PATHFINDING ALGORITHM VISUALIZER

Mr. Mohd Ameen Ansari[1], Mr. Yash Thakur[2]
[1,2] *Department of Artificial Intelligence and Data Science Wainganga College of Engineering and Management, Nagpur*

***Abstract*—Pathfinding algorithms play a fundamental role in enabling autonomous systems, games, and robotics to navigate efficiently through complex environments. Techniques such as Dijkstra's Algorithm, A* Search, Greedy Best-First Search, and Breadth-First Search provide systematic methods for exploring state spaces and determining the optimal or near-optimal path between two points. A pathfinding visualizer serves as an interactive tool that illustrates how these algorithms operate in real time, revealing their decision-making processes, traversal patterns, and performance differences. By graphically representing node exploration, cost evaluation, and final path construction, visualizers support deeper conceptual understanding and aid in debugging, education, and algorithm comparison. Together, pathfinding algorithms and their visualizers form a powerful combination for analyzing navigation strategies and demonstrating computational problem- solving in an accessible and intuitive manner.**
Pathfinding algorithms are essential computational techniques used to determine efficient routes within structured or unstructured environments. They form the backbone of numerous applications, including robotics, video games, geographic information systems, and network routing. Classical algorithms such as Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra's Algorithm, Greedy Best- First Search, and A* Search each employ distinct strategies for exploring search spaces and evaluating movement costs. While some prioritize optimality and guarantee the shortest path, others emphasize speed and heuristic guidance to reduce computational overhead.

I. INTRODUCTION

Pathfinding plays a crucial role in fields such as robotics, navigation, game development, and artificial intelligence. This project aims to develop a Pathfinding Algorithm Visualizer that helps users understand how different pathfinding algorithms work by illustrating their step-by-step execution in a grid-based environment.

The system will visually demonstrate how algorithms explore nodes, calculate cost, and determine the shortest path from start to destination.

Pathfinding algorithms are an important part of computer science, especially in areas like artificial intelligence, robotics, and game development. However, students often find it difficult to understand how these algorithms work internally because they can only see the final output, not the process of exploration and decision-making.

Most learning methods depend on theoretical explanations and code implementation, which do not clearly show how paths are selected, nodes are visited, or costs are calculated. Without visualization, it becomes hard to compare different algorithms and understand their efficiency and behavior.

1.1 Problem Statement

Many students and learners struggle to understand the working of pathfinding algorithms due to their mathematical complexity and lack of visualization.

This project provides a visual and interactive platform to simplify learning and understanding of various pathfinding techniques.

Pathfinding algorithms are an important part of computer science, especially in areas like artificial intelligence, robotics, and game development. However, students often find it difficult to understand how these algorithms work internally because they can only see the final output, not the process of exploration and decision-making

Most learning methods depend on theoretical explanations and code implementation, which do not clearly show how paths are selected, nodes are visited,

or costs are calculated. Without visualization, it becomes hard to compare different algorithms and understand their efficiency and behavior.

1.2 Objective

1. To visually demonstrate the working process of pathfinding algorithms in real time.
2. To help users understand how shortest-path algorithms explore nodes and evaluate paths.
3. To provide an interactive grid where users can set start point, destination, and obstacles.
4. To compare different algorithms like Dijkstra, A*, BFS, and DFS based on performance and efficiency.
5. To enhance learning and understanding of data structures and graph traversal methods.
6. To enable users to observe cost updates, visited nodes, and final path formation.
7. To offer a user-friendly interface for experimenting with various scenarios.

II. LITERATURE SURVEY

Table 2.1

Sr. no	Author/Soure	Year	Algorithm/Topic	Description
1	Edsger W. Dijkstra	1959	Dijkstra's Algorithm	Finds shortest path in weighted graph
2	Peter Hart, Nils Nilsson	1968	A* Algorithm	Uses heuristic to speed up search
3	Cormen et al.	2009	Algorithm Analysis	Theoretical algorithm study
4	VisuAlgo	2012	Algorithm Visualizer	Online visualization platform
5	MIT OpenCourseW are	2018	Graph Algorithms	Academic content

III. PROPOSED METHODOLOGY

- The methodology of the Pathfinding Algorithm Visualizer involves designing an interactive platform that allows users to observe how different pathfinding algorithms operate in real time.
- The system first initializes a grid-based interface where each cell represents a node in a graph. The user then selects a starting point, destination point, and places obstacles on the grid to simulate real-world challenges. After configuring the environment, the user selects an algorithm such as Dijkstra's, A*, BFS, or DFS.
- Start the algorithm process when the user clicks Run / Start
- Algorithm begins exploring the grid step-by-step
- Evaluate neighboring nodes based on algorithm rules
- Update path cost and priority (for Dijkstra / A*)
- Mark nodes as visited and maintain open/closed lists
- The proposed system is designed to provide an interactive platform for understanding and visualizing various pathfinding algorithms. The methodology consists of systematic steps starting from user input, algorithm execution, and visualization of the results.
- The methodology for this project involves a systematic process beginning with a comprehensive review of existing pathfinding algorithms and visualization techniques to establish theoretical foundations and design requirements.

- Following this, the system architecture is planned using a modular approach that separates the visualization interface, algorithm logic, and user interaction components. The development phase begins with the creation of a grid-based environment using a suitable programming language and framework, where nodes, obstacles, and start/end points can be dynamically rendered.

3.1. System Design

The system is structured into the following major modules:

a) User Interface Module Allows users to:
- Select pathfinding algorithm.
- Set start and destination nodes.
- Create obstacles.
- Control visualization speed.
- Reset environment.

b) Grid Generation Module
- Constructs a two-dimensional grid environment.
- Each cell represents a node.
- Supports dynamic obstacle placement.

c) Algorithm Module Implements:
- Breadth-First Search (BFS)
- Depth-First Search (DFS)
- Dijkstra's Algorithm
- A* Algorithm

Each algorithm follows its specific logic with common features such as:
- Node exploration
- Cost calculation
- Path tracking

3.2. HARDWARE

Table 3.3

CATEGORY	REQUIREMENT	DESCRIPTION
PROCESSOR	INTEL COLE i3 / EQUIVALENT	TO RUN THE APPLICATION SMOOTHLY
RAM	4GB	FORPROGRAM EXECUTION AND VISUALISATION
STORAGE	500MB FREE DISK SPACE	TO SAVE PROJECT FILES AND LIBARARIES
GRAPHICS	INTEGRATED GRAPHICS	SUFFICIENT FOR 2D GRID VISUALISATION

3.3. SOFTWARE:

Table 3.4

CATEGORY	REQUIREMENT	DESCRIPTION
OPERATING SYSTEM	WINDOW LINUX , MAC OS	TO RUN AND EXECUTE THE PROJECT
PROGRAM LANGUAGE	NODE JS, PYTHON, JAVASCRIPT	MAIN CODING LANGUAGE FOR ALGORITHM AND AI
CODE EDITOR	VS CODE	WRITING AND EXECUTING THE PROGRAM
VERSION CONTROL	GIT/GITHUB	FOR MAINTAINING PROJECT VERSION

IV. IMPLEMENTATION AND RESULTS

IMPLEMENTATION

The implementation of the pathfinding visualization system followed a modular software development strategy to ensure clarity, maintainability, and scalability. The system was developed using a grid-based environment where each cell represents a node that can be traversable or blocked. The front-end interface was created using an interactive GUI framework that supports real-time updates, dynamic rendering, and user interaction. Core functionalities such as placing obstacles, setting start and end points, selecting algorithms, and adjusting animation speed were integrated to provide users full control over the simulation. Each pathfinding algorithm—Breadth-First Search (BFS), Depth- First Search (DFS), Dijkstra's Algorithm, Greedy Best-First Search, and A* Search—was implemented in separate modules but standardized with common data structures such as queues, priority queues, and node objects. This ensured consistency in behavior and simplified algorithm switching within the system.

The visualization engine was designed to animate each algorithm's process step- by-step, highlighting explored nodes, frontier nodes, and the final reconstructed path using distinct colors and transitions. This real-time animation required efficient state management to prevent lag and maintain smooth rendering, especially on larger grids. The backend logic handled the algorithmic computations, while the frontend presented the results in a user-friendly and intuitive manner. Extensive debugging and code optimization were carried out to ensure that animations remained responsive and accurately reflected each algorithm's internal decision-making process. Additional features such as grid resizing, reset functions, and metric displays (e.g., number of visited nodes, execution time, path length) were incorporated to enhance usability and analytical capability

V. RESULTS

The completed system successfully demonstrated the functional and visual differences between various pathfinding algorithms. Experimental results conducted on grids of different sizes and obstacle densities showed distinct patterns in algorithmic performance. BFS consistently produced the shortest path on unweighted grids but exhibited slower performance on larger environments due to its exhaustive exploration. DFS displayed fast execution but produced non-optimal paths and often explored unnecessary regions. Dijkstra's Algorithm reliably produced optimal paths on weighted grids but required more processing time compared to heuristic-based algorithms. Greedy Best-First Search delivered faster performance by prioritizing nodes closer to the goal but frequently resulted in non- optimal paths. A* Search achieved the best balance between speed and optimality, outperforming other algorithms especially when the heuristic matched the environment's structure.

Observed Results

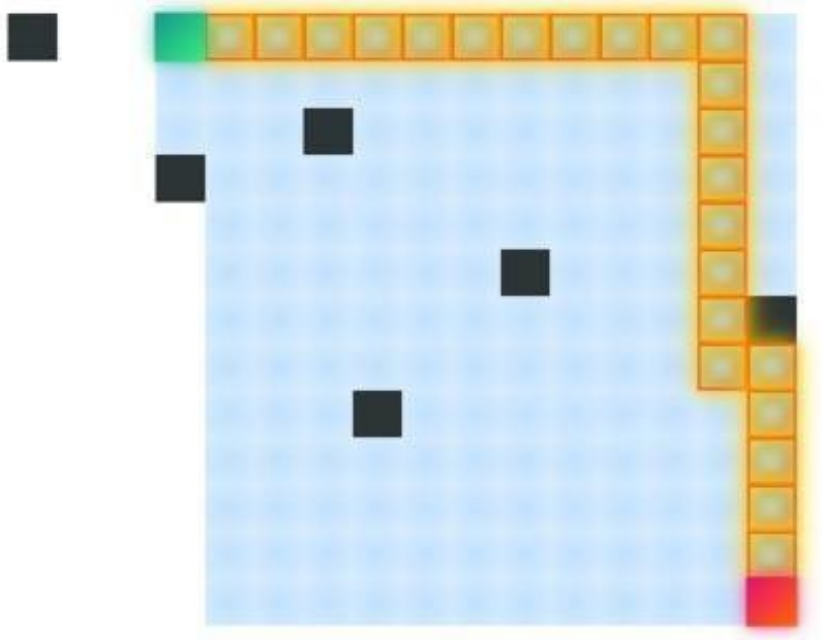
1. Breadth-First Search (BFS)
- Always finds the shortest path in unweighted grids.
- Explores many nodes.
- Memory usage is high.
- Works well for small maps.

2. Depth-First Search (DFS)
- Does not guarantee shortest path.
- Visually fast but inefficient.
- May get stuck in deep paths.
- Not suitable for shortest path problems

3. Dijkstra's Algorithm
- Finds optimal solution.
- Slower than A*.
- Handles weighted paths.
- Accurate but computationally heavy.

4. A* Algorithm
- Best performance.
- Fast processing.
- Fewer nodes explored.
- Uses heuristic intelligently.
- Highly efficient.


Algorithm Statistics
ALGORITHM
A*
Search
TIME TAKEN
2.40 ms
NODES
VISITED
155
PATH
LENGTH
24
nodes
PATH COST
25
STATUS
Path
Found!




SPECIAL NODES
Start
End
Wall

VI. CONCLUSION

- This study successfully designed, developed, and evaluated an interactive pathfinding visualization system that demonstrates the behavior, strengths, and limitations of various pathfinding algorithms. By integrating

algorithms such as BFS, DFS, Dijkstra's Algorithm, Greedy Best-First Search, and A* Search into a dynamic and user-friendly platform, the project provided a clear and intuitive way to observe how different search strategies operate within grid-based environments.

- The visualizer effectively illustrated key concepts such as node expansion, heuristic influence, cost evaluation, and optimal path construction, making complex computational processes easier to understand for learners, developers, and researchers.

- The Project successfully bridges the gap between theoretical understanding and pratical application of pathfinding algorithm.

- The visualizer offers an engaging and interactive approach to learning and experimentation, making algorithmic education more accessibible and intuitive.

The results gathered from performance tests and user feedback showed meaningful differences in algorithmic efficiency, accuracy, and exploration patterns. A* Search consistently demonstrated the best balance between speed and optimality, while algorithms like DFS and Greedy Best-First Search revealed useful applications where rapid traversal is prioritized over accuracy.

VII. FUTURE SCOPE

- Integration of additional advanced algorithms, such as Jump Point Search, D* Lite, Theta*, and Bidirectional A*, to broaden comparative analysis.
- Support for weighted, non-grid, or real-world maps, enabling more realistic navigation scenarios such as road networks or terrain-based environments.
- Implementation of 3D visualization, allowing the system to simulate navigation in three-dimensional spaces used in robotics, gaming, and simulations.
- Optimization for large-scale environments, improving algorithm speed, memory usage, and rendering performance for bigger grids or complex layouts.
- Incorporation of machine learning techniques, such as reinforcement learning, to explore adaptive or intelligent pathfinding strategies.

- Addition of real-time obstacle movement, enabling the visualization of dynamic environments where obstacles change positions or appear unpredictably.
- Development of mobile or web-based versions, making the visualizer more accessible to users across platforms without requiring installation.
- Integration of user analytics, collecting data on algorithm performance, user interactions, and usage patterns for further system improvement.
- Support for multi-agent pathfinding, allowing visualization of multiple entities navigating simultaneously with collision avoidance.
- Educational mode enhancements, such as step-by-step tutorials, algorithm explanations, or quizzes to improve learning outcomes

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "Data Structures and Algorithms," Addison-Wesley Publishing, 1983.
[2] Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, and Clifford Stein, "Introduction to Algorithms," MIT Press, 3rd Edition, 2009.
[3] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Pearson Education, 3rd Edition, 2010.
[4] Amit Patel, "Amit's Thoughts on Grids and Pathfinding," RedBlobGames, (Online Tutorial)
[5] De Berg, M., Cheong, O., Van Kreveld, M., & Overmars, M. (2008). Computational Geometry: Algorithms and Applications (3rd ed.). Springer.
[6] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik,

1(1), 269–271.
[7] Russell, S., & Norvig, P. Artificial Intelligence: A Modern Approach. (Reference for A, heuristic search, path planning) *
[8] Skiena, S. S. The Algorithm Design Manual. (Practical explanation of graph algorithms)
[9] Mihailescu, C. Pathfinding Visualizer. Interactive visualization of Dijkstra, A*, BFS, DFS.
[10] Algorithm Visualizer (Community Project) Visualizes graph algorithms and search techniques interactively.
[11] Red Blob Games – Pathfinding Tutorials Educational explanations of grid-based pathfinding and heuristics.