Magical Map - Pathfinding Simulation

Introduction

This project simulates a dynamic grid-based map in which a magical agent attempts to reach several objectives. The map reveals hidden obstacles as the agent moves, requiring adaptive pathfinding strategies. The agent uses Dijkstra's algorithm to find the shortest path to its goals and selects the best options when interacting with wizards.

Pathfinding with Dijkstra's Algorithm

Dijkstra's algorithm is used to determine the most cost-effective path from the agent's current position to the next objective. The algorithm dynamically recalculates paths when new obstacles are discovered, ensuring the agent always has the optimal route.

Dijkstra Class Analysis

- Node[][] grid stores the map layout.
- startX and startY define the current starting position.
- findShortestPath(targetX, targetY) computes the shortest path using a priority queue.
- distances[][] keeps track of the shortest known distance to each node.
- preData[][] is used to reconstruct the final path after reaching the target.
- reconstructPath() rebuilds the path from the target to the start.
- getDistance() calculates edge costs based on direction-specific weights.

Dynamic Recalculation and Wizard Choices

When obstacles become visible, the agent stops and recalculates the path. If a wizard presents multiple transformation options, the agent simulates each option on a temporary grid. It then chooses the one leading to the shortest overall path to the next objective. This decision-making process is managed by selectBestOption(). The selected grid transformation is then applied permanently.

Sample Movement Log Output

Moving to 1-0

Moving to 1-1

Magical Map - Pathfinding Simulation

New obstacle found at 2-1! Recalculating path...

Moving to 1-2

Objective 1 reached!

Number 3 is chosen!

Conclusion

Dijkstra's algorithm is a central component of this project, providing a robust solution for real-time navigation in a dynamic environment. Its integration with decision-making logic and adaptive recalculations makes the system flexible and efficient. This framework can be adapted for use in robotics, game AI, and navigation software.