Paul Ippolito

Pablo Rivas

CMPT220

4/30/17

Project 2 Write-up

Abstract

This paper contains the write-up for Project 2, my text-based game. It will go in depth as to how everything was implemented, my motivation behind choosing this type of project, and the "user manual" for the game.

Introduction

My basic motivation for choosing to do a text-based game was based on the principle of "doing it again, but doing it right". I had attempted to make a text-based game back in my senior year of high school when I first started learning how to program and code. I was extremely unhappy with the end result. The program was very buggy and was overall not well received. Now that my programming skill and understanding of coding has improved, I thought it would be a good idea to try again. I believe compared to the last text-based game I made, this one is a huge improvement. It works much more smoothly than the last one and I am very pleased with my work. The game is fun, mostly in part because it actually works this time around.

How the Game Works

The game itself is contained in one class named Krogon. This class has two methods named player and game. The main method of Krogon, and in fact the entire class, just calls the player method after displaying the game's title. This was due to complications of getting the game to restart after completing it. Once the user reaches "GAME OVER", they will be prompted if they wish to play again. If the user types anything but "y" or "Y", the game will terminate. However, if they do type either of the previously mentioned letters, the player method will be called again, which will then in turn call the game method again, effectively restarting the whole process without having to terminate and re-run the program. The player method prompts the user to type a String that will represent the player's name and then gives the game's introduction using the player's inputted name. The user can type any number of characters, even numbers. The name can absolutely be anything the user inputs. Afterwards, the method calls the game method, and the main game itself begins. All of the locations the player can access are contained within separate Strings, and most locations have their own corresponding Boolean variable to indicate whether or not it has been visited by the player or not. The only locations that do not have this feature are the two end locations within the game. These Booleans are to prevent the player from receiving points from the same location twice. All of these Booleans with the exception of the start location are set to false and the player's points are initialized at zero. The player also has a limited number of moves, set to 20. Every time the player does something, even checking things like points, they lose one move.

The game itself is run through a very large While-true loop. At the start of the loop, the game prompts the user to either enter a direction or command, held by a string value named cmd. The player can type either the one letter commands "n, s, e, w, h, p, x, t, or q". They can also

type the corresponding words for these commands, “north, south, east, west, help, points, search, take, or quit”. Depending on what the user types, they will either travel to the next location if their direction corresponds to a location, search for an item in their current location, take an item if it’s present, ask for help, or quit the game. Otherwise the program will tell the player they cannot do that and use up a move regardless. If the player does make it to a certain non-loop breaking location, the game will check the Boolean variables. For example, when the player travels to any location from the start, it will first change the player’s location based on what direction they input. Then it will check if the Boolean variable is true or false. If it is false, the variable will be set to true and the player will be rewarded five points. However, any other time they visit that location it will be true, resulting in no points being given to the player. This holds true for all locations that do not break out of the loop. After checking for these conditions, the loop displays the player’s current location, contained within String currLoc, and then the loop starts once again. This will continue until either the player runs out of moves, quits, or reaches one of the two ending locations. One of these locations is the boat which allows the player to escape, thus resulting in a “win”. The other location is where the monster Krogon is. Depending on if the player has the correct item or not, the player will either kill the monster resulting in a win or die, resulting in a loss. Regardless, both if these locations, when the right conditions are met, will break out of the loop and then reach the end game. It will display the ending depending on what items the player has and which location they finished with. The endings are completely dependent on what item the player has. For example, if the player has the RPG and travels to Krogon’s location, the player will kill Krogon. However, if the player does not also have the “cash” item, it will result in a neutral ending and inform the player that they missed an item. Afterwards the program will display the game over message, notify the player how many points

they completed the game with, and then prompt the user if they wish to play the game once again. Getting items within the game is very simplistic, as it is also dealt with through a large series of if-else statements. When the player searches or takes an item, it first checks which location the player is at. If that location has an item, it will inform the player there is an item or take it depending on the command given. Only if the player already has the item will the command not work. So if the user already has the axe and tries to take it again in the location, it will not let you.

The Design of the Game

Whilst creating this game, I wanted to pay homage to the early-mid 80's era of video games, where little to no instruction is given to the player. As a result, there is no map that can be found in the game. That and it made sense given that the player is in a demolished city in which not much is left standing. Either way, the player is more or less left guessing as to which way to go. Two of the locations are also locked. By this I mean they require a specific item in order to be accessed, otherwise it would be no different than if the user input something invalid. These locations include the boat area that will break the loop and the secret area where the cash is located. Since both of these locations are very close to the start location, I did not want the game to be finished so quickly. Before implementing this feature, the game could be completed in a mere five seconds with the only downfall being a minimal score. The player can still attempt to reach these locations, but without the correct item, the game will not change the player's location. So if you travel west from the harbor, it will take you to the boat, unless you do not have the key. In this case, run through an if-else check, the player's location will remain at the harbor and the game will inform them that a key is necessary. The same holds true for the secret area. If the player tries to travel east from the starting area but lacks the axe item, the player will

remain at the start and the game will inform the player that they need an axe to break the wall blocking their path. All of these will of course result in the loss of a move, which happens every time a user inputs a command, even invalid ones. As a result of these two locked locations and where the corresponding items to unlock them are, it requires the player to visit 9 of the 10 locations within the game in order to get the best possible ending. The ending is dependent on which location you choose to go to once you have the correct items. Having the cash guarantees a good ending unless Krogon kills you, but the amount of points you get depends on the location. Killing Krogon grants more points than simply escaping the city, but doing either while in possession of the cash will result in the "best" endings.

User Manual

The game will only accept the commands in lowercase. So typing "north" is fine, but typing "North" will be interpreted as an invalid command to the game. This holds true for all of the commands within this game. Follow this lest you spend 20 minutes struggling, run out of moves, and get a game over without having actually done anything. Another issue to note would be that you do not have to input the name of the item you want to take, nor is it recommended. The way the command is programmed is to only work if you are in a location in which the item exists. The only time this command will not work is if you already have this item in your inventory. To avoid wasting moves, the search command is not entirely necessary to use. If you already know that an item is there, you can just use the take command without having to search for it first. Also, for when the program asks the player if they wish to play again, it will only accept "y" or "Y". Anything else will terminate the program, but not cause an error.

Conclusion

This project's significance is that is practice for designing games for me. While making this project, I had to design the flow, story, and overall difficulty of the game. While the problem it solves is nothing more than a "toy" problem that one can spend about ten minutes of their life playing, for myself it is good practice in the long run. The game is simplistic in its design, but as someone who wishes to go into game design as a career, this program is a good start to getting into the field of video game design.