Gerard Andrews

4/29/2017

Prof. Rivas

Software Development I

Final Project Writeup

For Project 2 my final project is an application project in the form of a text based game. The game is officially complete and I have completed the additions that I stated in my milestone. For me the importance of this project and how to make games like these in different programs. Finally, this project will help me create more complex games in the future.

This application project is a text based game that relies on multiple classes and runs on a game loop. The basic idea for conditions is that the player must get to a certain location with all of the items in the inventory. This must be done before the player runs out of moves.

The system relies on multiple class files now that bring in important variables that the game uses. There are three total: Player, Locale, and Items. They all contain variables that the program requires using to run properly. The Player class brings in the player's move count, which is how many moves the player can make before he loses, it brings in the player's total score, and it brings in the player's current Location relevant to the game. The Locale class brings in the amount of locations in the game, the Booleans for whether the player has visited a location or not, and the location descriptions. The Item class contains the item names, locations, and item descriptions.

All three class files are brought into the driver and are made public for them to be used in multiple methods. Also, the variables from the classes are also made public because they are going to be used in the main method as well as the other methods in the driver. Add several variables from the main method, they make the game function without any errors getting in the way.

The Main method is where most of the work takes place. It uses a game loop, which is a while loop that has three functions, the functions are: 1) Get and Process Input, 2) Update the Game State, and 3) Render a Scene. Through this idea, the user will input a command, the game will take in that command to see where it fits and updates the player, location, and item variables and then will print a scene or description relevant to where the player is and what the player does.

The first function Get and Process Input is where the player's participation is required. The player will be prompted to enter a string which will then be brought then be determined if the string is a proper command according to the game. If the player does not know the commands, he/she can always type "help" or "h" as the input to see a list of commands.

The second function Update the Game State is where majority of the work takes place. If the input is an actual command it will go through different methods to see the output of the command. If the player types directional input such as "north" or "south", the command will put into a method that determines if the player can go somewhere based on the Player class's current location. If this is possible the player's current location will be updated and the loop will head to the third function. It will also subtract from the player's total moves which if it hits 0 the player has officially lost the game. If the player types "take" or "examine" the game will determine based on the player's current location whether there is an the Item classes item location array where the player is at. If an item is there the player will be informed about and the steps in the loop will be taken to either add the item to the inventory or tell the player that the item is there.

Some locations are initially blocked off from the player and to get to these locations they need to have an item in order to get there. This very simply done by using methods that determine if the player has items in the inventory. If they do they will be told that because they have this item they can enter. If they don't they'll be told otherwise and it will be hinted at what item they need to enter.

Some commands such as "help" or "quit" refer to method of their own which simply print strings. Another command called "map" will bring up a text based map of the game ONLY if the player had a map in the player's inventory, which is an array list that will have items added to it as the player takes items because it starts as an empty list.

The third and final function is to render a scene. This is simply done by printing a description of the location that the player is currently in. This uses the Locale classes array of descriptions as well as the Player classes current location.

So, what's been added? The biggest addition is something that makes the driver have less lines, a matrix. This two-dimensional array called paths, contains all the locations and where they can go from that location. It removes a lot of else if statements from the code and it still functions as intended. So, when the game checks for locations, it checks the matrix for where the player is, and based on what direction the player types it checks where the player can to a location in that direction. This is then returned in a method that updates the game state, which returns the player's current location.

Many of these variables use the same amount of numbers in their lists. This is made specifically so I can determine certain variables based on the player's current location. So, if I go to location 6 and I type “examine” or “x” the game determines if the player’s current location has an item in that location. Some commands like “help” and “quit” don’t need to rely on that.

As a result of what is required, the game must always have the additional classes in order for the game to run. If these classes are not provided to the game, the game will not run. The other main requirement is input and participation from the player. There are no real physical requirements.

In terms of other games out there. This is a game that takes after another text game called ZORK, it has similar idea of a text based adventure will the player types input and the game updates. ZORK has other commands when the player is in certain locations such as climbing trees in forests and trying to find objects by looking around and looking for other significant areas, while it does keep track of your move count there is no limit. My main difference between ZORK and my game is that it has a limit to how many moves the player makes. If it goes up to 50 moves the game is over and the player must restart.

Here is the user manual. To get the game started, the user must make sure he has the main driver file, the Game file, the Locale file, and the Items file all in the same folder. As long as they can all be found. They game can be run without any issues. If they are not together the game won’t even compile. The game has no command errors and does not use exceptions as a result.

In conclusion, the game has made further leaps and bounds. It has become much more efficient with the addition of the matrix. It has also become more efficient with the addition of classes which help in creating instances of the locations and items. And finally, it’s become more unique as development progressed and is now complete.