COMP 1405-Z Final Project Analysis

By: Amelie Haefele

DnD Game

Introduction:

This program allows the user to play a turn based Dungeons and Dragons text adventure game. This includes two normal fights and one boss fight(larger final fight) that has a different fighting style. The player chooses what they want to do in battle and the two NPC's(non-playable characters) on the players team which each have their own battle AI.

Code Analysis:

This program relies heavily on the structure of the code. The structure is kept well organized and as uncluttered as possible by using modular programming. It uses many functions to complete logic and keeps all description text in a file to further declutter. This starts off by having a main loop in the main function(play_game) that keeps the game playing. From there it gets all its code from `def getGameData(player_name):` so the main information can be stored away from the logic. From there it extracts this data it will need with three outputs: return `good_guys`(a dictionary with player and their ally stats)`, stages_list`(a list of stage information, see program)`, experience_points`(the int tracking experience points) which is all the data the game needs to run.

The game will then enter a loop that will continue until all the stages have been completed.
For the incoming fight it will print out the intro text and save a deep copy of the good_guys and bad_guys which will allow the game to return to the beginning of the last stage of the fight if the player dies and chooses to continue. After this it will call the combat function.

This combat function controls all fighting in the game and is called `fight`. It will start it's loop by first checking for a fight end case: if the player has died or if all of the enemies have died. Then it will call another function to print out useful information to the player before beginning any turns. The player goes first and is prompted to choose their action, at which the program will check if the action exists and if the player has enough action points to perform said action; and will re-prompt the player if either are invaild.

Then it will move onto the NPC allies' turn's; which is it's own function. Each NPC has their own fighting style. For example Peter will first check if anyone on the team needs to be healed before an attack while another NPC, Danielle, will go straight into attacking. This leads to

more complexity but an inability to make the code here more general. All bad guy NPC's apart from the boss fight will attack the NPC or player with the lowest health; if all health's are the same they will attack the player.

After each action if performed it will be printed out and saved into the output file allowing the player to better plan their next move and to reflect on after the game is finished. When the fight is over and the player has killed each enemy, the player will be able to level up if they reached greater than or equal to 800 experience points. The level up function will be called allowing the player to choose to increase their health or action points by 5. Each NPC will also be given the chance to level up where the individual NPC AI will choose what they want to level up, for example Peter choses to level up action points first.

For winning the player will also be awarded coins, based on experience points, and loot, based on a leveled loot system. Depending on the player's level the leveled loot system will give the player a new action. After this is all done a new stage will start looping through like the first but with updated information, text to read, and enemy team information.

If at any point the player dies the game will prompt them whether or not they want to continue. If they input "no" the game will end. If they input "yes" the fight stats will be reset and they will start back from the most recent fight. The save system allows the user to go back to the most recent battle with the saved statistics to avoid repetition.

Simulation is also implemented into this code in the form of unit tests. Tests are performed on several methods to ensure their success by simulation different situations in the code.

Important: Please note that this is just a general overview of the code and many functions and actions are not mentioned in the explanation.

Improvements:

I would like to have added a little more depth to the fight mechanics by possibly implementing a blocking and or critical features. However I'm happy with the overall quality of my code as it's structure is well done and is fast.

Here is a rough illustration of said structure of functions:

Time Complexity Analysis:

Almost all player and NPC information is searched using a dictionary, to save time. This lookup is O(1) time complexity. I have done this by purely indexing through and storing almost all information in dictionaries. However, these dictionaries take up more space than lists but are faster. This program is overall O(n) based with respect to the number of NPCs.

I could've used many different sorting methods to find the highest and lowest health targets, however it would not be as efficient so I choose not to.

Overall Code Complexity:

The difficulty from this code comes from making the algorithm structure, data structure, and overall logic of building the game. By having different NPC allies act differently it adds to the overall complexity of the code by creating a separate AI for each, see

This project shows usage of following course concepts by using:

- -creating complex, yet simply structured algorithms
- -showing an intricate knowledge of dictionaries and how data structures are best stored
- -using logic in the form of logic gates:
    - -if statements
    - -while statements
    - -else statements
    - -the use of return statements and breaks
- -shows understanding of nested loops and data structures
- -the use of file reading and writing
- -using modular programming throughout
- -clear understanding of functions and how they work
- -added simulation through the use of tests

Extra Story Analysis:

The story starts off with you and your two best friends as kids. As each fight progresses you get older until the boss fight. The boss's name in Latin is responsibility so in essence you're fighting growing old and being responsible. You first try to take responsibility/adult life all on your own but your friends come and stand by your side to help you. In the end you win as you and your friends have all become adults.

There are some other story elements; such as Danielle thinking the game(DnD) was lame at first but then learning to have fun without worrying about what others think, and references(Monty Python) throughout which I won't explain here.