

Treasure Hunt Game

Contents

Introduction	1
Project Overview	1
Challenges and Solutions	2
Search Algorithms.....	2
Pseudocode/flowchart	3
Test Evidence.....	4
Conclusions	5

Introduction

This report elaborates on the development and functionality of the Treasure Hunt Game I created. It is a grid-based thought-provoking game designed to evaluate players' strategic and analytical capabilities. The game combines various aspects of survival with basic but fun riddles. In this Treasure Hunt players must explore a 7x7 grid, overcome traps, collect power-ups, and locate the treasure within a limited number of moves to win the game.

Project Overview

The treasure hunt game is an interactive adventure that combines exploration, problem-solving, and decision-making. The player must explore through the 7x7 grid, which is shown as the map of a forest. To find the treasure, the player must go through a challenging journey through the forest, where they may face various traps, such as hidden pitfalls that could reduce the player's health, or stumble upon power-ups, like protective shields and health potions that could restore health, that may provide vital help throughout their adventure. These elements are scattered around in random locations within the grid. To upgrade the challenge, the game incorporates search algorithms that provide the player with clues about the treasure. Furthermore, as the player navigate their way through the grid, they must solve riddles to avoid traps. There are limited clues to find the treasure within the number of moves given in the game.

Challenges and Solutions

Throughout developing this game, I came across challenges, requiring careful attention and problem-solving. These challenges included managing the random placement for traps, power-ups, and treasures, implementing search algorithms and maintaining game flow with multiple players and limited moves.

One of the initial challenges I faced was with the random placement of game elements. It was important to ensure that elements like the treasure, traps, and power-ups were positioned randomly on the grid, without any overlap between them. Additionally, it was essential that these elements did not occupy the player's starting position. This was solved by implementing a random placement function that checked for valid positions before placing any item.

Another challenge I encountered was implementing various search algorithms. I incorporated Binary Search, Breadth-First Search, and Depth-First Search. These algorithms were designed to help the player by providing helpful clues that could speed up their progress in finding the treasure. For example, the algorithms help identify the most secure ways to avoid traps and indicate the area where the treasure is hidden, improving the player's chances of victory.

Search Algorithms

Incorporating search algorithms into the game was essential for providing clues that could help the players discover the treasure quicker. By utilizing three different search algorithms, players had multiple methods available to gather clues about the treasure's location within the grid. This provided them with a range of strategies to explore and find the treasure more effectively, enhancing the user's overall game experience.

Using Binary Search, the player would receive a clue about the general quadrant of the treasure, narrowing down the grid by 75%. This will help the player focus on a smaller area, making the search more efficient.

The Breadth-First Search works explores all paths from the starting point before progressing to the next level of depth. I incorporated this algorithm to help players identify the shortest path to the treasure while avoiding traps that could affect the player's health. This was crucial because finding the shortest route would not necessarily ensure the player's safety; by considering all paths, the algorithm maximized the chances of finding a safe route, ensuring that the player could reach the treasure without encountering too many dangers.

Test Evidence

The game was tested under different scenarios to ensure it functions correctly. Here are some examples of how it was assessed:

1. I solved the riddles correctly to avoid traps and keep health unchanged, which worked as expected. In another round of testing, where I did not solve the riddle, a description of the encountered trap was displayed, and health was reduced as the code was designed to do.
2. Other tests took place to see what would happen if the player attempted to move outside the grid. Initially, this did not work as expected, after modifying the code to keep the movement inside the grid. After fixing this, I tested it again, and when the player tried to go outside of the grid, the game displayed an error.
3. After adding a limit on the number of moves, I assessed to see if the game would end after 15 moves. Initially, it did not, and the move count went into negative numbers. After correcting this, the game worked as expected.

```
Currently on the Map:
  0 1 2 3 4 5 6
0 X . . . . .
1 . . ? ? . . .
2 ? . . . ? ?
3 . ? . . . ?
4 ? . . ? . . ?
5 . . . . ? ? ?
6 ? ? . ? ? ? ?

Your health is 100%.
Moves left: 0
Choose a direction to move (up, down, left, right): right
Sorry, ash! You've run out of moves! Game Over!

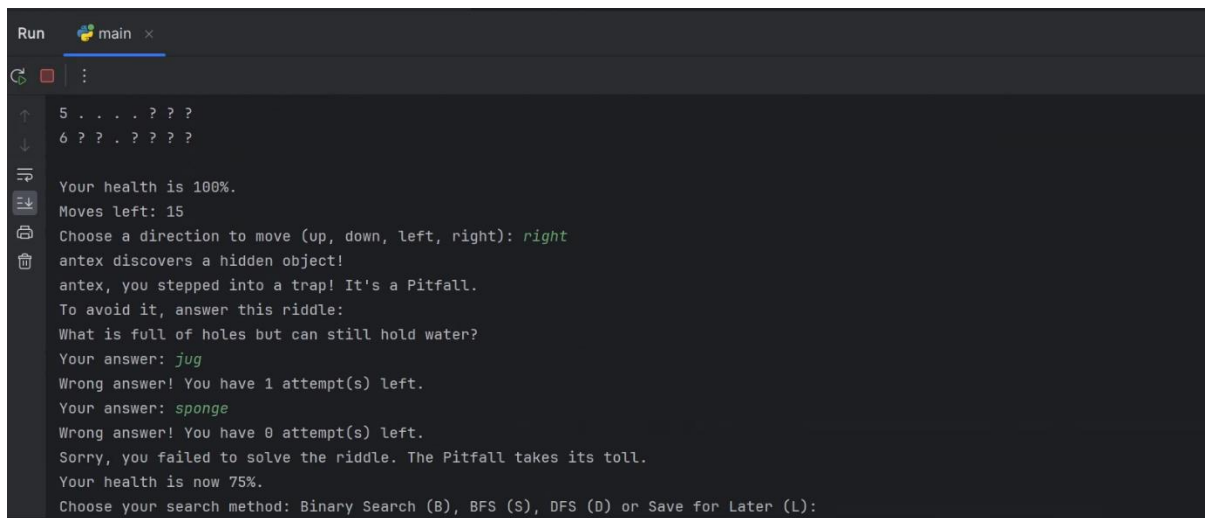
Process finished with exit code 0
|
```

```

ash's turn.
Currently on the Map:
  0 1 2 3 4 5 6
0 X . . . . .
1 . . ? ? . .
2 ? . . . ? ?
3 . ? . . . ?
4 ? . . ? . .
5 . . . ? ? ?
6 ? ? . ? ? ?

Your health is 100%.
Moves left: 14
Choose a direction to move (up, down, left, right): left
Error: You cannot move outside the grid! Please choose a valid direction.
Choose your search method: Binary Search (B), BFS (S), DFS (D) or Save for Later (L):

```



```

Run  main x
5 . . . ? ? ?
6 ? ? . ? ? ?

Your health is 100%.
Moves left: 15
Choose a direction to move (up, down, left, right): right
antex discovers a hidden object!
antex, you stepped into a trap! It's a Pitfall.
To avoid it, answer this riddle:
What is full of holes but can still hold water?
Your answer: jug
Wrong answer! You have 1 attempt(s) left.
Your answer: sponge
Wrong answer! You have 0 attempt(s) left.
Sorry, you failed to solve the riddle. The Pitfall takes its toll.
Your health is now 75%.
Choose your search method: Binary Search (B), BFS (S), DFS (D) or Save for Later (L):

```

Conclusions

This Treasure Hunt Game is designed to be both challenging and engaging, as it evaluates the player's ability to solve riddles, avoid traps, and win the game. By incorporating search algorithms such as Binary Search, Breadth-First Search, and Depth-First Search, to provide crucial clues that can save players valuable time in finding the treasure, this game offers an enjoyable experience. Additionally, this game gives players the option to play with a friend or alone, making it versatile.

Areas where I could improve would be to make sure the riddles are not repeated and in adding a variety of traps and power-ups to make the game even more engaging. Adding more rounds would also be an enhancing feature to add to this game.

References

Khan Academy. (2024, December 11). *The breadth-first search algorithm*.
<https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/the-breadth-first-search-algorithm>

Bari A. (2018, February 24). *5.1 Graph Traversals - BFS & DFS -Breadth First Search and Depth First Search* [Video]. YouTube. <https://www.youtube.com/watch?v=pcKY4hjDrxk>

OpenAI. (2023). *ChatGPT* (May 24 version) [Large language model].
<https://chat.openai.com/>

This assignment used generative AI in the following ways for the purposes of completing the assignment: finding errors, research, and planning.