

Development of Path Planning Simulation Software for Mobile Robots



Team Based Online Project (TOP) @ Comtel
Dec 2020 to Jan 2021

Team Number 2

Sawant Anand

U.I.E.T, Punjab Univ.

Chandigarh

sawantanand310@gmail.com

Soumya Ghosh

NIT Durgapur

Durgapur

soumya0341@gmail.com

Bharat Dadwaria

JNU Delhi

New Delhi

bkd0385@gmail.com

Rajeevlochana G Chittawadigi

Amrita Vishwa Vidyapeetham

Bengaluru

rg_chittawadigi@blr.amrita.edu

Contents

- Introduction
- Objectives
- Flowchart
- Literature Review-Mazes Algorithms
- Logic of Maze Generation
- Logic of Maze Solution
- Video of Demonstration
- Social Implication
- Conclusion
- References

Introduction

Mobile robots are used for various applications such as:

- Movement of material
- Scanning of rooms or regions, etc.

Before developing a physical prototype of a mobile robot, **it is important to simulate the motion of the robot for any given application.**



Micromouse Competition at
IIT Bombay (TechFest)

Objectives

- **Develop a simulation environment** that can be used to perform **path planning of wheeled mobile robots**, assuming no slippage for simpler implementation.
- **Develop an algorithm for random maze generation** using turtlebot and python coding
- **Develop an algorithm to traverse the generated maze** using turtlebot and python coding using left hand wall following algorithm.

Literature Review-Mazes Algorithms

- Types of Solver based on view:
 - There are two types of solver based on view:
 - The random mouse, wall follower, Pledge, and Trémaux's algorithms are **designed to be used inside the maze by a traveller with no prior knowledge of the maze**
 - The dead-end filling and shortest path algorithms are designed to be used by a **person or computer program that can see the whole maze at once.**

Popular Algorithms:

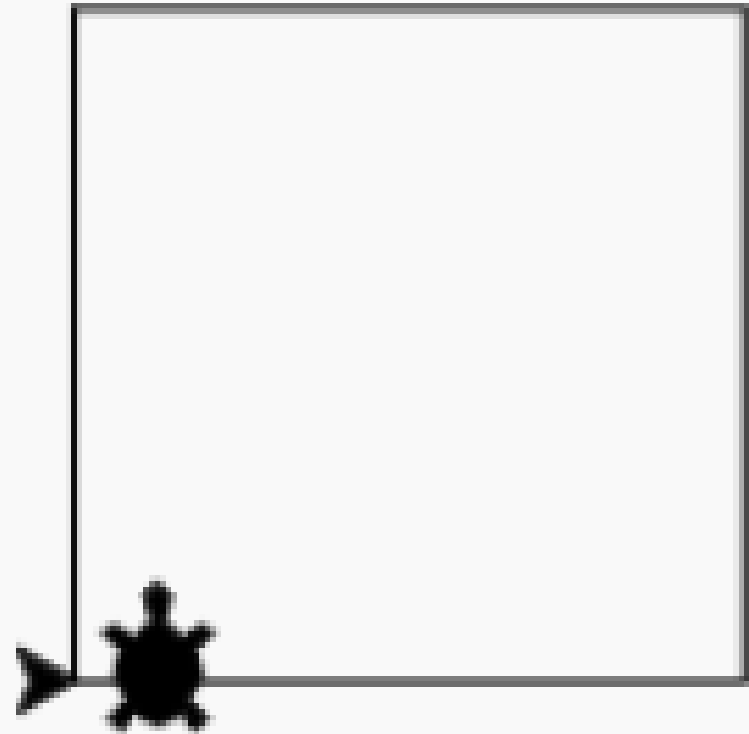
- Wall Follower
- Pledge Algorithm
- Trémaux's Algorithm
- Dead-end filling
- Recursive Algorithm
- Maze-Routing Algorithm
- Shortest path algorithm
- Backtracking Algorithm

Our Implementation

- Python Programming Language
- <https://trinket.io/python/>
 - Online Compiler
- Turtle programming
 - Easy to draw lines and animate motion

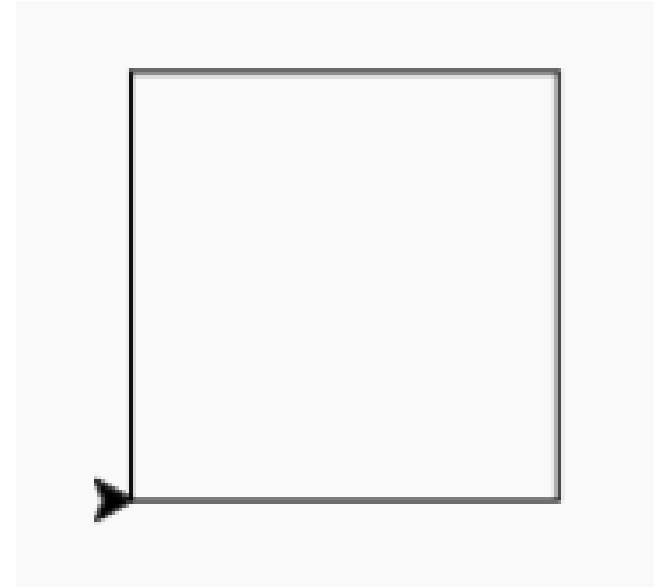
Sample Program

```
1 import turtle
2 size = 8; #even number
3 delta = 16;
4
5 s=turtle.getscreen()
6 myPen=turtle.Turtle()
7 myPen.speed(50)
8 myPen.up()
9 myPen.goto(0,0)
10 myPen.pendown()
11 myPen.shape('turtle')
12 myPen.width(1)
13 myPen.left(90)
14 myPen.forward(delta*size)
15 myPen.right(90)
16 myPen.forward(delta*size)
17 myPen.right(90)
18 myPen.forward(delta*size)
19 myPen.right(90)
20 myPen.forward(delta*size)
21 myPen.right(180)
22 myPen.forward(delta)
23 myPen.left(90)
24
25
```



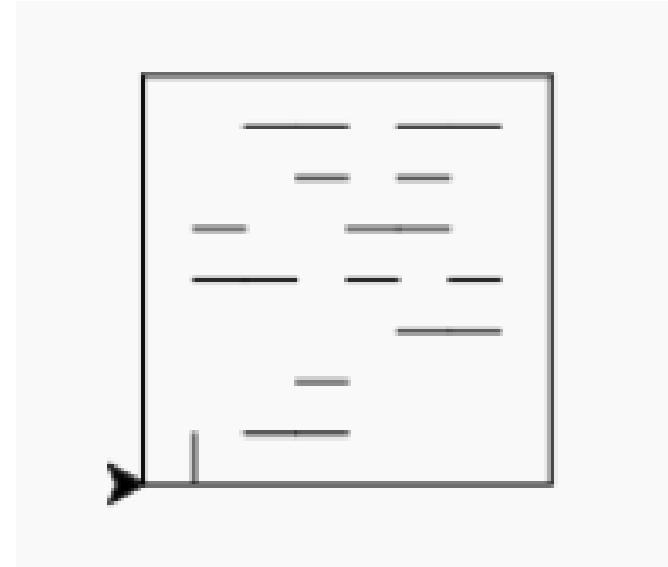
Maze Generation Program

- The maze is generated in 3 Distinct Steps:
- **First, given the size of the maze, walls of the maze using solid lines is created.**
- Second, using left-&-right motion, horizontal inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- **Third, using up-&-down motion, vertical inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)**
- Data of the walls are stored in two 2-D matrices in the binary form, where 1=Space and 0=Wall
- Disadvantages: This Algorithm may create a solvable or unsolvable maze



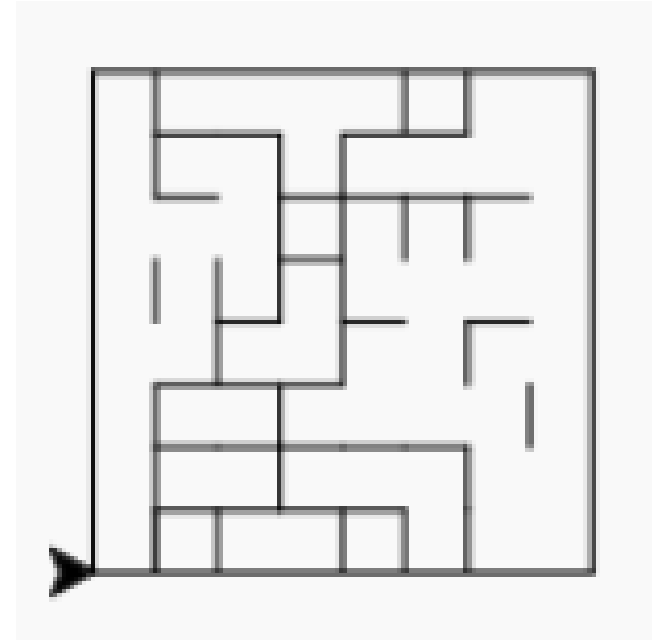
Maze Generation Program

- The maze is generated in 3 Distinct Steps:
- First, given the size of the maze, walls of the maze using solid lines is created.
- **Second, using left-&-right motion, horizontal inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)**
- Third, using up-&-down motion, vertical inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- Data of the walls are stored in two 2-D matrices in the binary form, where 1=Space and 0=Wall
- Disadvantages: This Algorithm may create a solvable or unsolvable maze



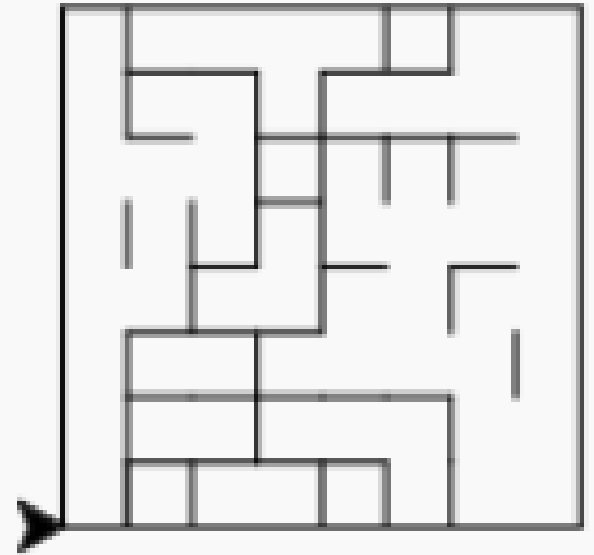
Maze Generation Program

- The maze is generated in 3 Distinct Steps:
- First, given the size of the maze, walls of the maze using solid lines is created.
- Second, using left-&-right motion, horizontal inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- **Third, using up-&-down motion, vertical inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)**
- Data of the walls are stored in two 2-D matrices in the binary form, where 1=Space and 0=Wall
- Disadvantages: This Algorithm may create a solvable or unsolvable maze



Maze Generation Program

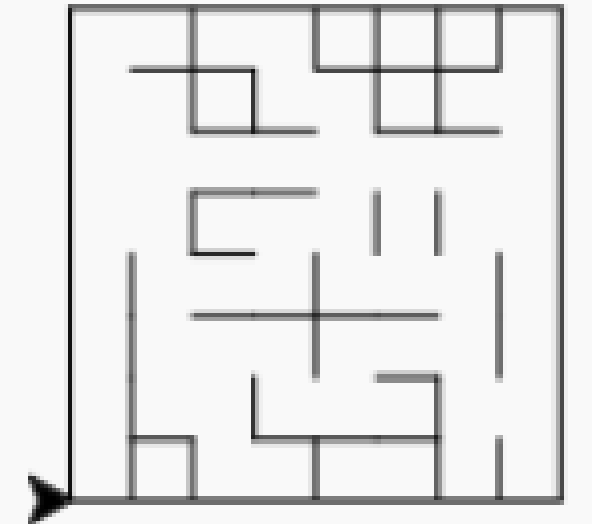
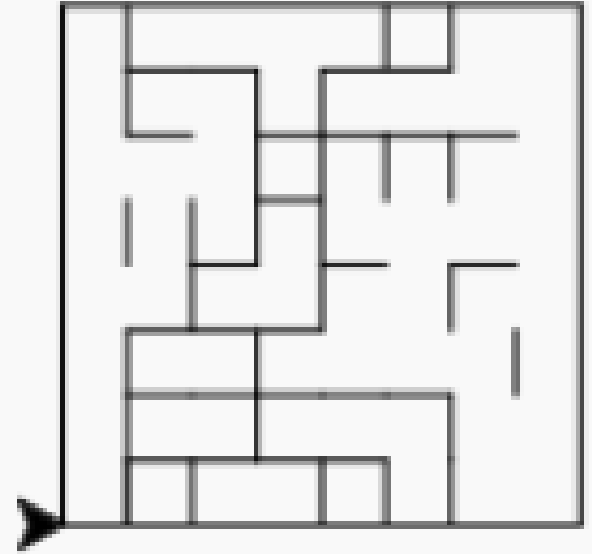
- The maze is generated in 3 Distinct Steps:
- First, given the size of the maze, walls of the maze using solid lines is created.
- Second, using left-&-right motion, horizontal inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- Third, using up-&-down motion, vertical inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- **Data of the walls are stored in two 2-D matrices in the binary form, where 1=Space and 0=Wall**
- Disadvantages: This Algorithm may create a solvable or unsolvable maze



```
1 horizontalEdgeMatrix
2 [[0, 0, 0, 0, 0, 0, 0, 0],
3  [1, 0, 0, 0, 0, 1, 1, 1],
4  [1, 0, 0, 0, 0, 0, 1, 1],
5  [1, 0, 0, 0, 1, 1, 1, 1],
6  [1, 1, 0, 1, 0, 1, 0, 1],
7  [1, 1, 1, 0, 1, 1, 1, 1],
8  [1, 0, 1, 0, 0, 0, 0, 1],
9  [1, 0, 0, 1, 0, 0, 1, 1],
10 [0, 0, 0, 0, 0, 0, 0, 0]]
11
12 (verticalEdgeMatrix)
13 [[0, 0, 0, 0, 0, 0, 0, 0],
14  [0, 0, 0, 1, 0, 1, 0, 0],
15  [0, 1, 1, 0, 0, 1, 1, 1],
16  [1, 0, 0, 1, 0, 0, 0, 1],
17  [0, 1, 1, 0, 0, 0, 0, 1],
18  [0, 1, 1, 1, 1, 0, 1, 0],
19  [0, 0, 1, 0, 1, 0, 1, 0],
20  [1, 1, 0, 1, 1, 1, 1, 1],
21  [0, 0, 0, 0, 0, 0, 0, 0]]
```

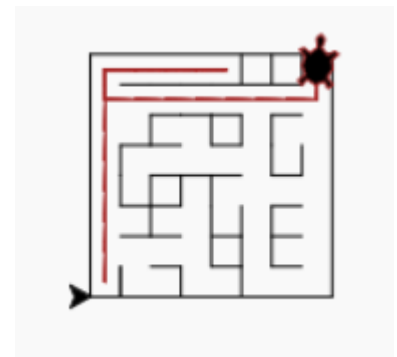
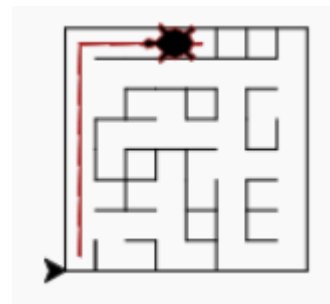
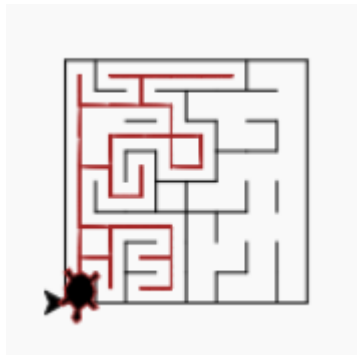
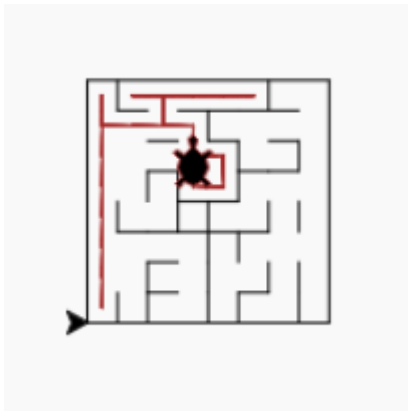
Maze Generation Program

- The maze is generated in 3 Distinct Steps:
- First, given the size of the maze, walls of the maze using solid lines is created.
- Second, using left-&-right motion, horizontal inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- Third, using up-&-down motion, vertical inner walls of the maze are generated, using a random function to generate a cell wall(0) or space(1)
- Data of the walls are stored in two 2-D matrices in the binary form, where 1=Space and 0=Wall
- **Disadvantages: This Algorithm may create a solvable or unsolvable maze**



Maze Solving Logic: Left Wall Follower

- Here, the bot **starts from bottom-left corner** and tries to **reach top-right corner**.
- The maze is viewed as a grid with 4 walls/spaces depending on the orientation of the bot : left, right, front, back.
- The bot checks the data of the current cell-walls and makes the a decision based on the following conditions:
 - It will try to keep a wall on left.
 - If in a cell, there is no left wall, it will turn left and move 1 cell forward
 - Else-If it has front & left wall only, then it will turn right and move 1 cell forward
 - Else-If, it has front, left & right wall, then it will turn around (180°) and move 1 cell forward
 - Else,(only left wall) then it will move 1 cell forward
- Either it will reach the destination cell, or rotate the maze in a loop.



Video of Demonstration

Microsoft Teams

New channel meeting

2021-01-18 17:44 UTC

Recorded by

Mr. Rajeevlochana G.
Chittawadigi

Organized by

Mr. Rajeevlochana G.
Chittawadigi

Channel

D2_6

```

1 import turtle; import math
2 height = 40; width = 10
3
4 s=turtle.getscreen()
5 myPen=turtle.Turtle()
6 myPen.speed(10)
7 myPen.up()
8 myPen.goto(0,0)
9 myPen.pendown()
10 myPen.shape('turtle')
11 myPen.width(1)
12 myPen.left(90)
13 myPen.forward(height)
14 myPen.left(90)
15 myPen.forward(width)
16 myPen.right(180)
17 myPen.forward(2*width)
18 myPen.right(180)
19 myPen.forward(width)
20 myPen.left(90)
21 myPen.forward(height)
22 myPen.left(90)
23
24 myPen.up()
25 myPen.forward(2*width)
26 myPen.down()
27
28 myPen.left(90)
29 myPen.forward(height)
30 myPen.right(180)
31 myPen.forward(0.5*height)
32 myPen.left(90)
33 myPen.forward(2*width)
34 myPen.left(90)
35 myPen.forward(0.5*height)
36 myPen.right(180)
37 myPen.forward(height)
38 myPen.left(90)

```

```

39 myPen.up()
40 myPen.forward(0.5*width)
41 myPen.down()
42 myPen.down()
43
44 theta = 70
45 thetaRadians = math.radians(theta)
46 myPen.left(theta)
47 myPen.forward(height/math.sin(thetaRadians))
48 myPen.right(2*theta)
49 myPen.forward(height/math.sin(thetaRadians))
50 myPen.right(180)
51 myPen.forward(0.5*height/math.sin(thetaRadians))
52 myPen.left(theta)
53 myPen.forward(height/math.tan(thetaRadians))
54 myPen.right(180)
55 myPen.forward(height/math.tan(thetaRadians))
56 myPen.right(theta)
57 myPen.forward(0.5*height/math.sin(thetaRadians))
58 myPen.left(theta)
59
60 myPen.up()
61 myPen.forward(0.5*width)
62 myPen.down()
63
64 theta = 60
65 thetaRadians = math.radians(theta)
66 myPen.left(90)
67 myPen.forward(height)
68 myPen.right(90+theta)
69 myPen.forward(height/math.sin(thetaRadians))
70 myPen.left(90+theta)
71 myPen.forward(height)
72 myPen.right(180)
73 myPen.forward(height)
74 myPen.left(90)

```

```

75 myPen.up()
76 myPen.forward(0.5*width)
77 myPen.forward(0.5*width)
78 myPen.down()
79
80 theta = 45
81 thetaRadians = math.radians(theta)
82 myPen.left(90)
83 myPen.forward(height)
84 myPen.right(180)
85 myPen.forward(0.5*height)
86 myPen.left(90+theta)
87 myPen.forward(0.5*height/math.sin(thetaRadians))
88 myPen.right(180)
89 myPen.forward(0.5*height/math.sin(thetaRadians))
90 myPen.left(2*theta)
91 myPen.forward(0.5*height/math.sin(thetaRadians))
92 myPen.left(theta)
93
94 myPen.up()
95 myPen.forward(width)
96 myPen.down()
97
98 myPen.forward(width)
99 myPen.left(45)
100 myPen.forward(math.sqrt(2)*width)
101 myPen.left(90)
102 myPen.forward(math.sqrt(2)*width)
103 myPen.left(45)
104 myPen.forward(width)
105 myPen.right(45)
106 myPen.forward(math.sqrt(2)*width)
107 myPen.right(90)
108 myPen.forward(math.sqrt(2)*width)
109 myPen.right(45)
110 myPen.forward(2*width)
111

```

THANKS 

Reference

- Books and Websites on Python programming