University of the Philippines Los Baños
Institute of Computer Science

Less Hell with Shell: An Educational Game for Shell Scripting

Author: Earl Joseph A. Palapar
Adviser: Zenith O. Arnejo
2020

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

TABLE OF CONTENTS

ABSTRACT

This paper presents the design and implementation of “Less Hell with Shell”, a Java-based desktop game application that aims to teach the basics and fundamentals of the Bourne-Again Shell (bash) scripting to students and programming enthusiasts. The game application was composed of various stages including crossword puzzles, mazes, and coding exercises. In order to determine whether the game was able to help the users learn bash, a pretest and posttest approach was conducted. Reliability testing of the questionnaires yielded a Cronbach’s alpha value of 0.91, which implied an excellent consistency of the items. Results of the pretest and posttest showed a 57% increase in the respondents’ bash proficiency score. Finally, in order to evaluate the usability and performance of the game, an adapted Game Playing and Game Software Technicality questionnaire was used. Survey results showed an average mean agreement score of 74% for Game Playing and 80% for Game Software Technicality.

ACKNOWLEDGMENT

I express my gratitude to my SP adviser, Ma’am Zenith, for providing me the insight in creating the project, helping me in developing this game, and being the bridge for this project’s accomplishment. I would also like to thank the respondents for having been a part of this project and making this possible. My gratitude is also extended to my family, who continued to support and believe in me. Lastly, I would like to thank God for providing me the strength and determination to finish this project.

I. Introduction

A. Background of the Study

In this day and age, education has evolved to be much more convenient and accessible, and more so with the assistance of developed technology. In this technological age, the teaching and sharing of knowledge has become relatively dependent on the growing usage of educational games, or what is called “game-based learning” (Dadheech, 2018). In order to impart knowledge on learners, particularly students, who are most exposed to today’s digital age, the utilization of educational games is one desirable way. Games are advantageous in providing knowledge and teaching learners, over other means or mediums of education (Squire, 2013).

A topic in Computer Science where the concept of “game-based learning” can be applied is through the teaching of shell scripting, or the usage of scripting language. Shell scripting is a relatively vague concept among students who are taking programming courses. Scripting languages have its advantages, and one of which is, it operates at a high level, compared to compiling languages. Manipulation of files and directories are made easier through shell scripting. Writing lengthy codes in compiling languages is a lot less efficient compared to writing a script. The shell is universal across most systems, particularly UNIX, and so scripts can be run in multiple systems (Robbins, Beebe, 2005).

As useful as the concept and process of shell scripting is, some developers, programmers, or students are not actually knowledgeable about it, and do not know their way around the shell. Some might be able to read or perceive a written script, but do not entirely understand how the scripting language works. In order to assist in the teaching of shell scripting and scripting language, an application is developed, which is an educational game, thus, employing the approach of game-based learning. The usage of an educational game as a medium helped learners to better understand and grasp the handy and essential knowledge of shell scripting.

B. Statement of the Problem

Shell scripting, and the use of scripting language are concepts and topics that programmers and students who are taking computer programming degrees encounter. Learning these is one of the fundamental skills that a programmer should possess, as this is useful and convenient, most especially in the industry of computer programming. Shell scripting relies on the usage of the shell. The shell of the operating system is a program, which the user interacts with in order to accomplish tasks. Commands are provided as input, which then the powerful shell executes. One of the most compelling features that the shell provides is its support for the creation of scripts, which contains the commands that the user wants the shell to execute (Veeraraghavan, 1999). In the industry, having the considerable skill in mastering shell scripting assures a promising career progression. Demand for this talent grows continuously, and there are various career opportunities for developers and systems administrators that possess the skill in shell scripting (Sethi, 2019).

Shell scripting, although is relevant in the programming industry, faces a scarcity in professionals who possess the necessary experience in the market, despite the booming

demand for it (Sethi, 2019). Generally, the skill in mastering shell scripting is rather unpopular among others, in the field of programming. There are a number of queries that are present in online question-and-answer communities such as Quora and Stack Overflow that are related to learning shell scripting, where most users who posted the questions encounter a difficulty in it. Likewise, there have been efforts to help learners to be more familiar about the concept of the shell and shell scripting, like the unfinished application “Bash Academy”, which is an initiative that promotes the bash (Bourne-Again Shell) language and educates people in its use. With these, there is a need to overcome this problem, for the benefit of the knowledge of the people, through a method and medium where the youth especially is most accustomed with, which is, by educational games.

C. Significance of the Study

The process of shell scripting is a concept in Computer Science which is generally not tackled much, nor taught thoroughly to students, therefore, such students, as they become developers, are not entirely knowledgeable about shell scripting. Shell scripting is a helpful and convenient process wherein commands for a UNIX-based system are written to a text file, and upon execution of the file, the commands in it are also executed sequentially. The command line is a primary tool which a developer uses most of the time. There are times when a developer must have to work on several repetitive tasks, and to avoid doing these tasks manually and to save a significant amount of time and effort, shell scripting is a convenient and desirable option. File manipulation, program execution, and text printing are some of the most common uses of shell scripting.

To help students and developers in grasping the skill in shell scripting, one way is through a method which most people enjoy: a game. Like Terminus, a game developed by the Massachusetts Institute of Technology and aims to help its users to familiarize and understand basic Linux commands by having a visual representation of what each command does, a similar educational game which also helps people learn about shell scripting was the aim of this study (Conrad et al, n. d.). With this, as the target people engaged in the application, while they become entertained, they also learn in the process. The study focused on creating the game through the principles and methods of game development, in a standalone programming language, specifically the Java Programming Language. In conclusion, the study’s significance was to basically let its target users, which are the students and developers, to be much more knowledgeable about shell scripting.

D. Objectives of the Study

The general objective of the study was to create an educational game that teaches shell scripting, particularly bash (Bourne-Again Shell) scripting, to be able to help and encourage people, especially students and developers, to learn more about the concepts of shell scripting through a method that is both entertaining and educational.

Specifically, this study aimed to:

- design a storyboard which made use of a word-type and multiple choice-type games to introduce concepts and terms in shell scripting, and a coding game where users can experience shell scripting firsthand
- develop a desktop game application that teaches shell scripting
- evaluate the usability and performance of the game through the use of pretest and posttest approach, as well as a game usability testing questionnaire

E. Scope and Limitations of the Study

This study aimed to develop an application that helped and assisted in the education of shell scripting, particularly bash (Bourne-Again Shell) scripting, and is titled as “Less Hell with Shell”. The application was a desktop application, and thus, ran in a Linux-based, desktop operating system, which has support for the bash shell. There was no mobile version for the application that was created. The game covered only the topics from writing a shell script and making it work, the basics of the shell, variables, substitution, shell functions, flow control, keyboard input and arithmetic, proper shell scripting practices, positional parameters, to errors, signals, and traps. Lastly, the developed application was aimed towards students, particularly those who are taking computer programming courses.

II. Review of Related Literature

On Educational Games

In the recent years that have passed, the academic community has taken notice of educational games and their concept. There are various improvements in the education system, and most of these are commonly associated with educational games, particularly with their capacity to motivate and gain interest from learners. However, even with these, the process of completely integrating video games in the system of learning is still a rather vague one, and the potential of games to make an impact diminishes. Research is being done to analyze the hindrances, to better integrate games in the learning system, and how to undertake these hindrances. The product of this is the adventure platform, which is a tool which could help instructors use video games as another learning tool, and use video games to the fullest of their advantage. In the integration of games, the adventure platform has three primary points, which are the reduction of the high development costs of educational games, involvement of instructors in the development process to enhance the educational value, and the production of the games using a white-box model (Torrente, del Blanco, Marchioni, Moreno-Ger, Fernández-Manjón, 2010).

Abstract computer games are currently enjoying a huge popularity among the youth and adolescents. With this being said, software designers and developers have the opportunity to exploit this and use this to its fullest advantage, to make software being created more motivational and enticing to their target market. In spite of these, the educational scope or the capacities of educational games to contribute in education is still to be explored. In the paper by Virvou, et al. (2005), it explored certain elements of educational games, particularly the educational effectiveness, and the appeal and scope of educational software games with the use of an evaluation study. The results of this evaluation explains and describes that the

educational virtual reality games truly have the ability to motivate and spark the interests of students, and in the process, even enhancing the educational effects on these learners. One of the most significant outputs is that the educational effectiveness of a certain game on a particular set of students who possessed inferior results in performance in a certain domain, before their exposure on the learning environment which employed the use of games, yielded high and desirable outcomes (Virvou, et al 2005).

On Shell Scripting

A shell script is basically a text file, which contains successive multiple and various commands. The system then performs the given commands accordingly upon the execution of the file, which is called the script. “Shell” is a specific term which indicates the command-line user interface or environment which, in turn, is connected and interacts with the kernel. In the Linux Operating System, there is a diversity of different shells, which are particularly the C shell (csh), the Korn shell (ksh), the Bourne shell (sh), and the Bourne-Again shell (bash). The shell, which is the command line interface, refers to the provided commands as input, whether it comes from the terminal or a script, and upon reading a command, translates and interprets it for execution, and performs the same on the succeeding commands. The Bourne-Again shell is the shell that includes the features that other shells possess (Saternos, 2005).

On Game Design/Development

A game consists of two terms or concepts, which is called the “Game Play” and “Game Data.” What the game play is constituted of are the actions that the user or player executes while playing the game. Different genres of games have different sets of actions, with one having its own set, but typically, most games share some conventional actions, which an example is, specifically, moving objects on the screen. In the real world sport of ping-pong, adapted in a classic arcade game, the player of the game makes use of the action of a moving object, by moving a paddle in different directions on the screen, to intercept a moving ball. Games can be classified as simple or complex, but regardless of the classification, the game designer is always the one responsible for making the decision of how the game play will occur. Designers interact and work with the programmers to determine how the events and happenings in the game shall take place, such as the speed of a moving ball on the screen, the sensitivity of a paddle according to a player’s reflexes, and its force upon contact with a moving object. Randomization and the element of chance is commonly utilized in simple games, to reduce the predictability of a game, through manipulating probabilities of events (Moore, 2011).

On Java Game Programming

Java is a relatively popular programming language where the common programmer would probably be discerning of its features and advantages, such as an object-oriented paradigm, cross-platform support, code reuse, ease of development, tool availability, reliability and stability, good documentation, support from Sun Microsystems, low development costs, the ability to use legacy code (e.g., C, C++), and increased programmer productivity. The Java programming language is a suitable programming language for game programming, despite the common and typical notions or misconceptions regarding it, which usually are the opinion of

people who firmly believe that games must be programmed and implemented with languages such as C, C++, assembler, etc. Such misconceptions are: it is too slow for programming, it has memory leaks, it is too high-level, it has a bad application installation, it has no support for game consoles, no one actually uses it to write games, and Sun Microsystems does not support Java gaming. These statements, however, are not true as the facts prove that they are substantially wrong. For one, Java is roughly the same speed as C++. Memory leaks are avoidable through better programming and profiling. It is high-level, but with the reward of direct access to hardware and external devices. Proper installation software ensures smooth installation. Lastly, more and more exceptional games are being developed with Java, with the support from Sun Microsystems (Davison, 2005).

The same advantages are noted by Davidson in 2003, in his paper. Programming with Java has its recognized advantages, such as object-orientation, cross-platform support, code reuse, ease of development, the availability of tools, reliability and stability, good documentation, continuing support from Sun Microsystems, low development costs, the ability to use legacy code, (e.g. C, C++), and increased programmer productivity. A study from 1998, which is a relatively old one, had indicated that software written in Java has a 25% time or cost saving, and a 30% increase in overall productivity, as compared to software written in C++. It can be noted that these figures are taken from an old research, and that these could have much more increase in this modern day as the features of JS2E have improved much, with larger libraries and better tool speed. There are multiple ways and approaches on how to engage in games while utilizing Java, namely, applets, pure Java applications, dirty java applications, and Java as a scripting engine. In the aspect of commercial games, while it can be noted that the popularity of Java's use in games is nowhere near to that of C, or C++, people have come to utilize it for games with its features, and still, it is used in the gaming industry. Commercial games programmed in Java are mostly built using the approach of dirty Java programming, which means it is a hybrid of 2 programming languages, Java and commonly C++. Some examples of these games are Tom Clancy's Politika (1997), Roboforge (2001), IL-2 Sturmovik (2001), Who wants to be a Millionaire (2000), You don't know Jack (1995), Majestic (2001), Vampire - the Masquerade: Redemption (2000), Star Wars Galaxies, Runescape, Skies of Arcadia (2000), and Daytona USA (2001). These games are a counter to the common misconception that no one writes serious games in Java (Davison, 2003).

III. Materials and Methods

A. Development System

The application was developed using the Java Programming Language, specifically Java OpenJDK 11 for Linux 64-bit, on a computer which had Ubuntu 18.04 64-bit as its operating system. The total specifications for the system were: Intel Core i5-3337U, Nvidia GeForce GT 740M, 8 GB RAM, and a 500 GB hard drive. The application was tested on the current system, and on a virtual machine running Ubuntu 14.04 32-bit. The virtual machine was run using the Oracle VM VirtualBox application.

B. Game Assets

The assets and sprites that were used for the application were images taken from free clip art libraries online, or original work created using the applications Adobe Photoshop CS6 and GUI Image Manipulation Program (GIMP) 2.10. The background music used as an asset, which is titled “Night in Venice” by Kevin Macleod, was obtained from incompetech.com, which hosts free music to be used, and was also explicitly cited in the credits section of the application.

C. Shell Scripting Concepts

The shell scripting concepts that were incorporated in the application were specifically about bash scripting. The game contained ten topics, which taught its users all about the fundamentals of the bash shell and bash scripting, which then helped them to be much more honed and familiarized when encountering other forms or types of shell scripting later on. These concepts or topics were, namely, writing a shell script, shell basics, variables, substitution, shell functions, flow control, keyboard input and arithmetic, proper shell scripting practices, positional parameters, and errors, signals, and traps. These fundamental concepts should be able to give the user a brief and essential background and knowledge in bash scripting.

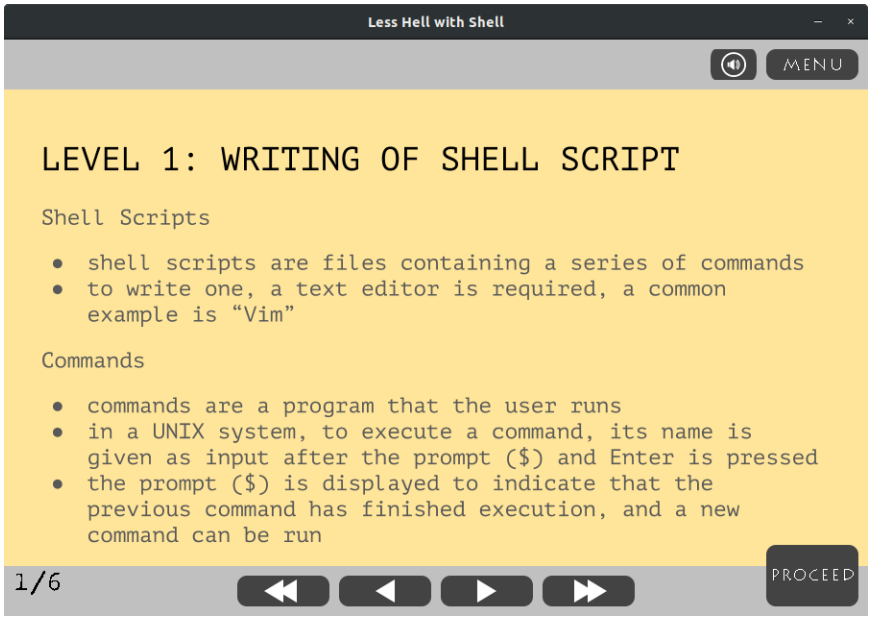
D. Game Details

A storyboard was created during the pre-development stage (see Appendix I). The storyboard contained the complete description of the entire flow of the game, where it had ten levels which correspond to the bash scripting topics, and to unlock a level, the level prior to that must be accomplished first. Only the first level is playable in the first execution of the game. In the flow of a certain level, the user will go through a series of slides first, which will provide an overview of the topic. Up next is the crossword puzzle stage of the game, where the user must complete all the required words to be formed in order to proceed to the maze stage, in which the user must garner a score of 55% or higher in under two minutes. In some levels, after completing the maze stage, the user must complete a coding exercise, to finally accomplish the game, and move on to the succeeding level.

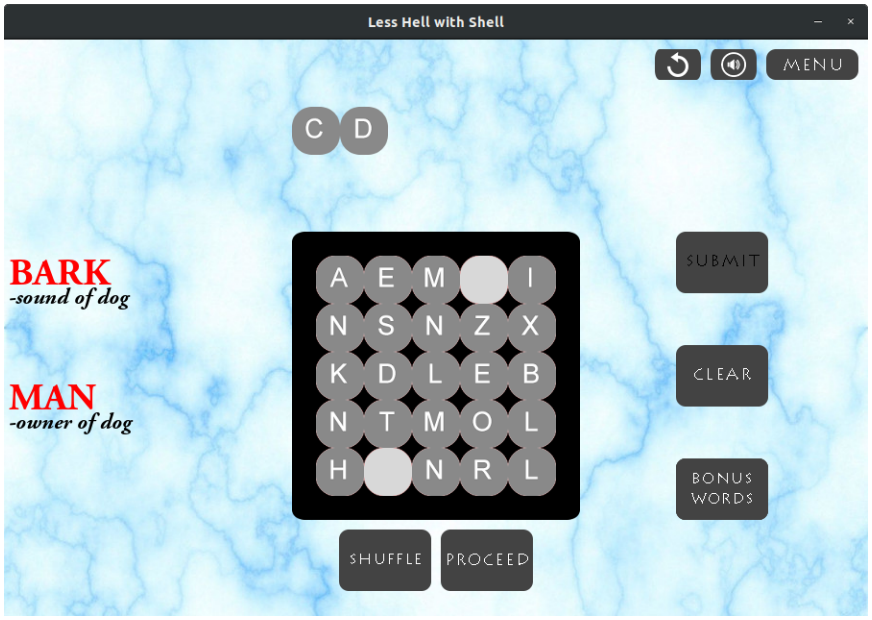
There are three stages in the game: crossword-type, maze-type, and a coding-style type. The main menu section of the game contains the credits and tutorial, which the user can click for an overview on how to play the game.

In the level selection screen, at first run, only the first level is unlocked. To unlock a level, the level prior to that must be accomplished first. Each shell scripting concept that will be taught in the game is at a level of its own. The level buttons which are highlighted as orange are the levels which contain coding exercises.

Before the player can proceed to the crossword stage of the game, the game displays some teaching slides first, which discusses the certain concept of shell scripting. These concepts and principles will be what the succeeding game stages are all about.



After going through the teaching slides, the player will now be redirected to the crossword-type game. To be able to finish the stage, the user must first form all the required words. Bonus words can be formed, which can be used as hint points in the later maze-type stage of the game. Linux commands are the words that can be accepted as bonus words.



Upon completion of the crossword stage, the player will now be redirected to the maze stage of the game. Here, the user must be able to complete the stage by scoring 55% or higher correct answers of all the questions answered, and finishing in under 2 minutes. The entire level is forfeited if the user does not meet either of these conditions. If the user has accumulated hints

during the earlier crossword stage of the level, two types of hints can be used, which are the torch, which lights up the entire map of the maze, and the hint, which lets the user take a “peek” at the teaching slides. Both of these consume a hint point.

Some levels, particularly levels 3, 5, 6, and 9 have coding exercises after the maze stage. After completion of the maze stage, the user must solve the question and come up with the correct code in order to ultimately finish the entire level.

After completing the level, the application will show a notification screen that the level is now accomplished. Upon going to the level selection screen, the next level after the completed level is now unlocked. The user can now proceed to play the further levels.

E. Game Development in Java

Java is one of the most common programming languages that is used in the programming industry, and one of the features it is most known for is its support for object-oriented programming. The common misconceptions debunked and disproved about Java explains that it can be used as a good programming language, example, for games (Davison, 2003).

1) Creating Game Objects

The creation of game objects in Java revolved mainly on treating everything in the application as an object, and creating a class for it, thus, employing the approach of Java's object-oriented programming. For example, the maze stage of the game had its own class "Maze", which contained a class "Room", so on and so forth. Assets such as the background music and sprites are obtained from free clip art libraries, or created using photo editor applications Adobe Photoshop CS6 and GNU Image Manipulation Program (GIMP) 2.10.

2) Programming in Java

The application is created in the Java Programming Language, specifically OpenJDK 11, through the Eclipse IDE, which is known for its integration with Java and its functionalities. There were no external libraries used for the application, and no game engines used for its creation. Java SWING and AWT components were used for creating the GUI of the game. In the coding parts of the game, the application connected to the bash shell for checking and executing the bash scripts submitted by the user. Java has support for executing scripts and

connecting to the shell of the operating system, hence, the application ran on a Linux operating system, which has a bash shell.

3) Application Distribution

The application was created in the Eclipse IDE, which has features made for Java development, hence, the packaging of the application is made easier. The application was exported to a JAR file, then bundled with a JDK binary, so running in systems did not require a Java installation. A 64-bit and 32-bit JDK binary was included for more compatibility across systems. A shell script which executed the application was also included, where the user can open the terminal and can run this script to start the game. All in all, these three files were compressed to a ZIP archive, and ready for distribution.

F. Usability Testing

Descriptive sentences or criteria were devised which were used for the usability testing of the developed desktop application. This system was adapted from the survey used by Delos-Reyes. Two criterias were present on the survey to subdivide the descriptive sentences that were used to measure and describe the usability of the application. These criterias were “Game Playing”, which contained 5 descriptive sentences, and “Game Software Technicality”, which contained 6 descriptive sentences. Respondents were ones who were taking or had taken a degree or courses in programming, or those who were inclined to programming, given that they had machines which ran a Linux operating system. These respondents were given a copy of the ZIP file which contained the application to be used and tested, through a shared folder in Google Drive.

After the users had finished using and playing the application, they were then given a link to the Google Forms survey which was the questionnaire for the usability testing. This questionnaire had a 7-point linear scale for every descriptive sentence, ranging from 1 (Disagree) to 7 (Agree). This survey also measured the level of satisfaction that the user had experienced in playing the game.

Likewise, comments and suggestions were also given by the respondents as input, which would be able to help in the overall betterment of the application.

IV. Results and Discussion

Using OpenJDK 11, and the Eclipse application, a fully working and functional game was created, that was capable of teaching and educating its users about shell scripting, specifically bash scripting.

The overall usability of the application depended on, first and foremost, the operating system that the user’s machine ran on, since the application can only be run on a Linux operating system, because of its requirement of the bash shell environment, which Linux systems possessed. The version of the Linux operating system of the respondent did not necessarily matter, as long as the operating system contained the bash environment, which was standard on Linux, and there were 32-bit and 64-bit versions of the application. Profiling was done to the application, using the application JProfiler, where the executable .jar file was tested,

to measure how much resources the application required in running. A screenshot of the telemetry retrieved from the profiling by JProfile is present below.

It can be inferred in the image that the application used the most resources during its initialization phase, where it mostly instantiated the Java classes and generated the assets by reading the files from the resource directory. It can be seen that the resource consumption of the application observed a spike in the first few seconds of execution, where its memory usage increased from 80 MB to 200 MB, and the load on the CPU was at its peak.

Figure 1. Telemetry of Profiling by JProfile

A 17-item pretest and posttest questionnaire was used to determine how the application had contributed to the respondents in their knowledge of shell scripting. The pretest and posttest questionnaires had identical questions, but different order of questions. The items in the posttest questionnaire were shuffled in order to test if there was an improvement from the respondent after playing the game. To determine the validity and reliability of the questions, two statisticians were asked to analyze the results from the questionnaires and to determine the Cronbach's alpha, which was the level of measurement used for reliability testing.

According to Chelsea Goforth (2015), from the Statistical Consulting Associate of the University of Virginia Library, "Cronbach's alpha is a measure used to assess the reliability, or internal consistency, of a set of scale or test items. In other words, the reliability of any given measurement refers to the extent to which it is a consistent measure of a concept, and Cronbach's alpha is one way of measuring the strength of that consistency."

Cronbach's Alpha

Cronbach's alpha is a measure of the internal consistency within a group of items. It determines how close and similar those items are in representing a unified theme.

$$\alpha = \frac{N\bar{r}}{1+(N-1)\bar{r}}$$

Where N is the number of items in a group and rbar is the average inter-item correlation.

Analysis

In the questionnaire, there were 17 questions which 2 were open-ended questions. Because of that, only the first 15 questions were used to determine the questionnaire's consistency.

Seen in Appendix II was the correlation matrix of items in the questionnaire. Below, was the computation of Cronbach's alpha:

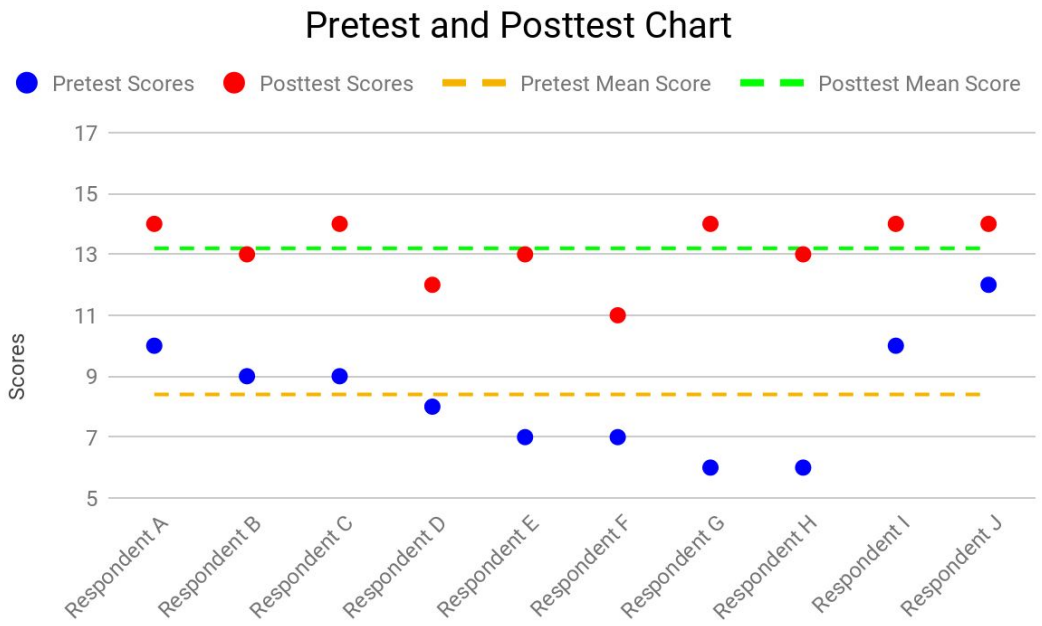
$$\alpha = \frac{15*0.4051969}{1+(15-1)*0.4051969} = 0.910861$$

The computed value of the Cronbach's alpha was 0.910861. In convention, 0.7 is the minimum acceptable value for Cronbach's alpha so a value of 0.91 implied an excellent internal consistency of items.

The multiple choice questions that were tested for reliability through the Cronbach's alpha and were used in the pretest and posttest questionnaires were presented in Appendix III.

The pretest and posttest questionnaires were given to the respondents before and after application testing. This set of respondents were those who were taking programming courses and degrees, given that they possessed a Linux machine. A link where they were able to download the ZIP file for the application was provided to them, and they were given 5 days to play and evaluate the game. All in all, a total of 10 respondents were able to provide input to the survey in a span of five days. The results that were yielded from the two surveys were presented in the table below.

Figure 2. Results from Pretest and Posttest



The figure demonstrated that the scores of the respondents in the posttest questionnaire had increased significantly compared to the scores of the pretest questionnaire. From a mean score of 8.4 in the pretest questionnaire, to a mean score of 13.2 in the posttest questionnaire, with 17 being the highest possible score, the results showed a 57% increase in the respondents' mean scores after application usage.

Another survey was provided to a set of respondents to be a measure of the usability of the game. This was the survey adapted from the one used by Delos-Reyes. This game usability test was provided to the respondents after application testing, along with the posttest questionnaire, and this served as their evaluation of the game. The results that were yielded from the said survey were presented in the table below.

Table 1. Results from usability testing

CRITERIA	MEAN RATING	TOTAL POSITIVE*	TOTAL NEGATIVE**	TOTAL COUNT
Game Playing				
The game is simple.	5.1	70.00	30.00	10
The game is challenging.	5.2	70.00	30.00	10
I was able to complete the tasks and scenarios in the game.	5.1	70.00	30.00	10

It was fun playing the game.	5.5	80.00	20.00	10
Overall, I am satisfied with the game.	5.8	80.00	20.00	10
Average for Game Playing	5.34	74.00	26.00	
Game Software Technicality				
The interface of this game is suitable and functional.	5.6	80.00	20.00	10
The background music/sound is appropriate.	5.7	80.00	20.00	10
The information provided in the game is clear and easy to understand.	5.6	80.00	20.00	10
The game has all the functions and capabilities I expect it to have.	5.5	80.00	20.00	10
Drawing and scenes incorporated in the game complement each other.	5.1	80.00	20.00	10
Whenever I want to go back to a particular segment of the game, I could do so easily and quickly.	5.5	80.00	20.00	10
Average for Game Software Technicality	5.5	80.00	20.00	

*Total Positive represents the percent of respondents who selected from the range 5-7 (Agree) in the linear scale. **Total Negative represents the percent of respondents who selected from the range 1-3 (Disagree) in the linear scale.

In the Game Playing sub category, a mean rating of 5.8 and 5.5 were recorded in the user's measure of overall game satisfaction and enjoyment, respectively. This showed that with a 5.8 and a 5.5 score in a scale of 7, the users affirmed that the game was satisfactory and fun. The respondents, however, gave the lowest agreement for the first and third categories, which contained the questions on simplicity and game completion. It was most likely that they had given the lowest score for the simplicity of the game due to the rules regarding the bonus words formed in the crossword stage of the game, and its corresponding hint points in the succeeding maze stage. It was also likely that they were not able to complete the tasks at hand in the game at their first try because some of the questions required memorization of terms. The respondents might have also found the game moderately challenging, giving it a mean score of 5.2. Overall, the respondents had given their game play experience a mean score of 5.34, with 74% of the participants on the total positive.

The sub category Game Software Technicality had the higher mean agreement rating, albeit by a slight margin. It appears that the respondents had taken a preference for the background music played during the game, as it recorded the highest mean agreement score; although, others had also provided the suggestion that a variety of music be played. The lowest

mean score was given to the game's drawing and scenes, and its complementation of each other, with a score of 5.1. This was probably due to the varying themes or motif of the game, where, for example, the menu screen of the application had an arcade-like theme, whilst the maze part of the game had a medieval-style theme. There is always room for improvement, especially in this area, and a much better overall theme for the game could be achieved with more creative talent added. The mean agreements for the other categories were fairly close to each other, having the moderate scores of either 5.5 or 5.6. It can be inferred that the users were satisfied with the interface and functionalities of the application, and they transitioned from the various segments and states of the game with ease, and also, grasped the given information accordingly. Overall, it could be concluded that 80% of the respondents gave a mean agreement rating of 5.5.

Comments and Suggestions for Improvement:

A number of comments and suggestions for improvement were also noted, during the usability testing. The following considerations were suggested:

- Adaptive screen resolution
- More levels or stages
- Varying background music
- More maze rooms
- Button click and hover sounds
- More maze backgrounds
- Lesser file size, if possible
- User creation functionality
- Varying difficulty
- Clear all data functionality
- Additional coding exercises

V. Conclusion

After having overall satisfactory results yielded from the surveys, it can be concluded that the developed application "Less Hell with Shell" is an ideal tool that could be used to help people, especially those who take programming courses and degrees, and those who are inclined to programming. This would help provide them with mastery in the fundamentals of shell scripting, which is relatively essential in the field of computer science.

Likewise, the development of this application, or game, also expanded the scope and diversity of education, and also provided much more opportunities for game-based learning, which continues to grow and evolve in this modern age, and integrates an educational and entertaining approach. This is especially useful and convenient for situations like the current pandemic, where platforms like this would serve as a tool for education, as a counter to the hindrance that the COVID-19 disease had caused the country.

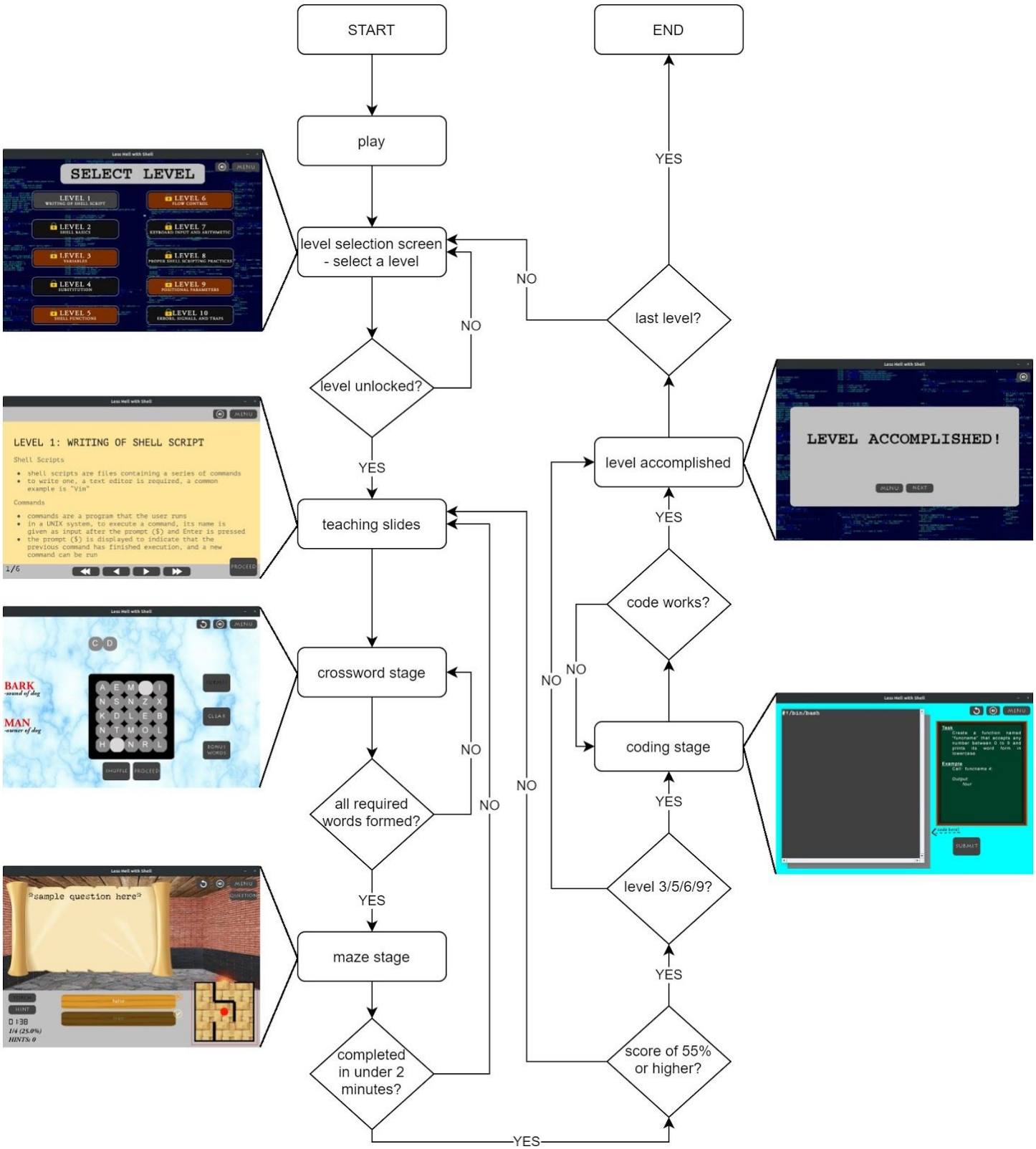
VI. Recommendations

The application "Less Hell with Shell" was created without a game engine, and since it is a game, it is recommended to further develop the game using common and modern and game engines like Unity and such. If possible, the application could preferably run in a Windows operating system, since it is an operating system that is most widely used. Optimization would also improve the application, for better execution without sacrificing the resources required and

the load on the machine and memory. To lessen the file size of the application, proper bundling of the Java or JDK binary must be done, which Oracle Java has currently been working on with its jpackage utility, a tool which will help in more convenient java bundling in applications. Also, better graphics, sprites, and assets could be made for the application if there are dedicated creative artists involved in creating it.

VII. Appendix I

Game storyboard



VIII. Appendix II

Correlation matrix of the items in the questionnaire

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	1.0000000	NA	NA	NA	NA	NA
[2,]	0.4498677	1.0000000	NA	NA	NA	NA
[3,]	0.2182179	0.2721655	1.0000000	NA	NA	NA
[4,]	0.5947617	0.7766432	0.2182179	1.0000000	NA	NA
[5,]	0.4830459	0.3333333	0.3333333	0.54916965	1.0000000	NA
[6,]	0.3162278	0.4594683	0.5091751	0.25197632	0.6683313	1.0000000
[7,]	0.4309458	0.2041241	1.0000000	0.37796447	0.6082763	0.4663690
[8,]	0.3895969	0.6123724	0.1666667	0.21821789	0.4330127	0.2886751
[9,]	0.4285714	0.3563483	0.2182179	0.42857143	0.2519763	0.4364358
[10,]	0.3779645	0.3726780	0.2721655	0.35634832	0.4216370	0.5000000
[11,]	0.4045868	0.3118048	0.5091751	0.57043565	0.4199868	0.2554952
[12,]	0.3380617	0.5163978	0.3333333	0.21821789	0.5033223	0.2309401
[13,]	0.4485426	0.5109903	0.3333333	0.25197632	0.5557777	0.5212165
[14,]	0.2781743	0.6666667	0.2721655	0.08908708	0.1178511	0.1178511
[15,]	0.2314550	0.2886751	1.0000000	0.32732684	0.4183300	0.5863020

	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	NA	NA	NA	NA	NA	NA
[2,]	NA	NA	NA	NA	NA	NA
[3,]	NA	NA	NA	NA	NA	NA
[4,]	NA	NA	NA	NA	NA	NA
[5,]	NA	NA	NA	NA	NA	NA
[6,]	NA	NA	NA	NA	NA	NA

[7,]	1.0000000	NA	NA	NA	NA	NA
[8,]	0.1767767	1.0000000	NA	NA	NA	NA
[9,]	0.2672612	0.2182179	1.0000000	NA	NA	NA
[10,]	0.2041241	0.4082483	0.87287156	1.0000000	NA	NA
[11,]	0.5143766	0.6123724	0.80178373	0.5270463	1.0000000	NA
[12,]	0.4690416	0.0000000	0.21821789	0.0000000	0.6582806	1.0000000
[13,]	0.3427827	0.2886751	0.54916965	0.5868939	0.3944053	0.5033223
[14,]	0.2886751	0.1020621	0.08908708	0.1666667	0.3632416	0.8164966
[15,]	0.7416198	0.2500000	0.53452248	0.5000000	0.4506939	0.5000000

	[,13]	[,14]	[,15]
[1,]	NA	NA	NA
[2,]	NA	NA	NA
[3,]	NA	NA	NA
[4,]	NA	NA	NA
[5,]	NA	NA	NA
[6,]	NA	NA	NA
[7,]	NA	NA	NA
[8,]	NA	NA	NA
[9,]	NA	NA	NA
[10,]	NA	NA	NA
[11,]	NA	NA	NA
[12,]	NA	NA	NA
[13,]	1.0000000	NA	NA
[14,]	0.6236096	1.0000000	NA
[15,]	0.4183300	0.4082483	1

IX. Appendix III

Table of the set of questions and its corresponding choices

Question	Choices		
What is a shell script?	a blueprint or prototype from which objects are created	a file containing a series of commands	a file containing hash strings
What is the command to make a shell script executable?	chmod 775	chmod 69	chmod 755
What is the symbol used as command separator, indicating where one command ends and another begins?	.	!	;
What is the input typed in the prompt (command line) to execute shell script?	<file_name>	gcc <file_name>.c	./<file_name>
What is the shebang to invoke Bourne-Again shell?	#!/bin/bash	#!/bin/sh	#!/bin/tcsh
What is not a syntax for invoking a function?	<function_name>	<function_name>;	<function_name>()
What is an example of a comment in a shell script?	#!/bin/bash	// this is a comment	# this is a comment
The shell function definition must appear before function call.	false	true	
A function must contain at least one command.	true	false	
What is the proper way of referencing a variable?	<variable_name>	$<variable_name>	&<variable_name>
What is the proper way of defining a variable?	<variable_name> = <value>	<variable_name> =<value>	$<variable_name>= <value>
What is an example of making a variable's value unchangeable?	const varname	let varname	readonly varname
What is an example of making the shell "forget" a variable?	unset varname	varname=null	deallocate varname

By default, aliases are enabled in Bourne-Again shell.	true	false	
Which are not allowed to be used in variable names?	spaces	punctuation marks	all of the above
Write an example of an alias.			
Convert the alias you have given to a function.			

REFERENCES

Conrad, S., et al. (n. d.). *Terminus: A Text Adventure Game To Promote Learning About Terminal Command Line Interfaces*. Retrieved from http://web.mit.edu/mprat/Public/web/Terminus/Java/CMS.590Game2FinalReport.pdf

Dadheech, A. (2018). *The Importance of Game Based Learning in Modern Education.* Retrieved from https://theknowledgereview.com/importance-game-based-learning-modern-education/

Davison, A. (2003). *Games Programming with Java and Java 3D.* Retrieved from http://www.comscigate.com/tutorial/ajay/Part%201/java%20games%20programming%20techniques/fivedots.coe.psu.ac.th/~ad/jg/intro/gamesJava.pdf

Davison, A. (2005). *Killer Game Programming in Java*. Sebastopol, California: O'Reilly Media, Inc.

Delos-Reyes, L.A. (2018). *2D Programming Using GDevelop: Agricultural Marketing*. University of the Philippines Los Banos. 2-5.

Goforth, C. (2015). *Using and Interpreting Cronbach's Alpha*. Retrieved from https://data.library.virginia.edu/using-and-interpreting-cronbachs-alpha/

Moore, M. E. (2011). *Basics of Game Design*. Boca Raton, Florida: CRC Press.

Robbins, A. & Beebe, N. H. F. (2005). *Classic Shell Scripting: Hidden Commands that Unlock the Power of Unix.* Sebastopol, California: O'Reilly Media, Inc.

Saternos, C. (2005). *An Introduction to Linux Shell Scripting for DBAs.* Retrieved from https://www.oracle.com/technical-resources/articles/linux/saternos-scripting.html

Sethi, S. (2019). *Top Reasons To Master Unix Shell Scripting In 2016.* Retrieved from https://www.edureka.co/blog/top-reasons-to-learnunix-shell-scripting/

Squire, K. (2013). Video Games and Learning: Teaching and Participatory Culture in the Digital Age. In *Alberta Journal of Educational Research 59* (pp. 129-132). Manhattan, New York: Teachers College Press.

Torrente, J., et al. (2010). IEEE EDUCON 2010 Conference. In *Introducing educational games in the learning process* (pp. 1121-1126). Madrid, Spain: IEEE.

Veeraraghavan, S. (1999). *Sams Teach Yourself Shell Programming in 24 Hours.* United States of America: Sam's Publishing.

Virvou, M., et al. (2005). Combining Software Games with Education: Evaluation of its Educational Effectiveness. In *Educational Technology Society 8* (pp. 54-65). Piraeus, Greece: International Forum of Educational Technology & Society.

About the Author

Earl Joseph A. Palapar is a BS Computer Science student of the University of the Philippines Los Baños. He currently resides at Las Piñas City together with his parents, 2 sisters, and brother. He is most proficient with the programming languages C and Java. Someday, through the skills he attained in the university, he hopes to create something that will benefit his nation.