TRIBHUWAN UNIVERSITY

INSTITIUE OF ENGINEERING

PULCHOWK CAMPUS

A PROJECT REPORT ON

DATA STRUCTURE AND ALGORITHM

PATH FINDER

SUBMITTED BY:

Bishal Lamichhane(078BCT035)

Bipin Bashyal(078BCT033)

Ayush K.C.(078BCT025)

Roshan Karki(078BCT098)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PULCHOWK CAMPUS

SUBMITTED ON:

9 MARCH, 2024

ACKNOWLEDGEMENT

First and foremost, we extend our heartfelt thanks to our lecturer Bibha Sthapit, for her unwavering encouragement, insightful feedback, expert guidance, and constant support.

We are indebted to the authors of the various textbooks, online tutorials, and programming forums that served as invaluable resources during the research and implementation stages. Their dedication to sharing knowledge is truly commendable and has been instrumental in expanding my programming skills.

We would like to thank the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering for providing us opportunity of collaborative undertaking which has helped us to implement the knowledge and provide us an experience in teamwork along with knowledge of DSA.

Finally, we would like to express our deep appreciation to our fellow classmates and friends who provided us with their perspectives and insights during the brainstorming and development phases of this project. Their constructive criticism and collaborative spirit helped shape the project into its current form.

Authors:
Bishal Lamichhane
Bipin Bashyal
Ayush K.C.
Roshan Karki Chhetri

ABSTRACT

The project report explores the effectiveness and efficiency of two widely used pathfinding algorithms: A* and Dijkstra. Pathfinding is a fundamental problem in computer science, finding applications in various domains like gaming, robotics, and navigation systems. The A* algorithm and Dijkstra's algorithm are among the most popular solutions to this problem.

The objective of this study is to compare these algorithms in terms of their performance, complexity, and suitability for different scenarios. The A* algorithm incorporates heuristics to guide the search process efficiently towards the goal, while Dijkstra's algorithm explores all possible paths from the source to determine the shortest path.

The report discusses the theoretical aspects of both algorithms, including their principles of operation, time complexity, and space complexity. Furthermore, it provides a detailed explanation of their implementation in the context of pathfinding.

To evaluate the performance of the algorithms, experiments are conducted using various datasets and scenarios. These experiments involve measuring metrics such as runtime, memory usage, and path quality. Additionally, the study explores the impact of different heuristic functions on the performance of the A* algorithm.

The findings of the study provide valuable insights into the strengths and weaknesses of each algorithm. While A* often outperforms Dijkstra's algorithm due to its heuristic-guided search, Dijkstra's algorithm guarantees the shortest path but may be computationally expensive in certain scenarios.

OBJECTIVES

The objectives of the visualization of pathfinding algorithms using A* algorithm and Dijkstra's algorithm are as follows:

1. Comparative and Complexity Analysis: Conduct a thorough comparison between the A* algorithm and Dijkstra's algorithm in terms of their efficiency, effectiveness and analyze the time and space complexities.
2. Performance Evaluation: Evaluate the runtime performance and memory usage of both algorithms under various input sizes and complexities. Measure and compare their execution times to identify their strengths and weaknesses.
3. Implementation: Implement both algorithms in C++ programming language, ensuring correctness and efficiency. Provide detailed explanations of the implementation steps, data structures utilized, and any optimizations employed.
4. Heuristic Function Exploration: Explore different heuristic functions for the A* algorithm and assess their impact on the algorithm's performance. Investigate how different heuristics influence the quality of the generated paths and the search efficiency.
5. Real-world Application: Discuss real-world applications of pathfinding algorithms and how the choice between A* and Dijkstra's algorithm can affect performance and functionality in these applications. They can be used in robotics, gaming, and GPS navigation systems.
6. Case Studies: Present case studies or examples demonstrating the practical usage of both algorithms in solving real-world pathfinding problems. Analyze the results and discuss the suitability of each algorithm for specific use cases.
7. Learning: We acquire teamwork and communication skills along with use of SFML library for the development of this visualization software using C++ programming language.

INTRODUCTION

The field of pathfinding algorithms plays a vital role in various applications, including gaming, robotics, and navigation systems. The ability to find the shortest or most efficient path between two points in a graph is crucial for optimizing processes and enhancing user experiences. In this project, we explore the implementation and visualization of two prominent pathfinding algorithms, namely A* (A-star) algorithm and Dijkstra's algorithm, using the C++ programming language with the SFML (Simple and Fast Multimedia Library) for graphical visualization.

APPLICATION

Pathfinding algorithms have a wide range of applications across various domains. Here are some common applications:

1. **Navigation Systems**: Pathfinding algorithms are extensively used in GPS navigation systems to find the shortest or fastest route between two locations. These systems help users navigate through road networks efficiently, considering factors such as traffic conditions, road closures, and real-time updates.
2. **Robotics**: In robotics, pathfinding algorithms are essential for autonomous robots to plan and execute their movements effectively. Robots can use these algorithms to navigate through complex environments, avoid obstacles, and reach desired locations while optimizing for factors such as energy consumption and safety.
3. **Network Routing**: Pathfinding algorithms are used in network routing protocols to determine the best path for data packets to travel from a source to a destination across a network. These algorithms help optimize data transmission, minimize latency, and ensure efficient use of network resources in communication networks.

4. **Supply Chain Management**: Pathfinding algorithms are applied in supply chain management to optimize logistics and transportation routes. Companies use these algorithms to plan delivery routes, minimize transportation costs, and ensure timely delivery of goods to customers.
5. **Map Routing Services**: Online map routing services, such as Google Maps and Apple Maps, rely on pathfinding algorithms to provide users with optimal driving, walking, or public transit routes between locations. These services consider various factors, such as traffic congestion and road closures, to offer real-time navigation assistance.
6. **Air Traffic Management**: Pathfinding algorithms are employed in air traffic management systems to optimize flight paths and minimize air traffic congestion. These algorithms help air traffic controllers plan efficient routes for aircraft, reduce flight delays, and enhance overall airspace management.
7. **Industrial Automation**: Pathfinding algorithms are employed in industrial automation to optimize the movement of robotic arms and machinery within manufacturing facilities.

LITERATURE REVIEW

C++ is a general-purpose programming language that was developed by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language. It is an intermediate level language, as it comprises confirmation of both high level and low-level language features.

Structure of program

A general C++ program includes the following parts:

1. Standard Libraries Section

2. Class Definition Section
3. Functions Definition Section
4. Main Function Section

Standard Libraries Section: This section is written at the top of the program and consists of the first lines of code read by the compiler. This is also known as the pre-processor section. This section is where all the header files and namespaces are declared in the program such as iostream and std. This includes standard libraries as well as user-defined header files and programs.

Class Definition section: The classes are used to map real world entities into programming. The classes are key building blocks of any object-oriented program. A C++ program may include several class definitions.

Functions Definition Section: Functions are the features of Procedure Oriented Programming but are very useful in OOP as well. In our program there are many functions used to perform various activities as required.

Main Function Section: The main function is the main body of the program which the compiler executes first after all preprocessing. The main function is where all other functions of the program are called from. In C++, we write the main function such as an int return type such that it returns 0 if the program is successfully executed and 1 if there was an error in the execution.

DSA

Data Structures and Algorithms (DSA) form the backbone of computer science and software engineering. They are essential tools for solving complex computational problems efficiently. In this introduction, we'll provide an overview of DSA in the context of

INTRODUCTION

Our project aims to enhance navigation efficiency through the implementation of a pathfinding algorithm visualization. Navigation is a critical aspect of many applications, from robotics to gaming and logistics. The goal is to provide a clear and interactive representation of how a pathfinding algorithm, specifically the A* algorithm, operates in real-time, aiding users in understanding and optimizing routes.

OBJECTIVES

The primary objective of our project is to develop a pathfinding visualization tool that allows users to observe, analyze, and comprehend the decision-making process of the A* algorithm in finding optimal paths. This tool will serve educational, research, and practical purposes, fostering a deeper understanding of pathfinding algorithms among users.

KEY FEATURES

- **Real-time Visualization:** Users can witness the algorithm in action, observing how it explores various paths to reach the destination.

- **Interactive Interface:** The tool will have an intuitive interface allowing users to modify the environment, obstacle positions, and start/end points to observe how the algorithm adapts to different scenarios.

- **Performance Metrics:** The tool will provide performance metrics such as time complexity, nodes expanded, and path cost, enabling users to evaluate the efficiency of the algorithm under different conditions.

IMPLEMENTATION PLAN

- **Development:** We will use C++ programming language and SFML graphics library to build the visualization tool.
- **Testing and Feedback**: Continuous testing will be conducted to ensure the tool's accuracy and user-friendliness. Feedback from users will be collected to improve the tool's performance and features.
- **Documentation and Tutorials**: Comprehensive documentation and tutorials will be provided to assist users in understanding the tool and its applications.

CONCLUSION

Our pathfinding visualization project seeks to revolutionize the understanding and application of navigation algorithms. By providing an interactive and educational tool, we aim to empower users with the knowledge and skills to make informed decisions in their respective fields.