Programming Project – Army of One – A Text Based Adventure Game In Python

To show what I have learned during the first term using the Python programming language, and my first introduction to programming in general, I aim to create a text based adventure game.

**Description of the Project**

Text based adventure games have been around since the advent of modern home computing, and were among some of the first games created for home audiences, so with that in mind it is a good place to start as a first short project. It will include elements of everything I have learned during the first term, from utilising strings and user input, using loops, lists and dictionaries, and heavily utilising file handling. I will also be using classes and some object orientated programming which goes a little beyond what we have learned so far but will be extremely useful for what I wish to achieve.

The aim of my game will be to move a player object through a series of rooms, battling enemies and collecting loot that can help you progress, towards the goal of defeating the final boss.

**Aims and Requirements of the Project:**

* The user will be able to move the player object through the game world:
* The player object can be commanded using user inputs to move around the game world. I want to achieve this using a while loop and the enumerate function while iterating through an adjacency list (using a dictionary as the data structure), that can keep track of the location the player object is in and allow the player object’s location to be moved through the room objects.
* The user will be able to attack enemies with the player object:
* The player object can interact through commands with enemy objects within the room objects. I want to achieve this through the use of a menu of choices ranging from the selection of melee attacks, magical attacks and a self-heal, and want the enemy object to attack the player and heal itself through a use of randomly generated move based on some conditions (if it has a certain health level, if it can still heal or attack, etc). It will be turn based with the decision of who goes first determined randomly. If the enemy dies it will return to the location menu.
* The user will be able to interact with the room objects to find items and then equip them and interact with items through the player inventory:
* The player object can interact with the room objects and items objects. I want to achieve this using a location menu that appears when there are no enemies are alive within the room object. This menu will allow the user to loot items found within the room object and append them to the player object’s inventory. The menu will then allow the user to look at the player object’s inventory, and remove items if desired. It will also allow the user to equip items and armour to the player object’s equipment slots through a sub-menu. The menu will also give the user the option of healing the player object. Finally, the menu will allow you to access the location choice screen and allow the user to move the player object between room objects.
* The user will be blocked from accessing the final boss of the game unless certain conditions are met:
* The player object will not be able to enter the final room object unless certain previous enemy objects are defeated. I want the user to have to overcome this obstacle so as to give the game a certain challenge, this will be achieved by having the program check the health status of the mini-boss enemy objects and relaying weather they are alive or not, if not then the player object can then pass through.
* The user will be able to win or lose the game:
* The program will check throughout the battle loop on the status of the player and an enemy, if the player object’s health reaches zero it will be considered dead and the program will push to the game over screen and then exit the program. If the player defeats the final boss object, the program will push to the victory screen and then exit the game. It will achieve this by checking on the player status after each turn in the battle loop and push to game over if the player object is considered defeated. At the end of the loop if the player has just defeated the final boss enemy object, by reducing it’s health to zero, it will break the main loop and push to the victory screen and exit the program.
* The game map and attributes will be imported from text file and formatted into classes:
* The program will import all its class attribute data and the game map from file. This will give the user the option to change anything about the game, from names, stats, loot and enemies, simply by editing the text file. The main game system will be kept the same, but the values and descriptions and map layout can be changed, allowing for a huge amount of variability. It will achieve this by reading from text files into a for loop that can iterate through each line and then split the data into lists to which can intern be iterated over. The data can then be processed by formatting the list data into the classes, so the program can then process them.

**Design of the project**

The following is a flow diagram to represent the way the program will run through:

Welcome to the game

Start game loop or exit, input name to begin – breaks with victory or game over.

Character starts at first location on adjacency list.

Proceeds through dungeon through use of looping through adjacency list (dictionary).

Access inventory, equip weapons and armour, heal.

Fights initiated, use of weapons and armour, if enemy dies the room remembers that it’s dead if the player dies the loop breaks back to the start.

Mini-bosses must be defeated to access final boss

GAME OVER

DEFEAT

VICTORY

Get to end of the dungeon, last room has the final boss. Impassable unless mini-bosses defeated.

FINAL BOSS

Exit program

Below is a series of class diagrams showing how the various classes fit together, which classes inherit from others, and how the text files are utilised with the classes:

Player Class

(All player attributes), (battle cycle), (victory), (game over).

Formatting for the game map, Room objects placed inside

Text File

The game map

Room Class

(name), (description), (enemy class), (items class)

Text File

Details for all Rooms

Enemy Class

(name), (description), (all attributes)

Items Class

(name), (description)

(Items)

Armour Subclass

(reduced damage taken)

(Items)

Potions Subclass

(Heal Amount)

(Items)

Weapons Subclass

(Attack bonus)

Text File

Details for all Enemies

Text File

Details for all Potions

Text File

Details for all Weapons

Text File

Details for all Armours

**Testing the Program**

The testing table below shows the part of the program tested and how it was done.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section Tested | Function Tested | Expected Output | Actual Output | Further Action |
| Enemy Module | Tested the unpacking function. | Each enemy in enemy text file formatted into class objects and placed into a list that can be iterated over. | Functions as expected each enemy from the file added to the list as a class object. | As this works, it means it will work when used in other modules that require it. |
| Items Module | Tested the potion unpack function. | Each potion in potion text file formatted into subclass objects and placed into a list that can be iterated over. | Functions as expected each potion from the file added to the list as a class object. | As this works, it means it will work when used in other modules that require it. |
|  | Tested the weapon unpack function. | Each weapon in weapon text file formatted into subclass objects and placed into a list that can be iterated over. | Functions as expected each weapon from the file added to the list as a class object. | As this works, it means it will work when used in other modules that require it. |
|  | Tested the armour unpack function. | Each armour in text file formatted into subclass objects and placed into a list that can be iterated over. | Functions as expected each armour from the file added to the list as a class object. | As this works, it means it will work when used in other modules that require it. |
| Room Module | Tested the room unpack function and class format function, both work in conjunction. | The unpack room function iterates through the room text file and formats it. The format class function returns the class objects for items and enemies as described in the text file to the room object, and the room object is then appended to a list of room objects, that can be iterated through. | Certain problems after trying to return the items and enemies classes to the room. | Had to unpack the enemies and items into the format function to be able to format them in and then specifically format the exact index in the room line list to get it to function correctly. |
| Player Module | Tested the victory function. | The victory function should check if location is at room17 (the final room object) and checks if the enemy is alive (hp > 0) if not it returns victory = true. | Functions as intended. | Can be used in the main loop. |
|  | Tested the game over function. | Returns the game over screen. | Functions as intended. | Can be used in the main loop. |
|  | Tested the looting function. | Allows user to move item objects from the room object to be placed into the player objects inventory and removes the item object from the room object. Only accepts valid commands. | Functions as intended. | Can be used in the main loop. |
|  | Tested the inventory management function. | Allows user to view the player object’s inventory, remove items from said inventory, equip weapons and armour to the player objects equipment slots, and exit the menu. Only accepts valid commands. | Functions as intended. | Can be used in main loop. |
|  | Tested the location menu function. | Allows user to exit to choose the next location, allows access to the looting function, allows access to the inventory menu, allows use of healing spells outside of battle cycle, allows use of healing potion objects outside of battle cycle. | Functions as intended. | Can be used in main loop. |
|  | Tested the attack cycle function. | Enters the attack cycle between player object and the enemy object within the room object at the current location. Allows the user to interact with the menu to attack, heal, deal “magic” damage increased based on how low the enemy’s health is, and use a potion object to heal the player object. The enemy can attack back, with randomly chosen moves, with the turn switching after each move is made. | Functions as intended. | Can be used in the main loop. |
| Game Loop | Tested the open map and format variables function, both work in conjunction. | Opens the game map text file to iterate through the lines and format the room variable names within. Places these variable names into a dictionary to be looped though for the location choice. | Functions as intended. | Can be used in the main loop. |
|  | Tested the game loop. | Runs through the main loop, all functions working together to create the main game, correct inputs all chosen. | Functions as intended. | All game functions work under the correct conditions. |
| Intentionally Inputting the Wrong Commands | Tested on each user input in the game. | Each unintended input should generate an “invalid command” line and loop back to ask again. | Name input accepts only enter as an input.  Looting function generating error.  Inventory menu loops again correctly but displays no invalid command text. Healing in location menu not exiting correctly, also no error message. | Corrected all checks and put in error checks for what was missing. |
| Text Files | Tested removing the text files so they were missing, and using empty text files. | Unaccounted for, so this will cause an error. | Causes errors, one an IO error for the missing files and errors iterating through lines for empty files. | Need to create functions to help exit the program gracefully when it encounters missing files or empty files. |

**Critique of the Program**

After running through the game many times over, I have got it to a point I am pleased with. It has achieved the aims I wanted to with it, however there are many parts I believe I could have done differently and certain aspects which I would like to have improved up considerably. Testing wise, everything I have been through has got the program to a stable point where it can handle any basic input for each individual menu and choice without breaking, and can handle closing gracefully if a critical component of either the text files or lines within the text files, are missing. Going through the aims one by one, I will explain what worked, what didn’t and how I might do things differently if doing something similar again.

Moving through the game world worked well with using an adjacency list as a dictionary to iterate through locations. I looked at using a standard 2-D array and a co-ordinate based system, but I felt it was too static always having 4 directions you could go, whereas in the adjacency list, each location could have multiple connections to others, allowing for the feeling of a “pseudo 3D” effect of the world, thus descriptions can use ascending and descending as well as moving on the same plane. What I would change about it, and this goes for any of the room interactions and some of the menu choices, is have the user type words on which way they want to move, like “go through the iron door”. Now that is a much more sophisticated level of what is known as “text parsing” and requires the program to look at the verbs a user has typed, it goes beyond what my current knowledge and level of expertise is, but I am very interested in finding out exactly how it works and incorporating levels of it into future endeavours.

Attacking enemies and interacting with the items in the game world worked as well as I hoped, and I found the combat system fun to implement and use. Some aspects were a little harder to set up, including the percentage based damage to enemies using the “magic” spell attack, but overall it was fairly straightforward. Doing things differently I would like to have expanded the range of combat choices and abilities, adding items that did different effects, damage over time effects and critical hit and dodge chances. The only reason they aren’t present is due to time constraints and getting a finished program up and running. The one I really wanted to try but didn’t want to spend too much time on was damage over time as it probably would have added some extra complexity to the system. Also adding to this in the future is looking to implement a basic level of AI where the attacker gets more possibilities and does things based on how the user acts. Again, this is beyond the timeframe and possibly my current knowledge level, but I aim to get there.

The act of blocking off the path to the final boss worked very well. The only downside with it is its not dynamic within the program, so it must always be the exact same set of rooms, it can be different enemies though, as long as they are present in those particular rooms. If I could change it, I would have it so the user could define which room enemies had to be defeated to get into the final room. Again, it was more a time constraint that I didn’t implement this, as I’m sure it would be possible to have this within a separate text file and place the contents into the defined arguments.

With regards to the victory and game over portions of the game, they both function, however I would have preferred it if I could have made the game loop back to the beginning again and ask if you want to replay. The problem I found when doing this is I did not have the knowledge to understand how to reset the class instances back to their base states when starting over again. After looking through numerous sources it seems an idea may have been to have the program reboot itself and start over, but I didn’t like this option, as I’d prefer to be able to do it within the current process.

I am very pleased with how the text file imports came to work out. I originally intended for the user to only implement a new map and room descriptions using them, but I found out while working on them that I could use it for other classes as well, so the player could effectively change everything to do with all the attributes of the classes from only text files, never needing to have to change code at all. This makes it extremely easy to create an entirely different feeling game, without changing the underlying code at all. This is the best part of the program to me. The only problem with it is it’s currently capped at seventeen rooms, three potions, five weapons and six armours. It’s hard coded this way as I currently don’t have the knowledge (and couldn’t find any descriptions elsewhere) to be able to dynamically update the variable names to match the amount of required class objects, or the ability to dynamically update the parsing of the text files so it can look for increasing iterations of “enemy” or “potion” or any of the class variations.

The only other things I would like to have had time to add to the program or at least consider, are the ability to have multiple enemies per room, to be able to fight multiple enemies at once, and to implement multiple items per room (again time constraint with this one).

There are hundreds of other design points I can think of and many other things I would love to include, but that is semantics, and I’m confident going forward of being able to do much more with what I have learned from this, my first actual programming project.