AKE Maze Game - Final Report

Class: Introduction to Artificial Intelligence

GROUP MEMBERS:

Aleksandra Ristic - A22005854

Ezekiel Nwokolo - E21000674

Kevin Sithole - K22000707

November 30, 2023

Abstract

This project presents a maze game that combines classic gameplay with dynamic elements, introducing variability and challenges in each playthrough. The core feature of the game revolves around the implementation of Dijkstra's algorithm, a graph-based pathfinding technique, to find the optimal path from the starting point to the goal state. Unlike traditional maze games, our creation boasts a dynamic environment that changes the pattern for each gameplay, and obstacles, in the form of enemies, dynamically change their positions and configurations with every game session.

The game environment is designed to challenge agent’s strategic thinking and adaptability as they navigate through the maze. Dijkstra's algorithm ensures that the agent can efficiently explore the maze and find the shortest path, overcoming obstacles strategically placed by adapting to the dynamic nature of the game. This implementation not only adds an element of unpredictability but also shows the ability to enhance problem-solving skills and real-time decision-making.

Key features of the game include a user-friendly interface, engaging graphics, buttons, and a seamless integration of Dijkstra's algorithm to create a challenging gaming experience.

In conclusion, this maze game stands as a testament to the synergy between classic gameplay and cutting-edge algorithms, offering a unique and dynamic gaming experience.

November 30, 2023

Introduction

Embarking on an innovative journey within the gaming landscape, the AKE Maze Game redefines the concept of maze challenges through its unique blend of randomness and strategic complexity. At its core, the game introduces a distinctive approach, generating random mazes with dynamically positioned enemies in every playthrough. This dynamic environment serves as the backdrop for an engaging quest where the player's agent strives to find the shortest path to the goal while skillfully navigating around unpredictably placed enemies. What sets the AKE Maze Game apart is its commitment to user accessibility, accomplished through an intelligently designed GUI that facilitates easy replay and game termination.

The game's GUI is thoughtfully crafted, featuring intuitive buttons for enhanced user experience. The "Play Again" button facilitates the easy reset of the game, generating a fresh maze layout and enemy configuration for a renewed challenge. Conversely, the "End Game" button provides a seamless exit, putting the player in control of their gaming experience. This strategic incorporation of user-friendly controls aligns with the AKE Maze Game's vision of delivering a gaming experience that is not only intellectually stimulating but also easily accessible to a broad audience.

Method

In the AKE Maze Game, Dijkstra's algorithm is used to find the best way for the player to navigate through the maze and reach the goal state. This algorithm is like a smart guide that excels at finding the shortest path between two points. Imagine the maze as a big puzzle, and Dijkstra's algorithm is like a helpful tool to figure out the quickest way from the starting point to the finish.

In simple terms, the maze is like a map, and each spot you could be in is a point on that map. Dijkstra's algorithm looks at all the possible paths and cleverly chooses the one that gets you from where you are to where you want to go in the least amount of steps.

During the game, this algorithm is always working in the background, recalculating the best path as the player moves. It helps the game's character (the agent) move through the maze, avoiding obstacles strategically placed in its way. Dijkstra's algorithm is the best choice for our dynamic maze game because of its ability to adapt to the changing environment, obstacles' position and recalculate the new path for every gameplay so the agent always takes the shortest (optimal) route.

In a nutshell, Dijkstra's algorithm is like the brain of our AKE Maze Game. It ensures that the game is not just fun and challenging but also smart in helping players find their way through the ever-changing maze, adding a touch of excitement and problem-solving to the gaming experience.

Results

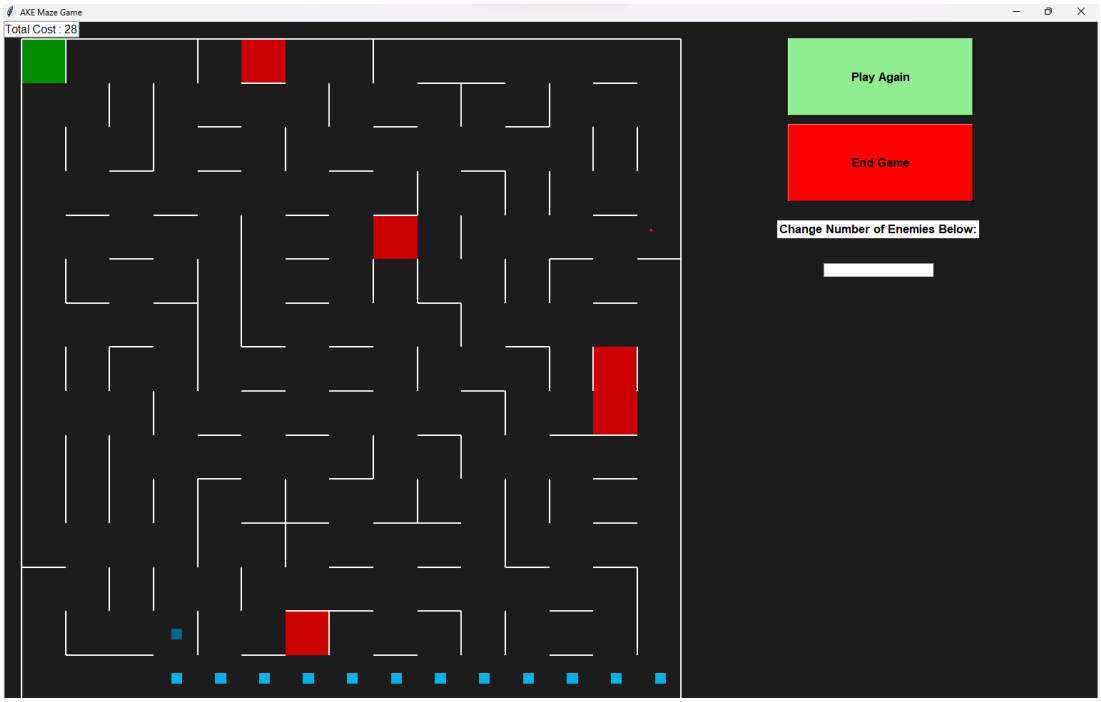
AKE Maze Game has a dynamic front-end environment that integrates intelligent agents, ever-changing obstacles or agent’s enemies, and Dijkstra's algorithm to find the optimal path to the goal state. The game's uniqueness lies in its dynamic nature, as the environment pattern is randomly generated for each gameplay session, including an element of unpredictability and freshness into every encounter.

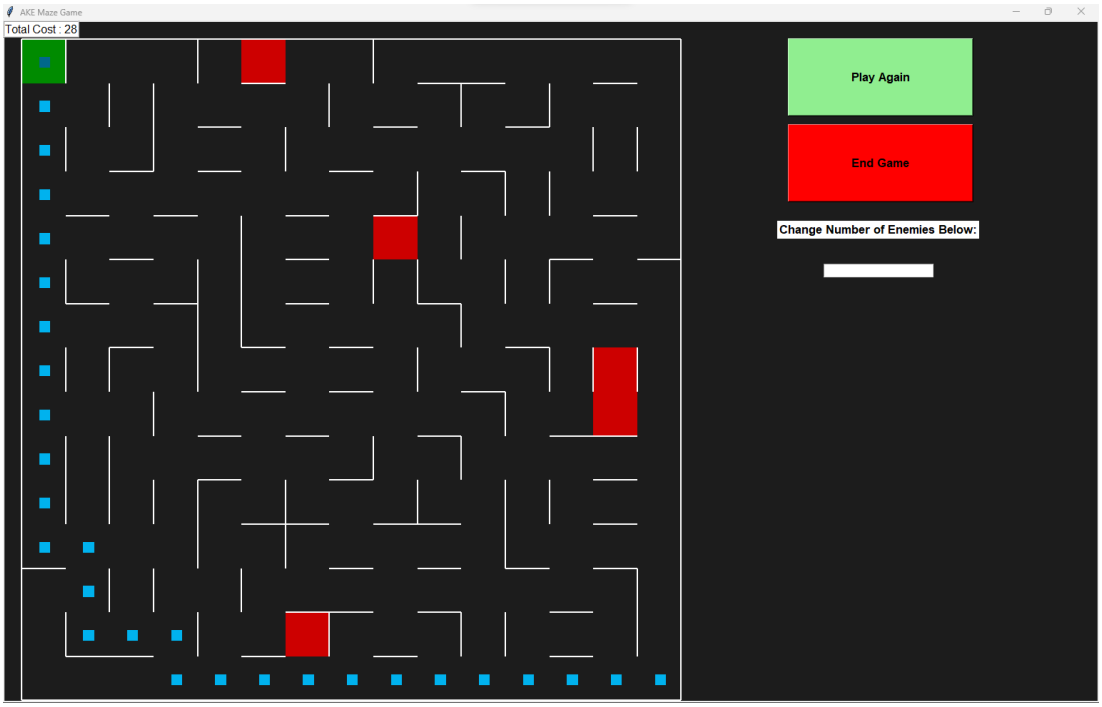
Within this dynamic maze, obstacles in the form of enemies are strategically placed, adding an extra layer of complexity. The Dijkstra's algorithm, a sophisticated pathfinding technique, recalculates the optimal path from the starting point to the goal state for each gameplay iteration. This not only ensures efficient navigation through the maze but also introduces an adaptive challenge, as the agent learns to contend with the ever-shifting landscape of dynamic obstacles.

The user interface is designed for an intuitive experience, featuring two optional buttons: "Play Again" and "End Game." Clicking "Play Again" triggers the creation of a new maze environment, complete with a fresh distribution of enemies and a recalculated path. On the other hand, the "End Game" button provides a seamless exit from the application.

First run (gameplay):

During the first gameplay, we can see the maze game with specific environment pattern which was randomly created by the code, random locations of the obstacles or enemies (marked as red fully-sized squares), and an agent who is the main player of the game (marked as blue small-sized squared). The agent is trying to avoid enemies and going towards the goal state (green square in the maze). In the second picture below is the result of the first gameplay - the agent has successfully reached the goal state and all five of the obstacles are avoided.

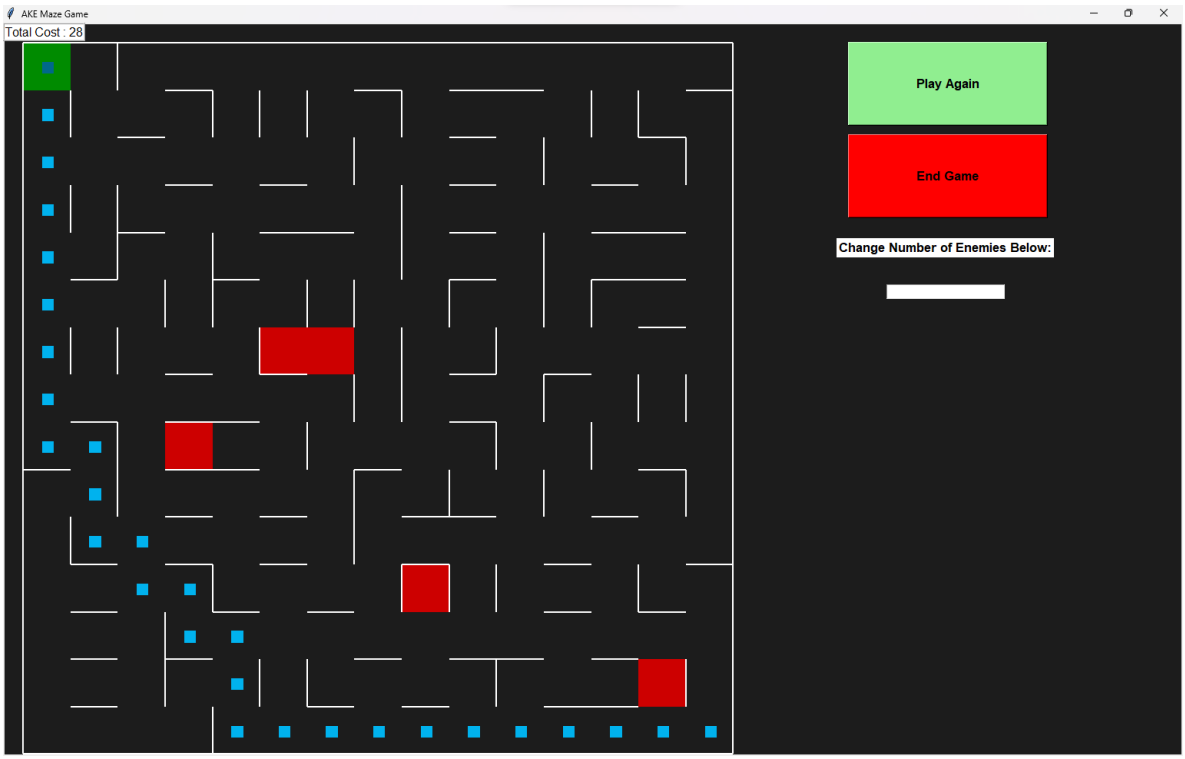




Second run (gameplay):

During the second gameplay, the maze environment has changed. Positions of the obstacles are different and the agent is using a new path to the goal state and avoid all the enemies on his way. The number of enemies can be changed only in the code.

Currently, the number is set to five. One of the ideas was to be able to change the number of obstacles in the front-end application before the next gameplay. Unfortunately, we were not been able to make it work, but we left the text field for our future development and enhancement of the game.



AKE Maze Game Development

We have been using the payamaze GitHub code which we found on the Internet as the reference of our maze game development. The code has already had object classes with all features and characteristics, such as: optional color range for the environment, and agent, agent's shape in form of squares and arrows. The environment was created with static walls, which means that during every gameplay, maze pattern would be the same. Originally, the A* algorithm was used to find the optimal path for the agent's movement towards the goal state and calculate the cost of the action.

Things that have been changed:

- Implementing Dijkstra's algorithm to find the optimal path and recalculate it for every gameplay.

- One of the first things that was changed from the original code was to make the maze environment dynamic. We created the following methods: _Open_East, _Open_West, _Open_North, and _Open_South, each responsible for opening a wall in a maze represented as a 2D grid. These methods take the coordinates (x, y) of a cell in the maze and update the corresponding walls to indicate that they are open. So, for each gameplay, the maze environment will be changed.

- We also added obstacles in the form of other agents which are "enemies" for our main agent. They have the same features, but they are not moving or trying to reach the goal state. Their positions were specified in the code. Also, the number of obstacles can be changed in the code, and there is no minimum or maximum number.

- After we created obstacles, we decided to make them dynamic. We just created a function which randomly position the obstacles in a maze environment. So now, for every gameplay, we would get different maze pattern and different locations of the obstacles.

- GUI application design is also improved by making the maze environment bigger, adding buttons which will trigger the code and play the game, and exit the game and close the application.

One of the problems that we faced during the game development was that whenever we created obstacles dynamic, our agent was not able to avoid them anymore.

Important note: Since our game has a dynamic environment and obstacle locations, it is not always possible to calculate the optimal path for the agent so the code will give us an error. That doesn't mean that something is wrong with the code, it's just the possibility to find the free path (without going through the walls) to the goal state and avoid all the obstacles.

Conclusion

To Sum up, the AKE Maze Game is an original mix of classic gaming elements and innovative dynamics that demonstrates the implementation of Dijkstra’s algorithm for the best possible pathfinding. Unlike static mazes, our game presents a dynamically changing environment where obstacles represented as enemies variably change their positions and arrangements during each gameplay session. As the agent makes its way through the maze, this dynamic landscape challenges the agent's adaptability and strategic thinking. The implementation of Dijkstra's algorithm functions as the game's cognitive framework, guaranteeing that the agent effectively investigates and adjusts to the changing path, underscoring the correlation between classic gameplay and state-of-the-art algorithms.

The AKE Maze Game evolved from a static maze environment to a dynamic and demanding experience over its developmental journey. We made significant changes to the payamaze GitHub code, such as introducing randomly placed obstacles and dynamic maze patterns, which increased the gameplay's complexity. The implementation of Dijkstra's algorithm skillfully tackled obstacles in the constantly shifting maze, demonstrating the game's flexibility and aptitude for solving puzzles. Despite early difficulties, like the agent's inability to avoid dynamic obstacles, algorithmic intelligence and strategic changes led to a sophisticated and refined gaming experience. The game's continuous development highlights its potential for further improvements and advancements in the maze-based gaming industry.