the game. The original concept of the game was Village. Village was a game idea that I had in mind. It was something that I wanted to work on the side while learning RPG Maker. Since we've decided to use *Village* as the concept of our game, we proceeded. Soon after working on our game, *Village,* we believed that we need something more than just the player walking around the village and talking to people. Eve, one of our teammate, has decided that by feminizing the game a little, we should make the major theme of the game to a Warrior Princess, Sera. Sera is a princess, captured by a monster Gargamel on the day of her 16th birthday. Arshes, a man from the village hears this news and tries to go rescue the princess. In order to rescue the princess, Arshes must go to castle. However, because of the fact that Arshes is poor, he must complete a multiple of quests to make money so he can buy a ticket to get on the boat.

Execution

At first, we had to create a basic layout of the game. I was in charge of building the layouts. Due to the fact that RPG Maker has such a simple way to develop interfaces, it was no time before the basic layout of the game was created. However, although RPG Maker game engine provided most of the functions that were needed for the game, due to the fact origin of this game engine was in Japan, we had some difficulties in scripting. Despite of all the difficulties, we've arranged to complete our game in success. Below are some screen shots of *Saving Princess Sera.*

Figure 1. Start Screen

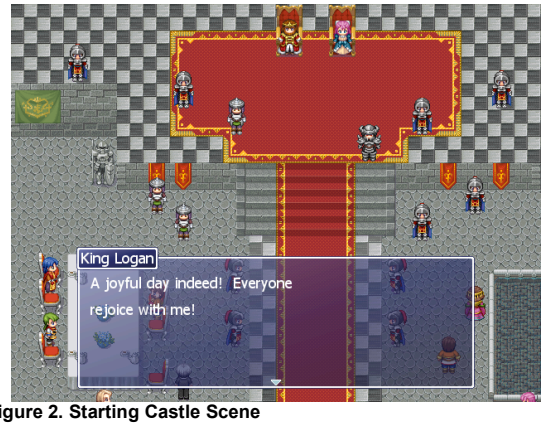

Figure 2. Starting Castle Scene

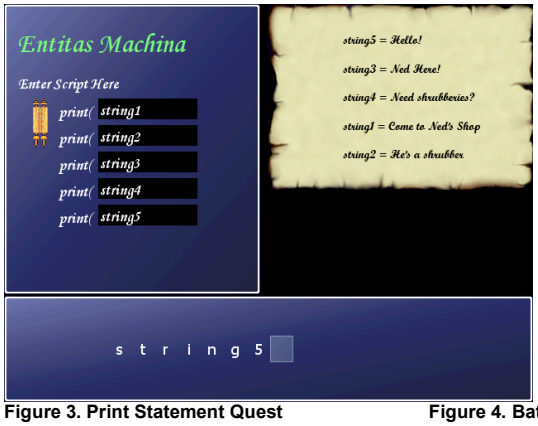

Figure 3. Print Statement Quest

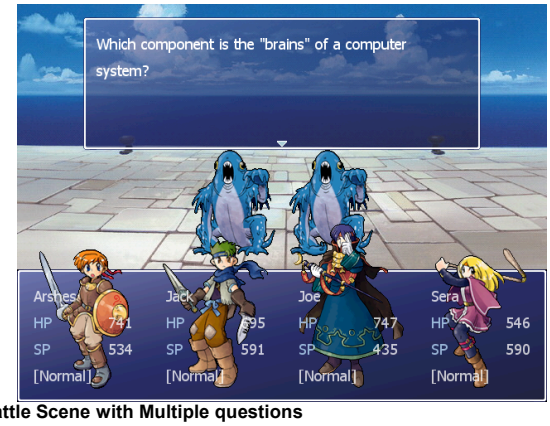

Figure 4. Battle Scene with Multiple questions

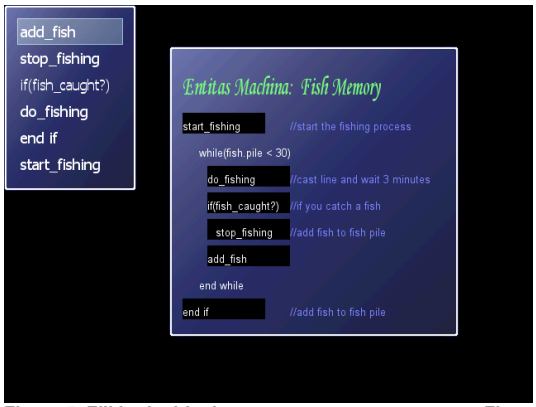

Figure 5. Fill in the blank

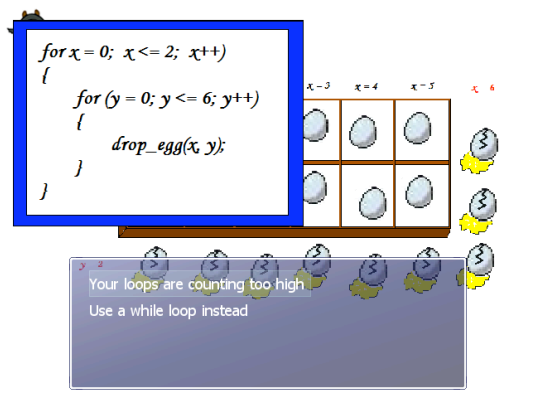

Figure 6. Egg Quest with Dialog Tree

Figure 7. Rescued Princess

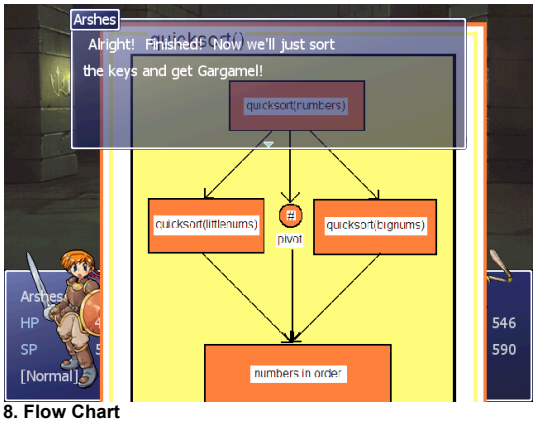

Figure 8. Flow Chart

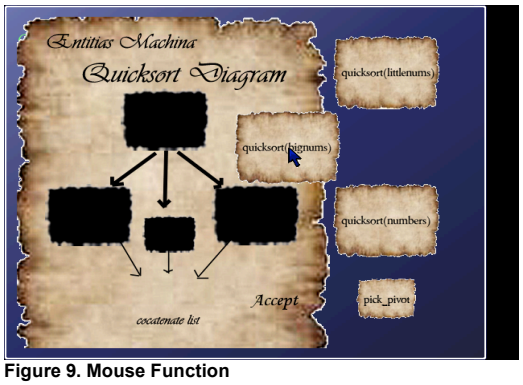

Figure 9. Mouse Function

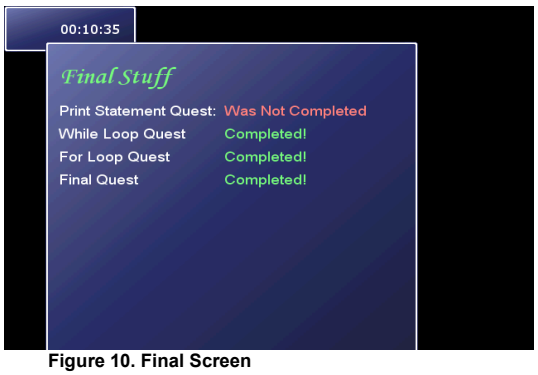

Figure 10. Final Screen

Due to its friendly and colorful interface, *Saving Princess Sera* looked fantastic and appealing. As you see in Figures 1-10, we've arranged to include all of our quests and concepts. As I have mentioned before, RPG Maker included most of functions and at this moment, I would like to talk about what other functions we have manage to include other than the basics of the engine. As you see in Figure 3, in print statement, we've included typing functions. The user uses the keyboard and the Tab control to input the answers and correct them by using the arrow keys. In Figure 9, shows the mouse control. Since the study was about the usability test, having a mouse control gives a variety in input controls. Figure 10 shows what quests the player has completed. Since the player is not aware of how many quests are in the game, this function lets the player know what he/she has accomplished and what quests were ignored.

Usability Test & Data Analysis

After the completion of *Saving Princess Sera* and *Catacomb*, the usability test initiated. There were going to be twenty subjects participating in our study and each subject plays both of the games. This usability test is consisted of a demographic survey, two written tests, playing *Saving Princess Sera, Catacomb,* and the final interview. Two written tests are made up of multiple choice questions and fill in the blanks. The composition of the post-test is very similar to the pre-test with slight changes. The pre-test given to determine the knowledge of the participant before the time of play and the post-test is for afterward. Both

of these tests contain basic knowledge of computer science, therefore, each tests can be completed in less than 10 minutes. Before each subjects started to play the game, they were giving a sheet of paper, which explains basic key functions for the game. They were to read the sheet and to have a slight understanding of what kinds of basic key functions are used in each game. While the participants are playing the game, each participant's monitor was recorded. During the interview sessions, we also recorded the answers to analyze their comments about the games.

Data analysis was the most time consuming part of the study.
Thus far we've had 13 participants and the data that are provided in this paper is based on the analysis of the 12 participants. Once our data from the demographic survey was analyzed, we've noticed that most of our participants were white males aged between 18-24. Once the pre-tests and the post-tests were graded we've noticed a sight improvement in our results. On the pre-test, the participants averaged 4.9 correct answers, and the post-test averaged 5.2 correct answers. Although both of these tests weren't consisted of complex questions, the participants seemed to have some difficulties. Despite of these appalling test results, some of our participants had 100% correct answers. During the interview, the participants told us that both of these games were excellent and that we should continue to work on developing these kinds of educational games. However, many of them expressed that these games were great as class supplements. Although both of the games illustrated great programming concepts, for a beginner, he/she will have difficult time learning from it. Some of the other participants also mentioned that by using these games as a test review or homework.

Although most of our participants had some familiarity in playing games, despite of our intentions, some participants also had difficult time playing the games as well. Some said that there wasn't enough instructions on what they were suppose to do during the game. In the game of *Saving Princess Sera,* the "Go Back" function is not available and once the player gets an answer wrong, often times the player had to go through the quest one more time in order to complete it. Overall, the participants were very satisfied with our work. They gave us some great feedback on the positives and the negatives.

Conclusion

This summer study has been such a wonderful journey. Because of the testing our products, we learned about the outstanding aspects of our game and what needs to be improved. We've also learned that these types of educational games can definitely be beneficial to standard classroom environments. In conclusion, students are most likely to enjoy doing their assignments through these types of non-traditional learning tools and they can actually learn from it.

Future Work

StormHaven is a project that can certainly break the traditional cycle of learning. Technology is growing while the number of students enrolling in computer science decreasing. In order to solve this problem, we must continue to make computer science more appealing. Through creating a learning tool such as StormHaven, we can show people that computer science is not difficult or boring. Although my main interest in computer science is not developing games, I am planning to work on developing user-friendly interfaces in the future to make the interaction between human and computer more effective and pleasurable.

References

Allen, E., Cartwright, R., and Stoler, B. 2002. DrJava: a lightweight pedagogic environment for Java. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (Cincinnati, Kentucky, February 27 - March 03, 2002). SIGCSE '02. ACM Press, New York, NY, 137-141. DOI= http://doi.acm.org/10.1145/563340.563395

Bergin, J. 2006. Karel universe drag & drop editor. In *Proceedings of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education* (Bologna, Italy, June 26 - 28, 2006). ITICSE '06. ACM Press, New York, NY, 307-307. DOI= http://doi.acm.org/10.1145/1140124.1140212

Cross, J. H. 2006. jGRASP: an integrated development environment with visualizations for teaching Java in CS1, CS2, and beyond. *J. Comput. Small Coll.* 21, 4 (Apr. 2006), 118-118.

Flint, S., Gardner, H., and Boughton, C. 2004. Executable/Translatable UML in computing education. In *Proceedings of the Sixth Conference on Australasian Computing Education - Volume 30* (Dunedin, New Zealand). R. Lister and A. Young, Eds. ACM International Conference Proceeding Series, vol. 57. Australian Computer Society, Darlinghurst, Australia, 69-75.

Gordon, A. 2006. Tutorial on Alice. *J. Comput. Small Coll.* 21, 3 (Feb. 2006), 130-130.

Guzdial, M. 2001. Use of collaborative multimedia in computer science classes. In *Proceedings of the 6th Annual Conference on innovation and Technology in Computer Science Education* (Canterbury, United Kingdom). ITiCSE '01. ACM Press, New York, NY, 17-20. DOI= http://doi.acm.org/10.1145/377435.377452

Microsoft. Kid's Programming Language. August 20, 2006. http://msdn.microsoft.com/coding4fun/coolapplications/kpl/default.aspx#top

Malgouyres, H. and Motet, G. 2006. A UML model consistency verification approach based on meta-modeling formalization. In *Proceedings of the 2006 ACM Symposium on Applied Computing* (Dijon, France, April 23 - 27, 2006). SAC '06. ACM Press, New York, NY, 1804-1809. DOI= http://doi.acm.org/10.1145/1141277.1141703

Myers, B. A. 1990. Creating user interfaces using programming by example, visual programming, and constraints. *ACM Trans. Program. Lang. Syst.* 12, 2 (Apr. 1990), 143-177. DOI= http://doi.acm.org/10.1145/78942.78943

Myers, B. A. and Rosson, M. B. 1992. Survey on user interface programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Monterey, California, United States, May 03 - 07, 1992). P. Bauersfeld, J. Bennett, and G. Lynch, Eds. CHI '92. ACM Press, New York, NY, 195-202. DOI= http://doi.acm.org/10.1145/142750.142789

Reis, C. and Cartwright, R. 2003. A friendly face for Eclipse. In *Proceedings of the 2003 OOPSLA Workshop on Eclipse Technology Exchange* (Anaheim, California, October 27 - 27, 2003). eclipse '03. ACM Press, New York, NY, 25-29. DOI= http://doi.acm.org/10.1145/965660.965666

Wikipedia: The Free Encyclopedia. "Never Winter Nights". No pages. Cited 25 August 2006. Online: http://en.wikipedia.org/wiki/Neverwinter_Nights