

PATH PLANNING CODE

Libraries used:

- random library for obstacle generation
- heapq library to assist in formation of algorithm
- tkinter library to generate the graphical user interface

Classes utilized:

- robot class to assign the path generated by the algorithm to an object
- GRIDGUI class to easily manipulate elements of the grid such as colours, number of cells, and size of cells

Process:

Firstly, classes were initialized to create elements of the grid such as a colour code for each attribute present on the UI (robot, path, obstacle, goal). The draw_grid function was used to describe each cell and categorize it based on the object present inside it

The grid was initialized, and obstacles ranging from 10 to 20 are randomly assigned to elements of the matrix created. Starting position of robot was set to (0,0) and goal was set to last element of matrix to ensure validity of algorithm code in long distances across the entire grid.

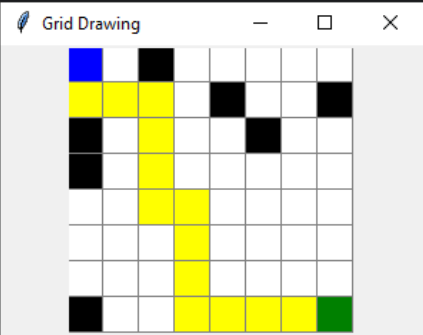
The A-star algorithm function was then created using the Manhattan distance as its heuristic due to the limited movements available for the robot (up, down, left, right)

Neighbours of the starting cell were each checked for whether they are an obstacle or not or if they are within bounds, and an array is created to check each one for its real and heuristic cost. The process is repeated each time calculating the cumulative cost and the path is checked for completeness by the `valid_path` function and assigned to the robot via the `set_path` function.

A `__main__` function was created to initialize all elements of the system including create the robot object, apply obstacles, figure out a valid path using the A_STAR algorithm and colour code the grid

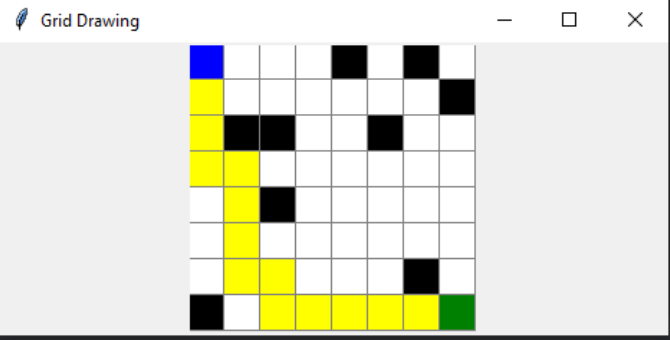
The integration of TKINTER into the code was mostly done using AI and functions provided were then studied to add the integrated function into the code whilst understanding it, and to add the proper comments to the code. GEEKSFORGEEKS, SIMPLILEARN, and Python docs were used whenever new functions were provided by the AI software.

```
C:\Users\compufast\PycharmProjects\gear\.venv\Scripts\python.exe "C:\Users\compufast\PycharmProjects\gear\path planning.py"
robot path -->[(0, 1), (1, 1), (2, 1), (2, 2), (2, 3), (2, 4), (3, 4), (3, 5), (3, 6), (3, 7), (4, 7), (5, 7), (6, 7)]
```



The screenshot shows a terminal window with the command `C:\Users\compufast\PycharmProjects\gear\.venv\Scripts\python.exe "C:\Users\compufast\PycharmProjects\gear\path planning.py"` and the output `robot path -->[(0, 1), (1, 1), (2, 1), (2, 2), (2, 3), (2, 4), (3, 4), (3, 5), (3, 6), (3, 7), (4, 7), (5, 7), (6, 7)]`. Below the terminal is a Tkinter window titled "Grid Drawing" showing a 10x10 grid. The start cell at (0,1) is blue, and the path cells are yellow. Obstacles are represented by black cells at (0,2), (1,2), (1,3), (1,4), (1,5), (1,6), (2,5), (2,6), (2,7), (3,2), (3,3), (3,6), (4,2), (4,3), (4,4), (4,5), (4,6), (5,2), (5,3), (5,4), (5,5), (5,6), (6,2), (6,3), (6,4), (6,5), (6,6), (7,2), (7,3), (7,4), (7,5), (7,6), (7,7), (8,2), (8,3), (8,4), (8,5), (8,6), (8,7), (9,2), (9,3), (9,4), (9,5), (9,6), (9,7).

```
C:\Users\compufast\PycharmProjects\gear\.venv\Scripts\python.exe "C:\Users\compufast\PycharmProjects\gear\path planning.py"
robot path -->[(0, 1), (0, 2), (0, 3), (1, 3), (1, 4), (1, 5), (1, 6), (2, 6), (2, 7), (3, 7), (4, 7), (5, 7), (6, 7)]
```



The screenshot shows a terminal window with the command `C:\Users\compufast\PycharmProjects\gear\.venv\Scripts\python.exe "C:\Users\compufast\PycharmProjects\gear\path planning.py"` and the output `robot path -->[(0, 1), (0, 2), (0, 3), (1, 3), (1, 4), (1, 5), (1, 6), (2, 6), (2, 7), (3, 7), (4, 7), (5, 7), (6, 7)]`. Below the terminal is a Tkinter window titled "Grid Drawing" showing a 10x10 grid. The start cell at (0,1) is blue, and the path cells are yellow. Obstacles are represented by black cells at (0,4), (0,5), (0,6), (0,7), (1,1), (1,2), (1,5), (1,6), (1,7), (2,1), (2,2), (2,3), (2,4), (2,5), (2,7), (3,1), (3,2), (3,3), (3,4), (3,5), (3,6), (4,1), (4,2), (4,3), (4,4), (4,5), (4,6), (5,1), (5,2), (5,3), (5,4), (5,5), (5,6), (6,1), (6,2), (6,3), (6,4), (6,5), (6,6), (7,1), (7,2), (7,3), (7,4), (7,5), (7,6), (7,7), (8,1), (8,2), (8,3), (8,4), (8,5), (8,6), (8,7), (9,1), (9,2), (9,3), (9,4), (9,5), (9,6), (9,7).

