Path-finding Algorithms and Solving Mazes

Oliver Temple

September 11, 2021

Contents

1 analysis

1.1 Project Description

Path finding algorithms are essential in many aspects computer science, especially in games and simulations. However, they can be complex things that are hard to visualize, especially when being taught about them. Path finding algorithm visualizers do exist, however, I feel that none of them are perfect, and that they lack features.

Because of this, I am going to make a path finding visualizer that ticks all of the boxes.

1.2 Project Background

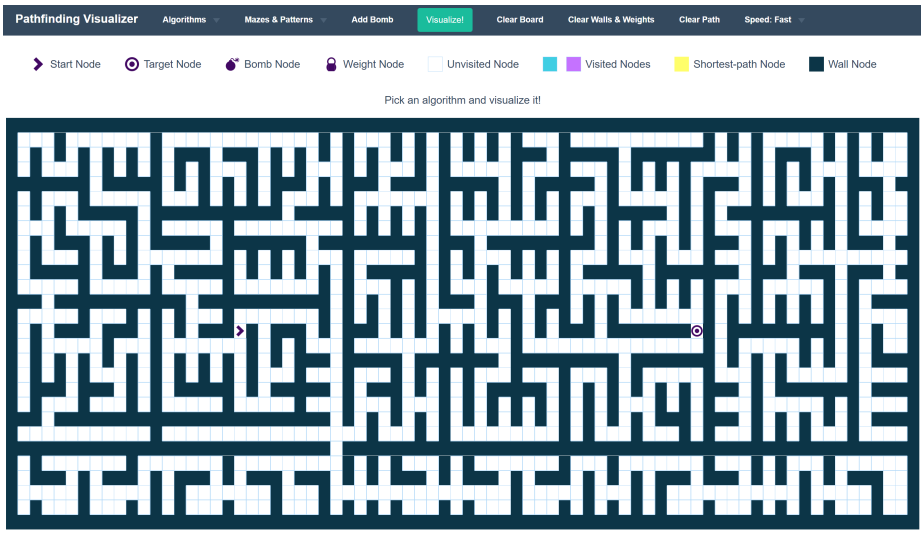
1.2.1 Example 1

https://clementmihailescu.github.io/Pathfinding-Visualizer
In this example, mazes can be generated with various algorithms, as well as being drawn by the user. The mazes can also be edited after they have been generated. The available maze generation algorithms are:

- Recursive Division
- Recursive Division (vertical skew)
- Recursive Division (horizontal skew)
- Basic Random Maze
- Basic Weight Maze
- Simple Stair Pattern

These mazes can then be solved with a number of different path finding algorithms. The available path finding algorithms are:

- Djikstra's Algorithm
- A* Search
- Greedy Best-first Search
- Swarm Algorithm
- Convergent Swarm Algorithm
- Bidirectional Swarm Algorithm
- Breadth-first Search
- Depth-first Search



1.2.1.1 pros

- Many different algorithms to choose from.
- Start and end nodes can be moved.
- Maze can be altered.
- If nodes are moved after visualization has run, then the visualization will update.
- "Bomb" node, adds a via point that the path must go through.

1.2.1.2 cons

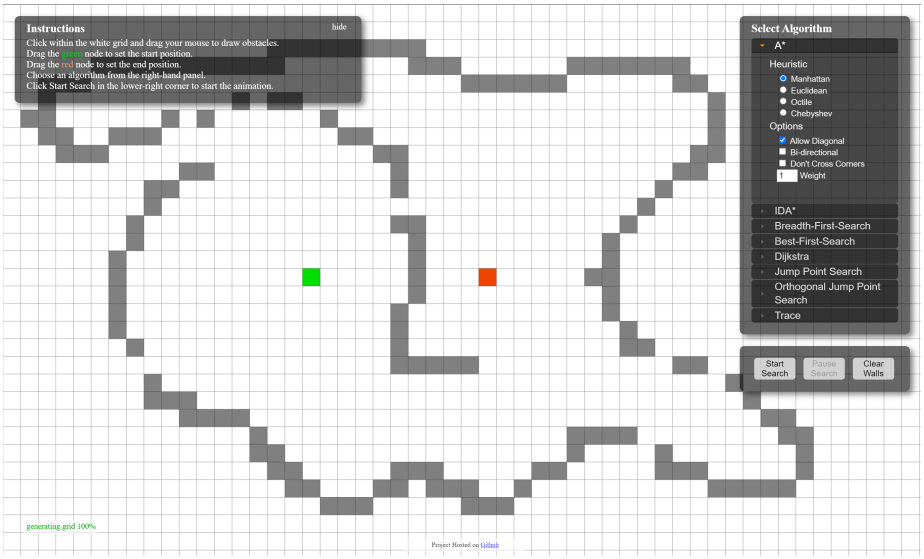
- The visualization is too slow.
- If maze is altered by user after visualization has run, then the visualization will not update.
-

1.2.2 Example 2

https://qiao.github.io/PathFinding.js/visual/
In this example, mazes have to be drawn by the user. The maze can then be solved with a number of different algorithms, however, these algorithms have more choice. For example, in the A* option, you can change the heuristic that is used. The available algorithms are:

- A*

- IDA*
- Breadth-First-Search
- Best-First-Search
- Dijkstra
- Jump Point Search
- Orthogonal Jump Point Search
- Trace



1.2.2.1 pros

- More options to choose from within each algorithm.

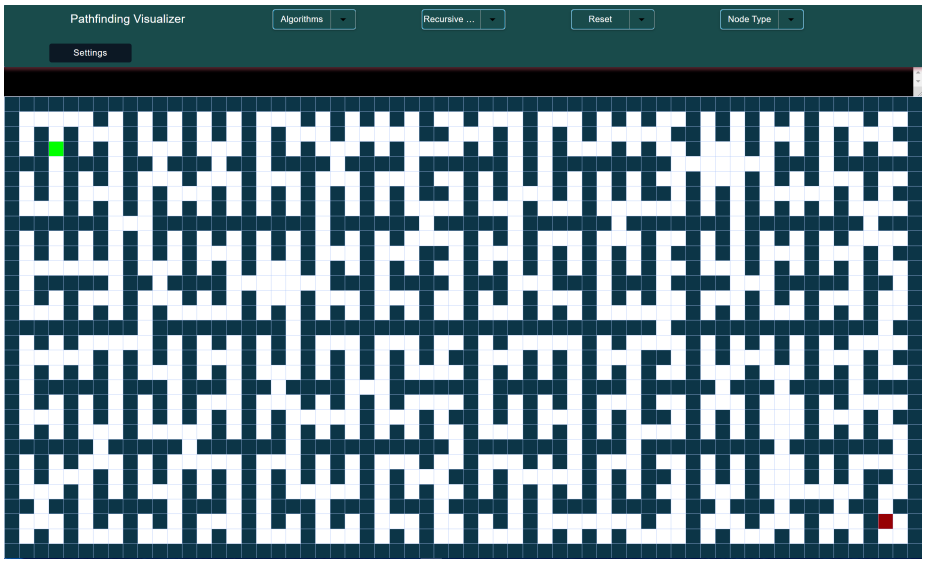
1.2.2.2 cons

- No maze generation.
- Visualization does not update when maze or start/finnish nodes are changed.

1.2.3 Example 3

https://pathfindout.com/
In this example, mazes can be generated or drawn, however, mazes can only be generated with the recursive division algorithm. There are fewer path finding algorithms to solve the mazes than the others. The available algorithms are:

- Dijkstra's Algorithm
- A* Search
- Breadth First Search
- Depth First Search



1.2.3.1 pros

- Different weighted nodes available.
- Shows how many nodes visited.
- Shows final path length.
- Data structure for some algorithms can be changed.
- Weights of specific node types can be changed.
- Node size can be changed.

1.2.3.2 cons

- Sometimes generates mazes that cannot be solved.
- Cannot edit maze after visualization has run.
- Only one maze generation algorithm.
- Fewer path finding algorithms to solve the maze.