MINI PROJECT
MAZE SOLVER

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering in

Artificial Intelligence and Data Science

by

Harshita Anala –2

Shruti Devlekar – 9

Akshiti Kachhawa –22

Abhishek Thorat – 61

under the guidance of Supervisor:

Mrs. Sangeeta Oswal

Vivekanand Education Society's

Institute of Technology

(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Artificial Intelligence and Data Science

Vivekanand Education Society's Institute of Technology

2021-2022

Vivekanand Education Society's
Institute of Technology
(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that **Ms.Harshita Anala, Ms. Shruti Devlekar, Ms. Akshiti Kachhawah & Mr. Abhishek Thorat** of Second Year of Artificial Intelligence and Data Science studying under the University of Mumbai have satisfactorily presented the Mini Project entitled **MAZE SOLVER** as a part of the MINI-PROJECT for Semester-III under the guidance of **Mrs. Sangeeta Oswal** in the year 2021-2022.

Date: 18 Dec 2021

(Name and sign)
Head of Department

(Name and sign)
Supervisor/Guide

Vivekanand Education Society's
Institute of Technology
(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Artificial Intelligence and Data Science

DECLARATION

We, **Ms.Harshita Anala, Ms. Shruti Devlekar, Ms. Akshiti Kachhawah & Mr. Abhishek Thorat** from ***D6AD***, declare that this project represents our ideas in our own words without plagiarism and wherever others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our project work.

We declare that we have maintained a minimum 75% attendance, as per the University of Mumbai norms.

We understand that any violation of the above will be cause for disciplinary action by the Institute.

Yours Faithfully

Ms.Harshita Anala

Ms. Shruti Devlekar

Ms. Akshiti Kachhawah

Mr. Abhishek Thorat

Acknowledgement

We are sincerely grateful to our college, VESIT, our respected principal Dr. J. M. Nair ma'am and our HoD Dr. M. Vijayalaksmi ma'am for giving us the opportunity to work on our Mini project i.e. an AI based game development. Completing this project would not have been possible without their guidance and support. We extend our gratitude to our mentor Mrs. Sangeeta Oswal ma'am who guided us in all the meetings with her. She provided us some valuable information and resources to help increase the efficiency of our project. We would also like to thank our peers who helped us whenever we faced any difficulty while developing the game and our parents who always supported us in their own ways. Finally, we thank all the mentioned and not mentioned who were a part of our project.

Table of Contents

Abstract

Mazes are often simple puzzles for humans, but they present a great programming problem that we can solve using shortest-path techniques like Dijkstra's algorithm.

In this project we have discussed a unique general algorithm for exploring and solving any kind of line maze .For the general algorithm, we need a method to map the whole maze, which is required if the maze is complex. The proposed maze mapping system is based on a coordinate system and after mapping the whole maze as a graph in standard 'Adjacency-list representation' method, shortest path and shortest time path was extracted using Dijkstra's algorithm.

1. Introduction

1.1 Introduction:

We have a shortest path finding algorithm between two points using a modified Dijkstra algorithm.

Software used: Python, Tkinter.

Divided into 3 main scripts:

- **First script generates an Adjacency Matrix.**
- **Second script imports the Adjacency matrix and applies Dijkstra Algorithm logic on the generated matrix.**
- **Third script will import the second script and give a GUI to our project.**

1.2 Problem Statement:

There is N*N matrix, let's say M. One has to start from a source cell in the matrix and need to reach the destination cell avoiding the obstacles along the way.

The task is to find the lowest number of cells (or shortest path) to pass through in order to reach the destination.

One can move in any direction to reach the destination.

There is a source point, a destination point, and there are obstacles. Anything else is a free path. Can you do it ?

1.3 Objective:

The Maze Solver project using Python will allow you to select a starting point, select the obstacles and select the destination (flexibility to increase and decrease the complexity), and not only can it display the shortest path between the two points, it can also solve different mazes.

The project also has a GUI , which will capture the User's attention and keep them engaged.

1.4 Future Scope:

This is a Toy-problem designed such that the solutions are broadly applicable :

- This project can be extended to make a Maze Solver Robot, Using Artificial Intelligence with Arduino.
- It can have applications in autonomous home appliances like vacuum cleaners ,etc.
- In intelligent traffic control that helps ambulances, fire fighters, or rescuing robots to find their shortest path to their destination.
- In driverless cars and disaster management work using robots.
- Maze solving can be further extended for autonomous navigation in an occupancy grid to get to the nearest EV charging station.

2. Literature Survey:

1.1 Literature/Techniques studied:

PYTHON PROGRAMMING LANGUAGE

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

1. web development (server-side),
2. software development,
3. mathematics,
4. system scripting.

Python can be used alongside software to create workflows.

Python can connect to database systems. It can also read and modify files.

Python can be used to handle big data and perform complex mathematics.

Python can be treated in a procedural way, an object-oriented way or a functional way.

Python was designed for readability, and has some similarities to the English language with influence from mathematics.

Python Comments : Comments can be used to explain Python code.

Comments starts with a #, and Python will ignore them

#This is a comment

Python Variables : Variables are containers for storing data values.

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

```
x = 5
y = "John"
```

Casting

If you want to specify the data type of a variable, this can be done with casting.

Example

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
```

Global Variables : Variables that are created outside of a function are known as global variables.

To create a global variable inside a function, you can use the global keyword.

Example

```
def myfunc():
  global x


    print("a is greater than b")
```

Nested If

You can have if statements inside if statements, this is called nested if statements.

Example

```
x = 41
if x > 10:
  print("Above ten,")
if x > 20:
 print("and also above 20!")
else:
 print("but not above 20.")
```

Python Lists : Lists are used to store multiple items in a single variable.

Lists are created using square brackets:

Example

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-oriented programming languages.

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  print(x)
```

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):

  print(x)
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop".

Example:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

Example:

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Python Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

In Python a function is defined using the def keyword:

To call a function, use the function name followed by parenthesis:

Example

```
def my_function():
  print("Hello from a function")
my_function()
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

Example

```
def my_function(fname):
  print(fname + " Refsnes")
```

```
my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Return Values

To let a function return a value, use the return statement:

Example

```
def my_function(x):
  return 5 * x

print(my_function(3))
```

Recursion

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

Example

Recursion Example

```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result)
  else:
    result = 0
  return result

print("\n\nRecursion Example Results")
tri_recursion(6)
```

Python Lambda

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

Syntax

lambda arguments : expression

The expression is executed and the result is returned:

Example

Add 10 to argument a, and return the result:

```
x = lambda a : a + 10
print(x(5))
```

Initializing a Matrix in Python :

Method #1 : Using List comprehension

```
N = 5
M = 4
res = [ [ 0 for i in range(M) ] for j in range(N) ]
print("The matrix after initializing : " + str(res))
```

Python import Statement

The Python import statement imports code from one module into another program. You can import all the code from a module by specifying the import keyword followed by the module you want to import.

The syntax for the import statement is:

```
import [module]
```

Example:

import time

time.sleep(3)

copy — Shallow and deep copy operations

The difference between shallow and deep copying is only relevant for compound objects (objects that contain other objects, like lists or class instances):

A shallow copy constructs a new compound object and then (to the extent possible) inserts references into it to the objects found in the original.

A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.

copy.copy(x)

Return a shallow copy of x.

copy.deepcopy(x[, memo])

Return a deep copy of x.

tkinter — Python interface to Tcl/Tk

The tkinter package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit.

Tkinter Modules

Support for Tkinter is spread across several modules.

from tkinter import *

There are two main methods used which the user needs to remember while creating the Python application with GUI.

Tk(screenName=None, baseName=None, className='Tk', useTk=1): To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

m=tkinter.Tk() where m is the name of the main window object

mainloop(): There is a method known by the name mainloop() that is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

m.mainloop()

```
import tkinter

m = tkinter.Tk()

'''

widgets are added here

'''

m.mainloop()
```

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes.

pack() method: It organizes the widgets in blocks before placing them in the parent widget.

grid() method: It organizes the widgets in grid (table-like structure) before placing them in the parent widget.

place() method:It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

Button: To add a button in your application, this widget is used.

The general syntax is:

w=Button(master, option=value)

master is the parameter used to represent the parent window.

There are a number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas.

2.2 Papers/Findings:

Following articles and paper were used:

1. https://towardsdatascience.com/solving-mazes-with-python-f7a412f2493f
2. https://towardsdatascience.com/maze-rl-d035f9ccdc63
3. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
4. https://youtu.be/XB4MIexjvY0

Analysis and Design:

Analysis of the system:

The program will consist of an interface where a 5x5 grid will initially be available to the user.

The user will be given the option to select the start point, the destination and construct the maze by selecting appropriate obstacles.

All these operations can be performed using the buttons available at the top of the application.

The start point will be highlighted in red, while the destination will be in blue colour.

The obstacles will be selected one by one and the selected obstacle squares will be greyed out.

After constructing the desired maze, the user will be able to view the shortest path between the start point and the destination by clicking the 'go' button.

This path will be highlighted in green colour and would be the solution of the maze.

Proposed Solution:

The maze solution will be derived using the Dijkstra algorithm.

All the selected obstacles, starting point and the destination will be treated as a part of a graph.

The algorithm will analyze all possible paths from the starting position to the destination and will return the shortest part.

The user interface to implement the maze solver will be constructed using Tkinter.

The backend and frontend part of the program will work together and the final application will be implemented.

Design of the proposed system:

The system will be divided into three scripts:

The first script will generate and handle the adjacency matrix

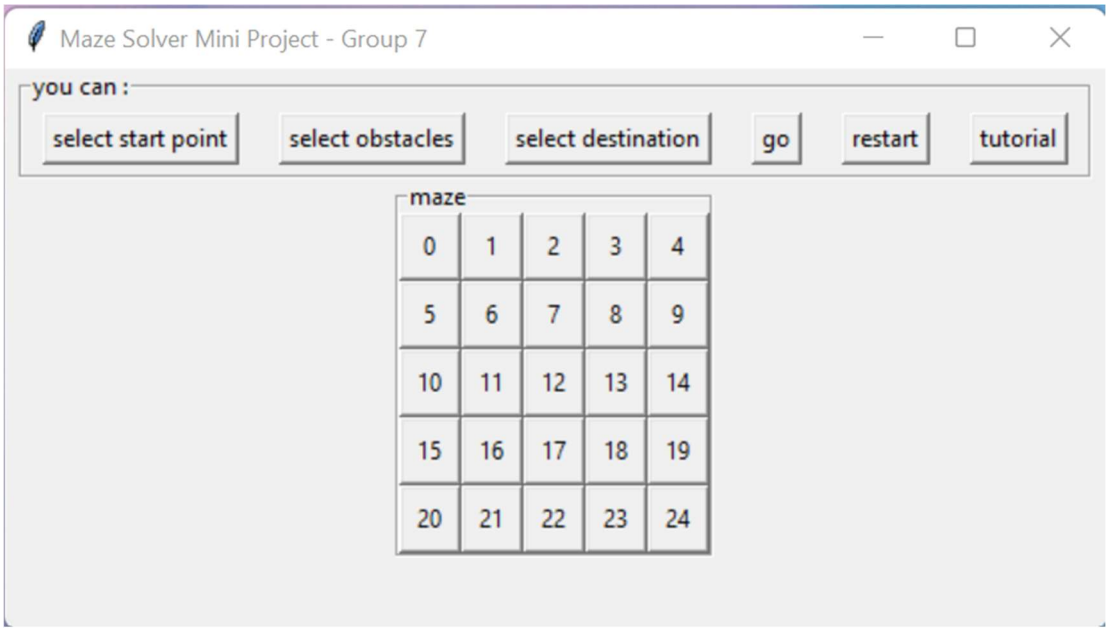
The second script will receive the adjacency matrix and will implement Dijkstra algorithm logic on the generated matrix.

Third script will import the second script and give a GUI to our project.

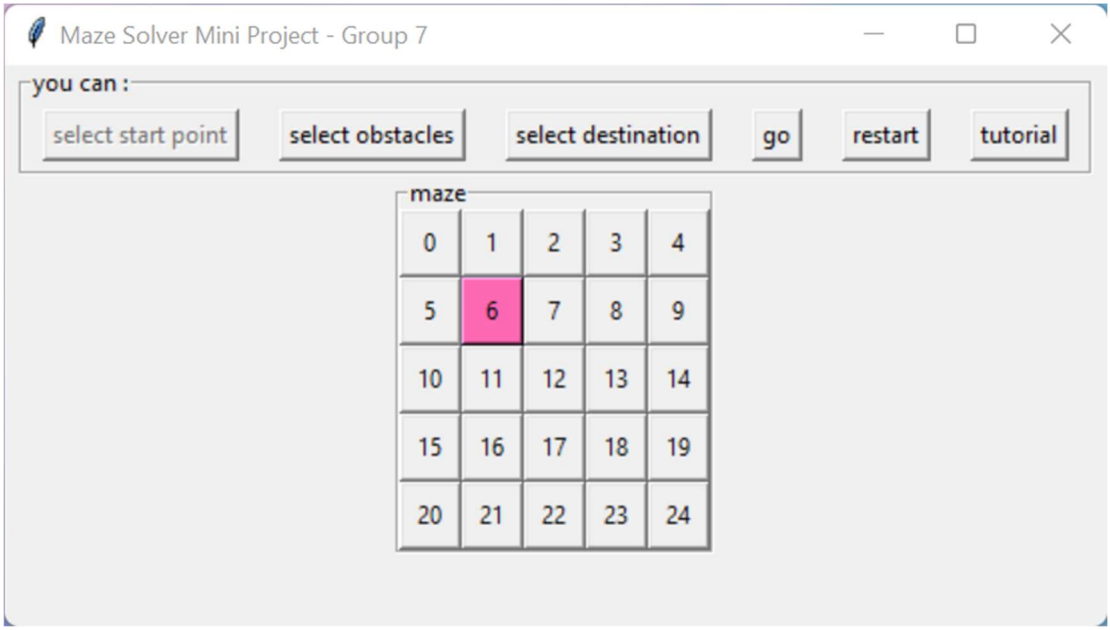
The first script will generate the adjacency matrix depending on the obstacles generated by the user.

Results and discussion:

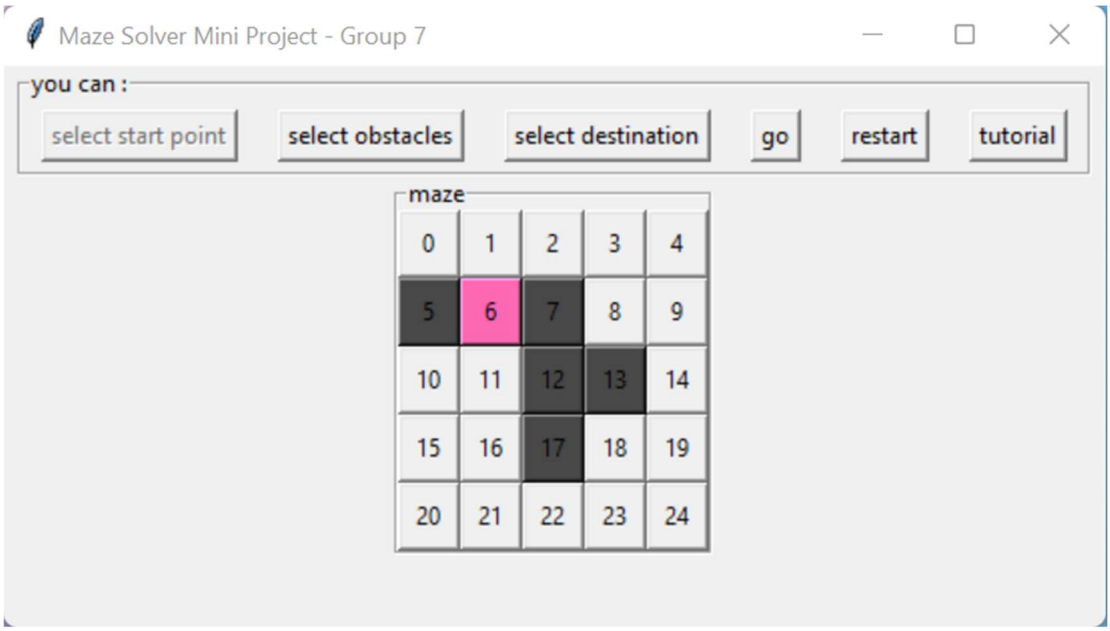
On starting



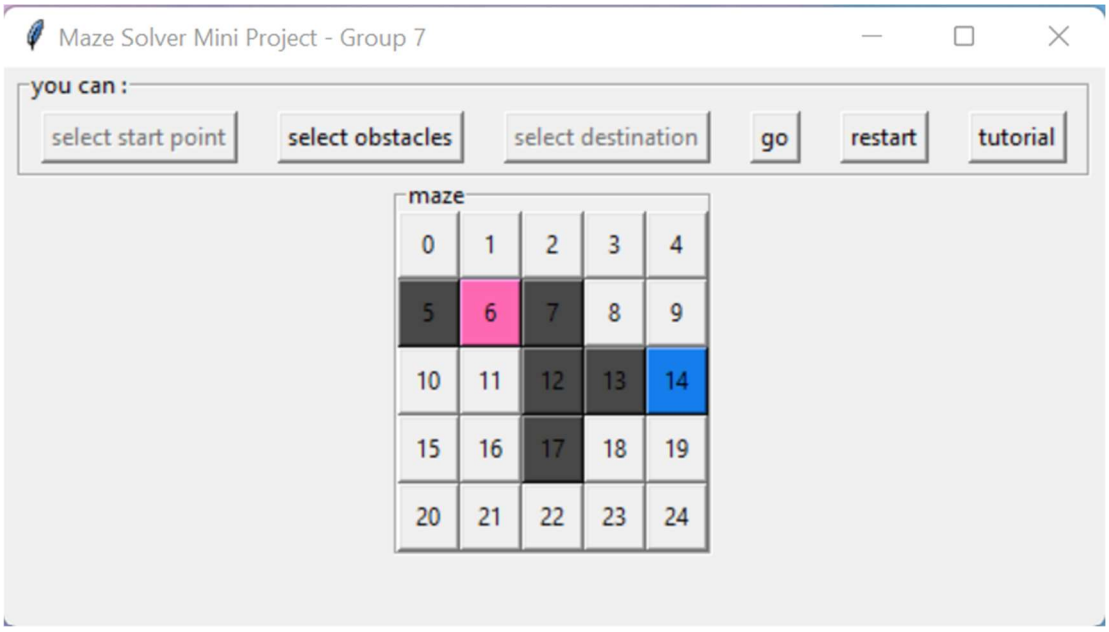
On selecting start point



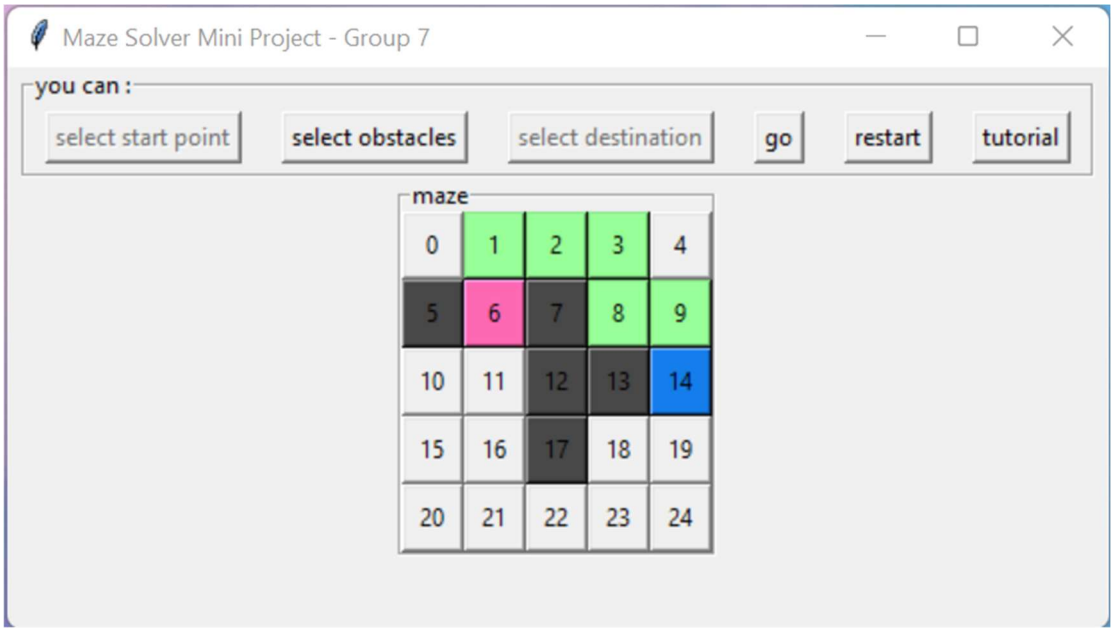
On selecting obstacles



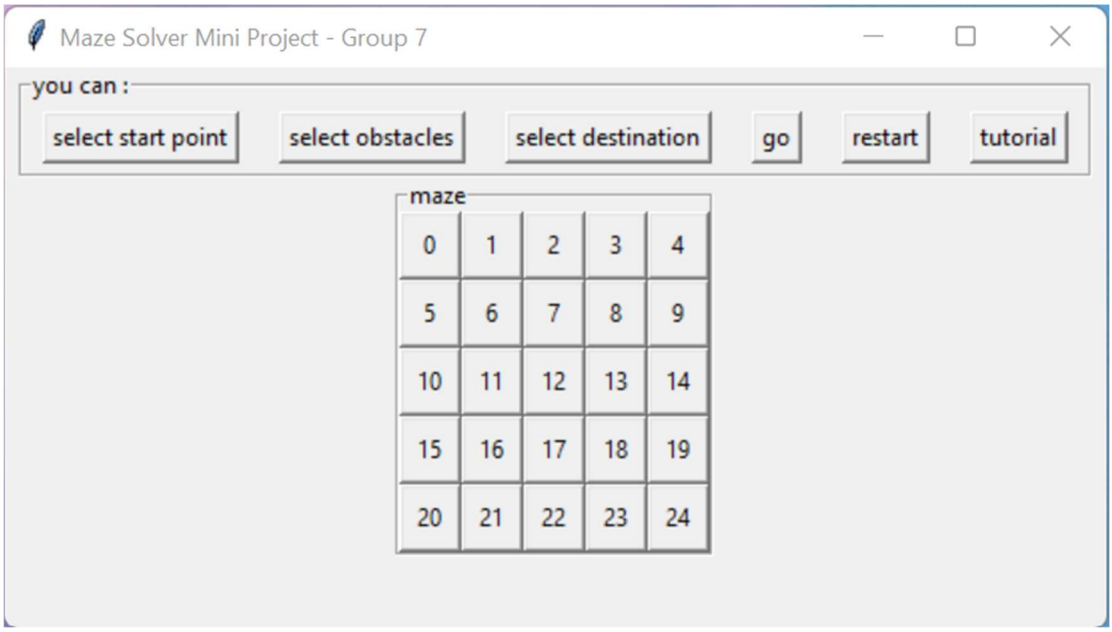
On selecting destination



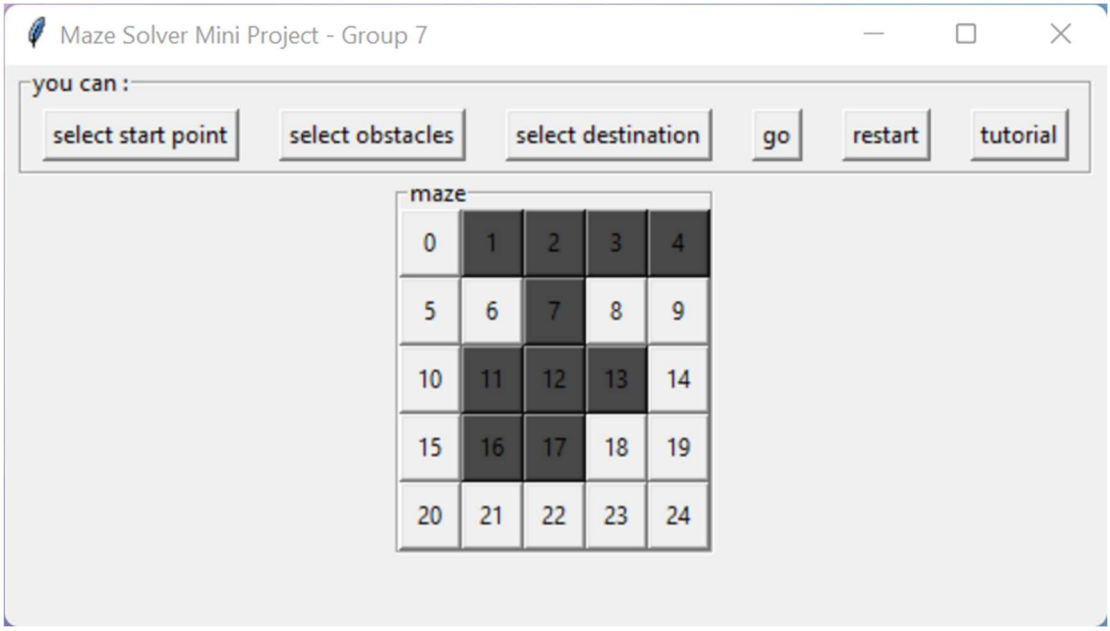
On clicking 'go' button



On clicking restart



On clicking tutorial(already generated maze is displayed)



This was the final result of our project.

Conclusion and future work:

Conclusion:

Different types of maze-solving algorithms are often needed

for different mazes. However, there is no perfect general algorithm for that purpose. The algorithm discussed in this paper will give us a general method applicable for almost any kind of maze. The algorithm can also be used in sectors beyond line following. After some modifications, it can be used for autonomous road mapping, cave mapping, exploring and many other purposes. In these cases, low-cost GPS devices can be used instead of wheel encoders for tracking coordinates.

Future work:

This program can be refined by adding the concept of auto maze generation. A random maze will automatically be generated by the program and the user can solve the maze.

The solution of the generated maze also will be available.

Mechanics can be used in combination with this program to create a real life maze solving robot.

This robot could be able to detect the obstacles and hence find the shortest path to its destination.