Treasure-Hunter

Introduction

Welcome to Treasure Hunter a text-based adventure strategy game built using python, where the aim is to have the most money by the end of the game while avoiding dying to traps along the way.

Project Overview

I plan to create a text-based treasure hunting game which can be played as many times as the user wants due to all rooms generating randomly upon searching the room. For this program I intend to have treasure be money and have the player with the most money be the winner of the game this is to stop the game from randomly generating the treasure on the first tile the user or enemy visits, finishing the game straight away before the rest of the map is explored.

Challenges and Solutions

while working on this program I faced several different challenges. My first challenge I faced during development was figuring out the map I tested multiple iterations including using 3D lists as well as using embedded lists. My solution to this was to use one 3D list where position was calculated using the row and column number excluding the depth and using that to locate the next unidentified room. Another challenge I faced while developing the program was how to decide the winner, my ideas were to either have the player and enemy race for one treasure on a map or have the users race to see who can accumulate the most treasure in a specific number of turns. I opted to make the player and enemy fight to see who could accumulate the most treasure (money) while avoiding traps at the same time. I chose to do this as the concept seemed more appealing to me and gave the game a level of strategy while also keeping the adventure theme of the game. This, combined with keyboard activated responses, makes the game run smoothly and making the users experience the game like they are playing a computer game while keeping the game running without the user having to press enter for every

input they make. Another problem I faced was methods on stopping cheating this provided me with another reason why I chose to use random probability for room generation as it prevents plays from memorising the map so they cannot know where the traps or treasures are hiding.

Search Algorithms

For my program I have made use of linear search to find specific numbers within my 3D list for the game map. I chose to use Linear Search due to the fact my list will never be ordered as the list if constantly being updated due to this using binary search or interpolation sort would be ineffective because they require a sorted list to work. In addition, I used Depth First search to find the fastest path to the nearest unidentified room. As my program assigns the room upon the room being searched the user cannot find only treasure or traps making finding the nearest unassigned room the next best option

Test Evidence

during my testing I had a few errors including miss spelt variable names and a few class inheritance issues however all these issues have been fixed.
However, the largest error I faced during testing was that my classes would break this was due to adding variable names to the links which did not exist within the method this was fixed by re-arranging where the variable is called.
Another problem I had was when it came to text inputs as a result, I had to do some research into the keyboard import library and found that it while using a testing file found that the import resolved my input problem as I could have it be more responsive than having to press enter after every input was made.
The table bellow is a log of all errors and how they were resolved.

Test Name	problem	solution
Classes	Classes were not inheriting from one another correctly caused by the order of the classes	Changed the ordering of the classes within the program as the issue was the classes being inherited before the class was created
Classes	Class functions were not	Removed the variables

	being called correctly resulting in the code finishing abruptly plus variables being send to the class but not called by the class functions	from the locations where the class functions were being called as well as fixing the calling of the class functions
inputs	While testing the inputs they where not responsive enough due to having to click enter after every input	Import the keyboard module due to it recognising when keys are pressed allowing for specific keys to preform specific functions
rooms	While testing the rooms would not be 'looted' after being search and items being added to the plays or enemies' inventory	After the room was searched, the room was overridden to be empty to avoid the user being able to gain infinite amounts of items

Conclusion

in conclusion, even though I had multiple errors and issues regarding my code as explained above, my program runs well and can be played as many times as the user likes with the locations of rooms completely random each time granting a new experience. While the opponent cannot be played by a second person it was made to move around the rooms and search for treasure and can avoid traps if the tile is already discovered opting to avoid it unless the opponent is surrounded by traps. As a result, this game becomes a strategy adventure game like Cluedo due to the nature of the traps not being de-activated making the user be forced to remember what tiles have traps on and which do not.