Path-finding Algorithms and Solving Mazes

Oliver Temple

May 7, 2021

Contents

1 Introduction

1.1 Motivation

I like programming and path finding algorithms are cool, so are maze generation algorithms.

1.2 Background

There are many path finding algorithms and maze generating algorithms around today. For example, a few of the more popular ones are Dijkstra's algorithm and the A* algorithm for path finding and Kruskal's algorithm and recursive division for maze generation.

1.2.1 A* Algorithm

The A* algorithm is a very popular choice for a path finding algorithm, as it is fairly flexible and can be used in a wide range of contexts. It can be used to find the shortest path between two points.[1]

1.2.2 Dijkstra's Algorithm

In Dijkstra's algorithm, adjacent nodes are checked for the target node, moving outwards until the target node is found. It repeatedly looks at adjacent but not yet visited nodes. It is guarantied to find the shortest path from the starting node to the finishing node, however, it can run slowly for large mazes. [1]

1.2.3 Kruskal's Algorithm

Kruskal's algorithm works by assuming all of the edges on each node are walls. A random node is selected, and if it connects to a node that is not connected to anything, then the wall is removed. [2]

1.2.4 Recursive Division

Recursive division works by taking an initially empty field and bisecting it with a wall, either horizontally or vertically, and adding a single passage in the wall. This step is repeated either side of the wall, continuing recursively until the maze has been created.[3]

2 Objectives

2.1 Main Objectives

1. Research and understand different types of path-finding and maze generating algorithms.
   (a) Focus on a few key algorithms.
      i. Dijkstra's Algorithm.
      ii. A* Algorithm.
      iii. Kruskal's Algorithm.
      iv. Recursive Division
   (b) Continue on to research more algorithms.
      i. Prim's Algorithm.
      ii. Greedy Algorithms.
      iii. Swarm Algorithms.
      iv. Breadth First Search.
      v. Depth First Search.
2. Write the algorithms using pseudocode.

3. Create a web app to visualize the algorithms. [4]
    (a) Use maze object that will store node objects.
    (b) Show the visualization in realtime.
    (c) Allow user to draw their own maze.

2.2 Extensions if time permitting

1. Attempt to create my own path finding algorithm as an extension.
    (a) What makes a good path finding algorithms?
    (b) What makes a path finding algorithm efficient?
    (c) Heuristic vs optimum solution.
    (d) Time complexity of path finding algorithms.
2. Scan mazes in using a camera and solve using path finding algorithms.

3 Methodology

3.1 Reaserch

When researching algorithms I will use websites on the internet, as they allow me to learn about the algorithms easily. For example `http://theory.stanford.edu/~amitp/GameProgramming/` has some very useful information on the A* algorithm and `https://weblog.jamisbuck.org/2011/1/3/maze-generation-kruskal-s-algorithm` has some on Kruskal's algorithm.

To understand the algorithms, I will write them is pseudocode. This will allow me to easily code them in javascript, as each step of each algorithms will be clearly shown.

3.2 Programming

For the programming I will use web based technologies, as i want to make a web app. More specifically, i will use react, a library for javascript, as I will be able to use state management to alter the properties of the nodes easily.

4 Timeline

References

[1] A* Algorithm
`http://theory.stanford.edu/ amitp/GameProgramming/AStarComparison.html`

[2] Kruskal's Algorithm
`https://weblog.jamisbuck.org/2011/1/3/maze-generation-kruskal-s-algorithm`

[3] Recursive Division
`http://weblog.jamisbuck.org/2011/1/12/maze-generation-recursive-division-algorithm`

[4] Example of web visualization.
`https://clementmihailescu.github.io/Pathfinding-Visualizer/`