International Journal of Scientific and Technical Advancements
IJSTA
ISSN: 2454-1532

Maze Generation & Solver

Preeti Dubey, Komal Sarita
Email address: preetidubey2000@yahoo.com,

***Abstract*—**The first half of the paper provides brief introduction to the maze problem, followed by a summary of several maze generation and maze solving algorithms. Later it discusses the algorithm used by the author for automatic and randomized maze generation and its solver.

***Keywords*—** Maze, maze generation, maze generation algorithms, maze solver.

I. INTRODUCTION

A maze is a path or collection of paths, typically from an entrance to a goal through which the solver must find a route. Maze generation is a puzzle tour which has many branched passages. Some branches are closed, and some lead to the end of the tour. The maze solver starts from a starting point and aims to find a valid path between the start and the end point.

Mazes have become very popular as a fun and entertainment tool and also as a very interesting domain from the mathematical point of view. Brain training and entertainment are considered as primary goals of maze application. Mazes have been applied in navigation problems which brought up the need of automatic maze generation.

II. MAZE GENERATION

Maze generation involves designing the layout of passages and walls within a maze. There are many different approaches of generating mazes, with various maze generation algorithms for building them. The two methods used to generate automated mazes are:

i. Wall adding: This method requires setting obstructions (called walls) in an open area.

ii. Passage Carving: This method requires marking the valid path carving passages", one marks out the network of available routes. ii. Some maze generation algorithms (e.g. Prim's algorithm) can be implemented both ways, i.e. as a passage carver as well as a wall adder.

III. IMAZE GENERATION

Automatic maze can be generated by different methods such as graph based methods, recursive division method and other non graph theory based algorithms such as Eller's algorithm and Wilson's algorithm etc. Graph based algorithms are popular for maze generation, since it is a set of vertices and a set of edges. The following constraints have to be taken care off while generating a maze:

i. The maze should not have any loops.

ii. The maze should not have any isolated areas.

iii. There should be exactly one path between any pair of cells.

Graph based Maze generation algorithms:

Some graph based maze generation algorithms are:

i. Depth First Serach (DFS)

ii. Kruskal's Algorithm

iii. Prim's Algorithm

IV. MAZE SOLVING ALGORITHMS/PATH FINDING ALGORITHMS

Maze solving is the act of finding a route through the maze from the start to finish. Different solving approaches can be followed. In some methods of solving maze there no prior knowledge of the maze and in some approaches heuristics can be used. A very typical approach to solve a maze is so called wall following. The two basic approaches used for solving mazes are:

i. Brute force solving: This approach, all paths are traced to check whether the destination has been reached. It is a very fast approach. This approach is not practical for large sized mazes as it is likely to increase the time complexity.

ii. Analytic solving: In this approach, in order to find the solution path the solver is allowed to analyze a part of the maze or the whole maze and rule out a possible dead end. It is helpful in large-sized mazes.

V. IMPLEMENTATION

The automatic maze generation and solver developed by the authors as a part of the miniproject for the completion of the degree of MCA in the department of Computer Science & IT, Central University of Jammu is discussed in this section of the paper. This project has been implemented using the Depth First Algorithm. This project has been developed using VC#.net, GDI+.

Implementation of DFS to generate the maze:

The algorithm used is discusse dg ebneelorawt:e a Maze:

1) Push a random start point onto the stack. Call this start point as initial square 'current square'.

2) Repeat while the stack is not empty:

a) Get list of all unvisited neighbouring squares to the current square

b) If there are neighbours then

i) Choose one of the neighbours at random. Call it 'temp'.

ii) Remove the wall between 'temp' and the current square.

iii) Push the current square into the stack.

iv) Make the current square equals 'temp'.

v) Else if there are no neighbours, pop a square from the stack. Make the current square equal to it.

After executing this algorithm, a 'prefect maze with no' dead end' is generated and has a single solution. The following

Preeti Dubey and Komal Sarita, "Maze generation & solver," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 139-142, 2016.

ISSN: 2454-1532

figure I shows the beginning of the maze and figure II shows a perfect maze generated by implementing the above mentioned algorithm.

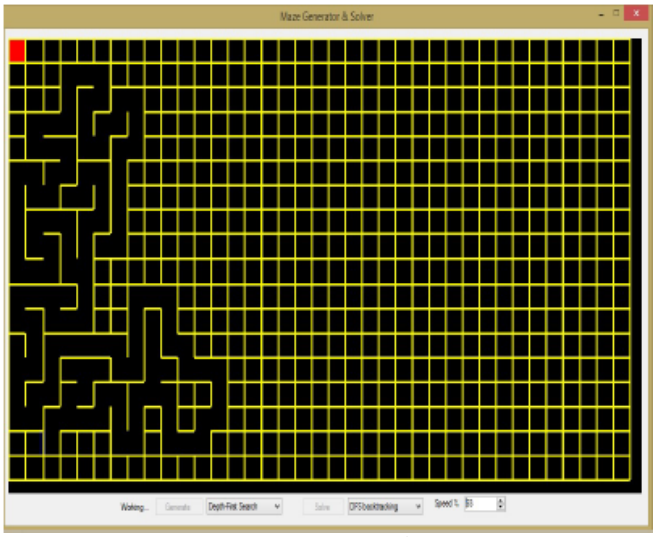


Fig I: below shows the beginning of maze generation

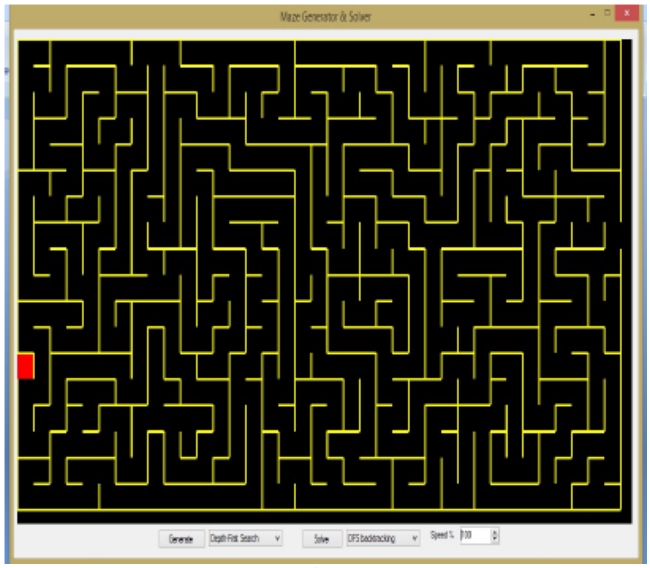


Fig II: below shows an automated perfect maze generated on the click a button

Implementation of DFS to Solve the Maze

DFS with backtracking is used to solve the generated. The following algorithm is used for path finding in our project:

function DFS(Cell start) : Boolean
if start is equal to the maze end
Add start to 'foundPath'
Mark start as visited
Return true
Else if
Start is visited already
Return false
Else
Mark start as visited
For each neighbour of start
If the wall between start and neighbour is knocked
Recursively call DFS function with the neighbour
If the call returns true
Add start to 'found Path'
Return true
If this point is reached, return false

This algorithm finds path to the maze end. When it returns true, it adds the current location to 'found Path', causing all other calls in the stack to return true and add their current locations. The following figure III shoes the beginning of the path finding in the maze.

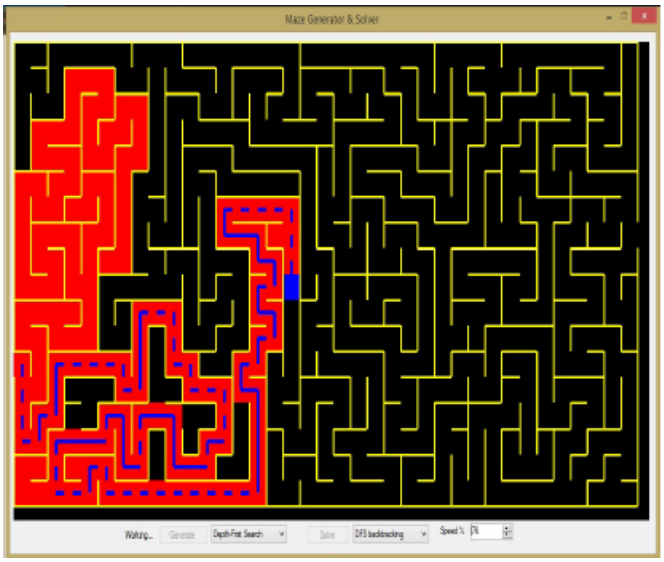


Fig. III: beginning of path finding in maze.

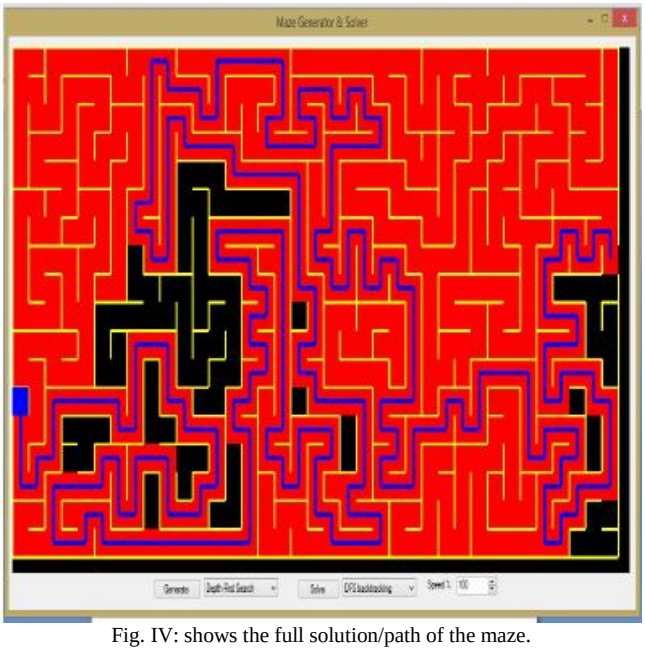


Fig. IV: shows the full solution/path of the maze.

Preeti Dubey and Komal Sarita, “Maze generation & solver,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 139-142, 2016.

International Journal of Scientific and Technical Advancements
ISSN: 2454-1532

VI. CONTROLS

The shape of the maze changes randomly. For this the random function has been used. private Random random = new Random(); The speed of maze generation and path finding can be increased or decreased. The generation and solver of the maze can be controlled by the following controls designed.

Fig. V: Shows the controls.

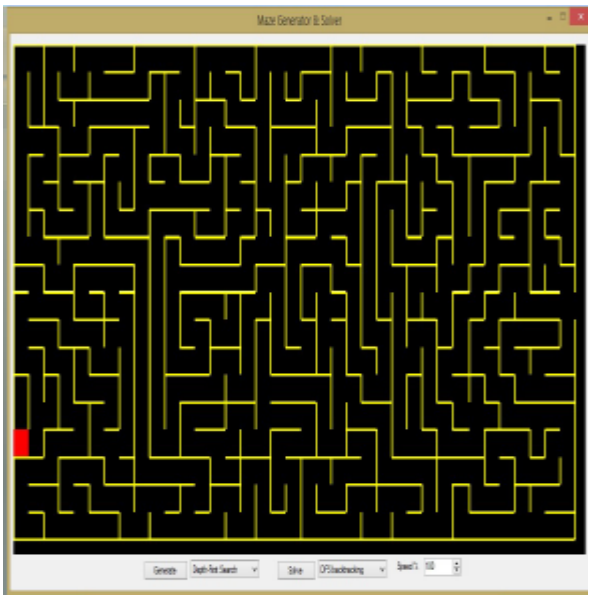
GENERATE: It generates a random maze.
SOLVE: starts finds the path / solution of the maze.
SPEED: It controls the speed of maze generation and solver.

REFERENCES

[1] http://is.muni.cz/th/143508/fi_m/thesis.
[2] www.wikipedia.org
[3] Abbott,R.:LogicMazes,availableatURL
[4] <http://www.logicmazes.com>. 1.1.3
[5] Bouda, O.: Sbeˇr záznamu˚ postupu˚ rˇešení logických úloh, Masarykova univerzita, Fakulta informatiky, 2010. 4.1
[6] Astbury, M.: Mike's Mazes, available at URL
[7] <http://www.mikes-mazes.com> . A.3

APPENDIX

This appendix shows some other patterns of maze generated randomly.



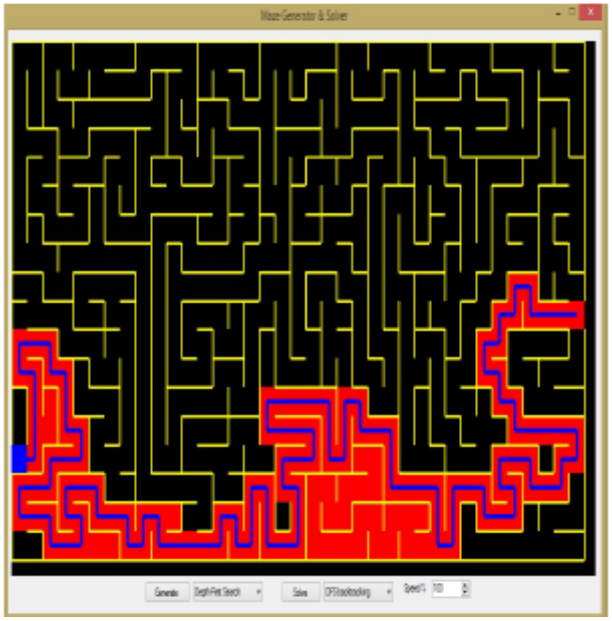


Fig. VI. Random Maze & Its solution.

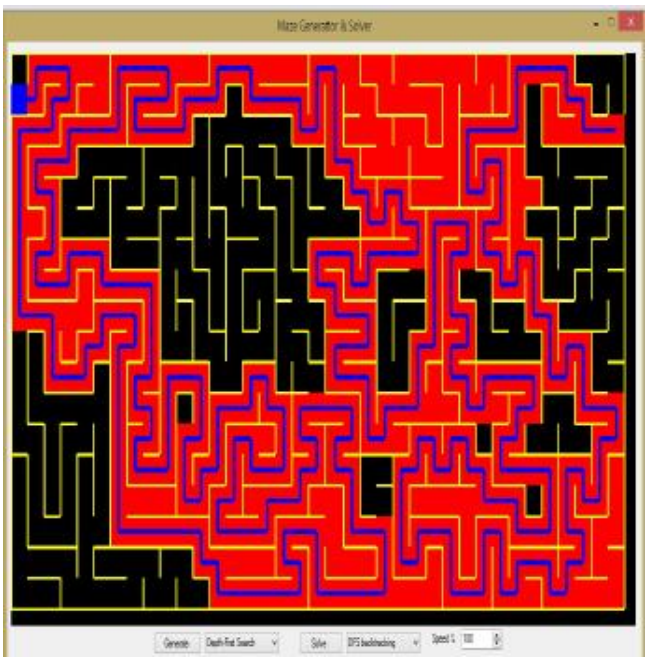


Fig VII: randomly generated maze 2

Preeti Dubey and Komal Sarita, "Maze generation & solver," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 139-142, 2016.

ISSN: 2454-1532

Preeti Dubey and Komal Sarita, “Maze generation & solver,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 139-142, 2016.