INTRODUCTION

1.1 PROBLEM STATEMENT

Finding the shortest path between two points in a maze can be a challenging task, especially for applications that require real-time decision-making. This project aims to develop a Unity app that implements four pathfinding algorithms (Floodfill, Dijkstra, A*, and Q-Learning) to efficiently navigate complex mazes and find the shortest path between any two points. The goal of the project is to provide users with a tool that can visualize the algorithms in action and allow them to input their own maze configurations. This app has the potential to be used in a wide range of applications, such as game development or robotics.

1.2 MOTIVATION

The motivation behind this project is to address the challenges of pathfinding in complex mazes and to provide a tool that can efficiently solve pathfinding problems in real-time.

For example, game developers may require pathfinding algorithms to create AI-controlled characters that can navigate through complex environments in a realistic and efficient manner. Robotics engineers may require pathfinding algorithms to develop autonomous robots that can navigate through complex indoor or outdoor environments without human intervention.

Moreover, providing a tool that can help users visualize the algorithms in action and compare their performance could improve understanding of how these algorithms work and help users choose the most suitable algorithm for their specific use case.

Also, it was a challenge for the editors to get used with Unity and C# environment.

Overall, the motivation for this project is to provide a tool that can efficiently and effectively solve pathfinding problems in complex mazes, improving the efficiency and realism of applications that require real-time decision-making.

1.3 SCOPE

The scope of this project includes several aspects related to developing a Unity app that implements pathfinding algorithms for mazes. Some aspects of the project's scope include:

1. Implementing multiple pathfinding algorithms: The project could include implementing several pathfinding algorithms, such as Floodfill, Dijkstra, A*, and Q-Learning. These algorithms could be optimized to efficiently solve pathfinding problems for mazes of different sizes and complexities.
2. Designing a user-friendly interface: The app could include a user-friendly interface that allows users to input their own maze configurations, select the pathfinding algorithm to use, and view the pathfinding results in real-time.
3. Maze generation: The project could include generating different types of mazes, ranging from simple to complex, to test the pathfinding algorithms' efficiency and accuracy.
4. Optimizing the algorithms for performance: The pathfinding algorithms could be optimized for performance to improve the app's speed and efficiency in real-time applications.
5. Providing visualizations of the algorithms in action: The app could provide visualizations of the pathfinding algorithms in action to help users understand how each algorithm works and to compare the efficiency of different algorithms.

1.4 GOALS

The goals of the project could be to develop a functional and efficient Unity app that implements multiple pathfinding algorithms for complex mazes. Some potential goals of the project could include:

1. Developing functional pathfinding algorithms: The app should accurately and efficiently solve pathfinding problems for mazes of varying complexities using the implemented algorithms.
2. Optimizing algorithms for performance: The algorithms should be optimized for performance to provide real-time solutions to pathfinding problems and to reduce the app's resource requirements.
3. Creating a user-friendly interface: The app should have a user-friendly interface that is easy to navigate and allows users to input their own maze configurations, select the pathfinding algorithm, and view the results in real-time.
4. Providing visualizations of the algorithms in action: The app should provide visualizations of the pathfinding algorithms in action to help users understand how each algorithm works and to compare the efficiency of different algorithms.
5. Testing and evaluating the app: The app should be tested and evaluated to ensure that it meets the project's requirements and goals, and to identify areas for improvement.
6. Providing visualizations of the algorithms in action: The app could provide visualizations of the pathfinding algorithms in action to help users understand how each algorithm works and to compare the efficiency of different algorithms.