CoconutFinder Manual

Programming II, MFF UK

Norbert Horváth

Table of Contents

1. Introduction

This application allows you to explore different path-finding algorithms by visualizing them on a board. The application provides four algorithms: DFS, BFS, Dijkstra's, and A*. Additionally, you can generate a random maze and reset the board using the provided buttons.

2. Getting Started

Before using the application, please ensure that your system meets the following requirements: -

.NET Framework: Version 7.0 or above

Installation

Please refer to the provided README.md file in the root directory of this project

Application Layout

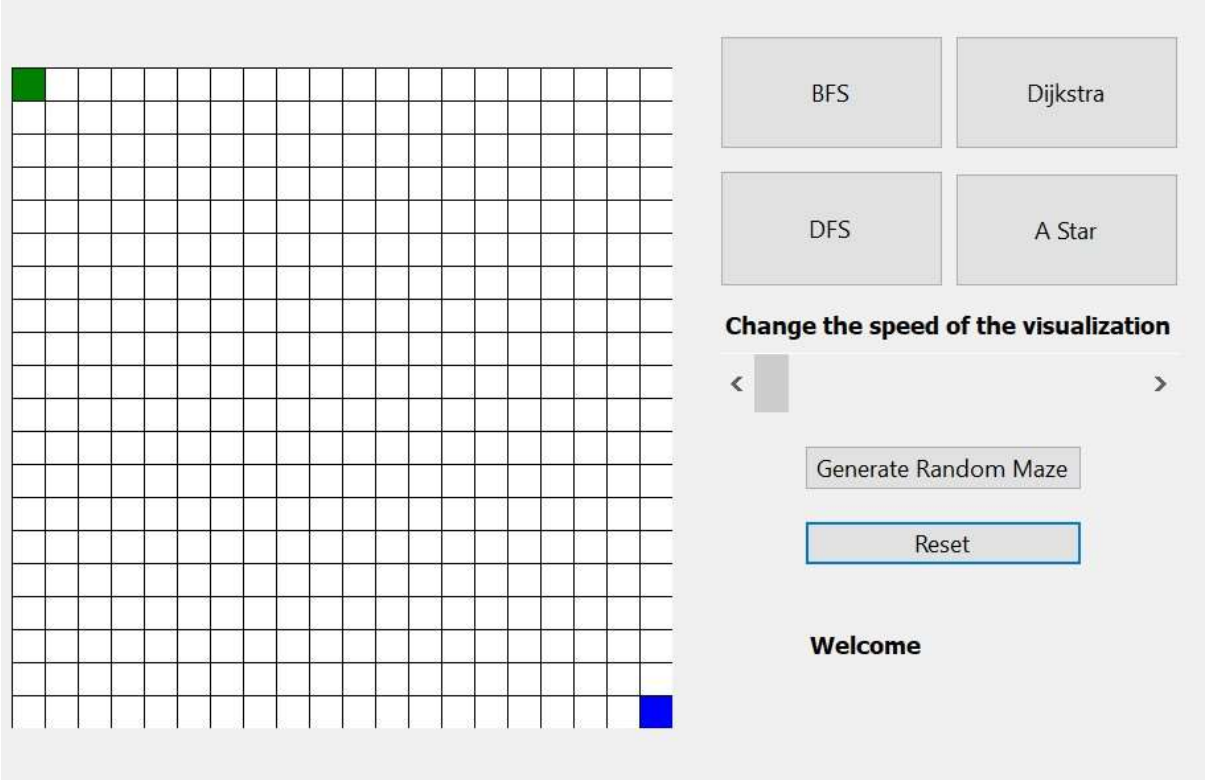


Figure 1.1

Upon launching the application, you will see the main window, shown in Figure 1.1, which consists of the following components:

- Board: A grid representing the path-finding board.

- DFS Button: Executes the Depth-First Search algorithm.

- BFS Button: Executes the Breadth-First Search algorithm.

- Dijkstra Button: Executes Dijkstra's algorithm.

- A* Button: Executes the A* algorithm.

- Generate Random Maze Button: Generates a random maze on the board.

- Reset Button: Resets the board to its initial state.

- Slider: Controls the speed of the animation

Figure 1.2

In Figure 1.2 we can see the bottom part of the application. In the right side we can add nodes using the left-click of the mouse. We can see here the following parts:

- From textbox: The user can choose the beginning of the edge
- To textbox: The user can choose the ending node of the edge
- Weight textbox: The user can choose the weight of an edge
- Starting node and ending node textboxes: These can be used to determine where should we compute the path finding algorithms
- Add edge button: Adds an edge after all the credentials are filled in
- Reset button: Resets the board.

3. Using the Application

The Board

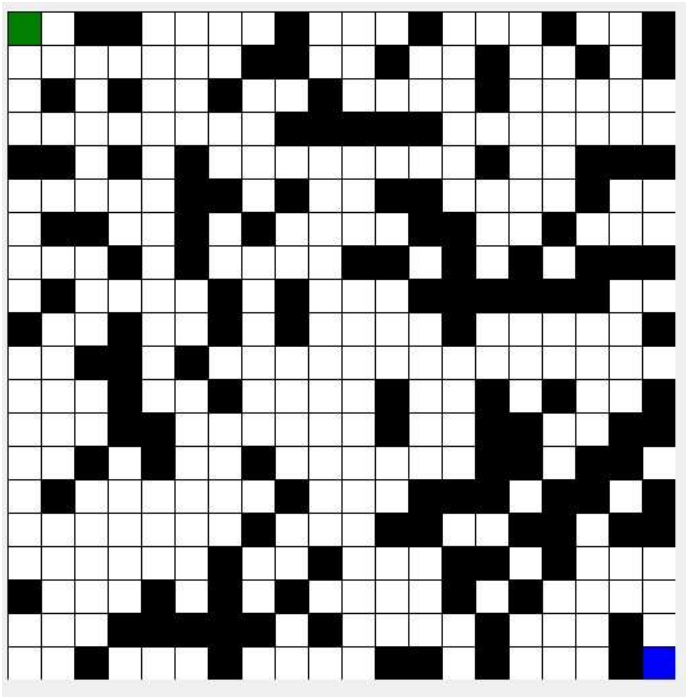


Figure 2

The board represents the environment in which the path-finding algorithms operate. The layout of the board can be seen in Figure 2. Each cell on the board can be in one of the following states:

- Empty: Represented by an empty cell, in this board these are the white cells.
- Start: The starting point of the path. Represented by a green square.
- Target: The target point or destination of the path. Represented by a blue square.
- Obstacle: Impassable areas or obstacles. Represented by a black square.

The Board during animation

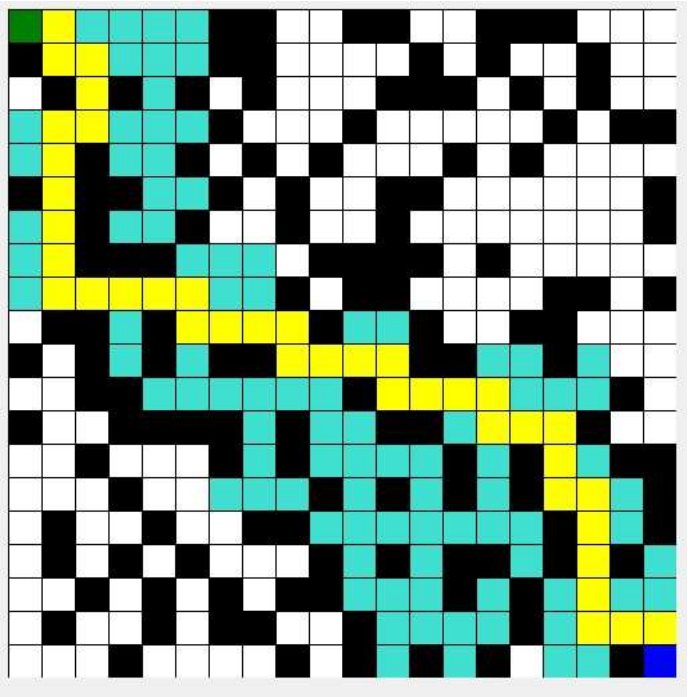


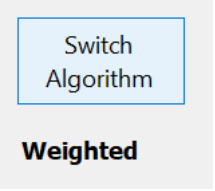
Figure 3

After the chosen algorithm finished, we can see the path from the start node (green) to the end node (blue) as we can see in Figure 3. Here we can see two more colours of squares. The nodes can be in the following two states:

- Path “helper”: The nodes, which were searched trough by the algorithm. Represented by turquoise colour.
- Path: The actual path from the starting node to the finishing node. Represented by yellow colour.

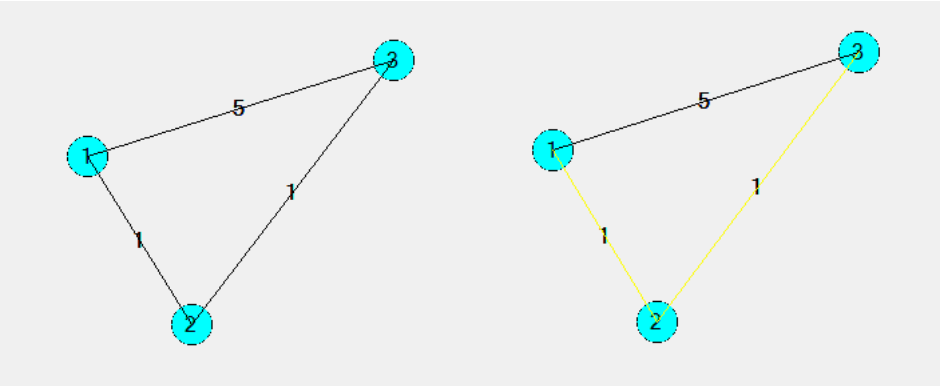
To interact with the board, you can use the **mouse**:

- Left-click on a cell to place an obstacle. - Right-click on a cell to clear it.



Using this button we can switch our algorithms to compute the algorithm for the weighted or for the unweighted graph. The label underneath the button indicates the state of

our algorithm. (Weighted meaning we compute for the weighted graph, Unweighted means we are computing for the unweighted graph, grid)



In the following example we can see the algorithms before and after computing the path-finding algorithms for them. If we want to use the weighted graphs for path-finding visualization, we need to specify the starting and ending node of the algorithm, which I have described earlier.

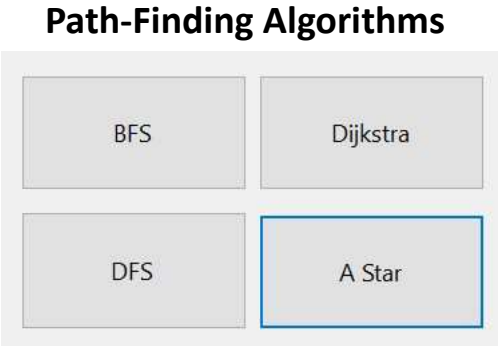


Figure 4

The application currently provides four path-finding algorithms, which can be accessed using the buttons shown in Figure 4:

1. DFS (Depth-First Search): Clicking the DFS button initiates the Depth-First Search algorithm, which explores paths by going as far as possible along each branch before backtracking.
2. BFS (Breadth-First Search): Clicking the BFS button initiates the Breadth-First Search algorithm, which explores all the neighbor cells at the present depth before moving to the next depth level.
3. Dijkstra's Algorithm: Clicking the Dijkstra button initiates Dijkstra's algorithm, which finds the shortest path between the start and target points by considering the cumulative cost of each path.

4. A* Algorithm: Clicking the A* button initiates the A* algorithm, which combines elements of both DFS and Dijkstra's algorithm by considering both the cumulative cost and the estimated distance to the target.

Additional Buttons and sliders

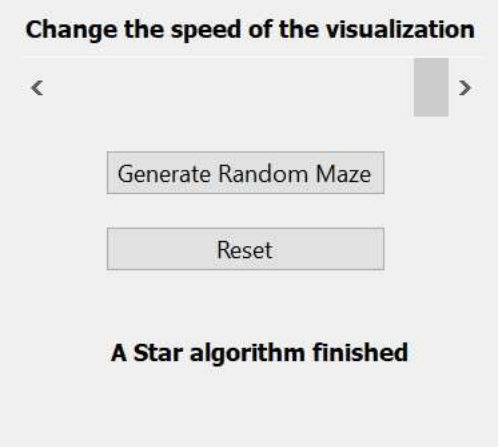


Figure 5

The application also includes two additional buttons and a slider, which can be seen in Figure 5:

- Generate Random Maze Button: Clicking this button generates a random maze on the board. This is useful for testing the path-finding algorithms in different scenarios.
- Reset Button: Clicking the Reset button clears the board and resets it to its initial state, removing any obstacles, the path and path-helpers.
- Visualization speed controller: Controls the speed of the visualization.

4. Conclusion

You have successfully learned how to use the CoconutFinder application.

This project was developed at the Faculty of Mathematics and Physics, Charles University.