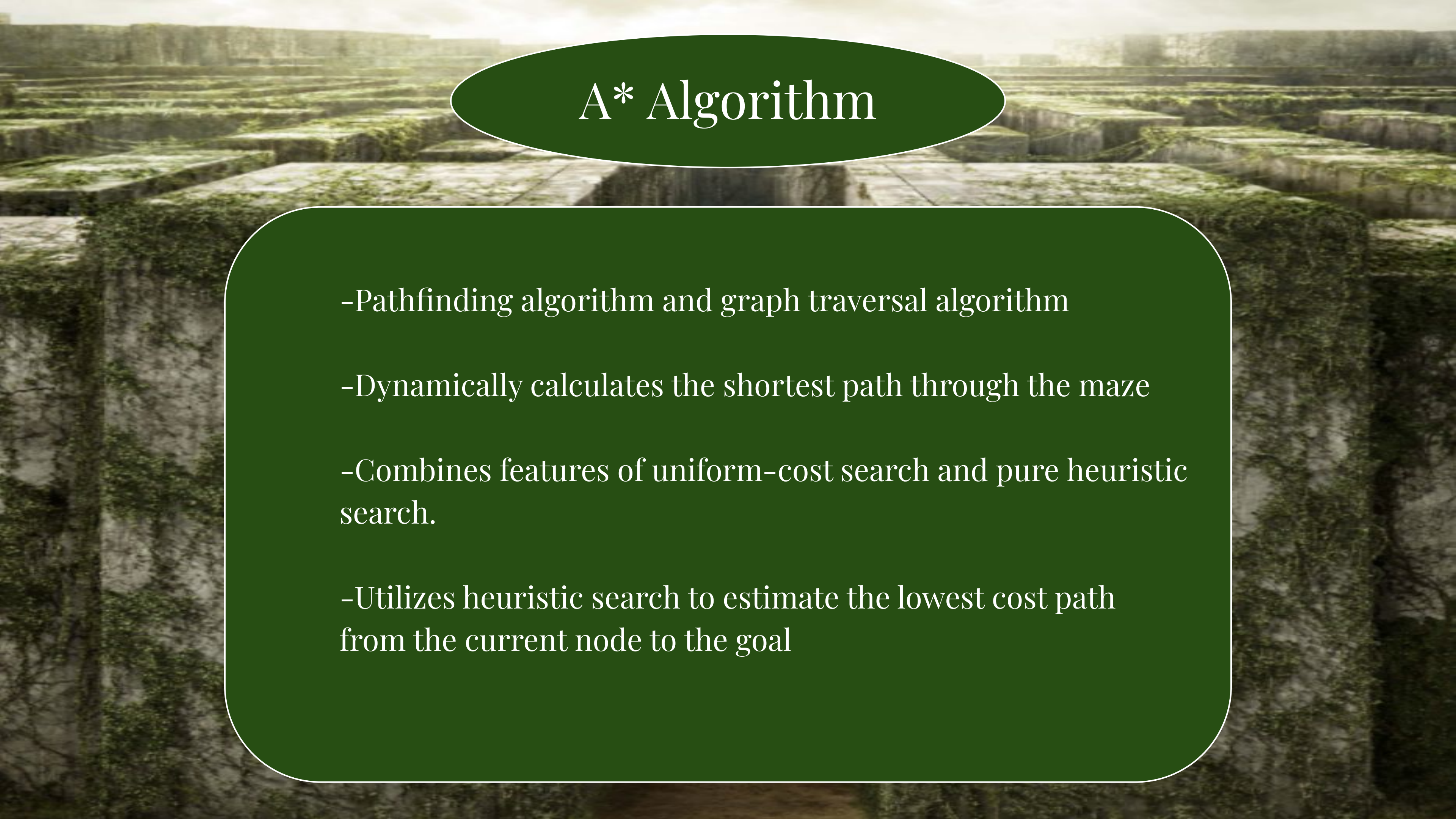
A* Maze Runner

Shayan Darian
Loc Nguyen
Jake Watson
Michael Yu

Problem Statement

-Educate users on A* algorithm and provide step-by-step visualization of the A* pathfinding process through the maze solution

-Develop a software application that generates and solves mazes using A* pathfinding algorithm.



A* Algorithm

-Pathfinding algorithm and graph traversal algorithm

-Dynamically calculates the shortest path through the maze

-Combines features of uniform-cost search and pure heuristic search.

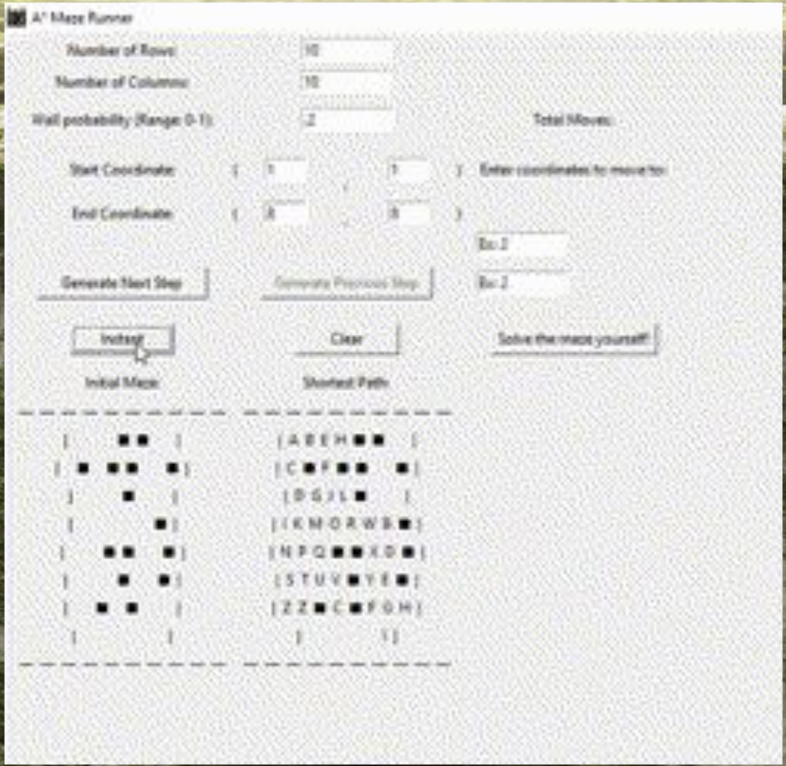
-Utilizes heuristic search to estimate the lowest cost path from the current node to the goal

Approach

Python

Tkinter

A* Algorithm



A* Maze Runner

Number of Rows: 10

Number of Columns: 10

Wall probability (Range 0-1): .2

Total Moves:

Start Coordinate: (1 , 1)

End Coordinate: (8 , 8)

Enter coordinates to move to:

Ex: 2

Ex: 2

Initial Maze:

```
- - - - - - - - - -
|    ■ ■    |
| ■  ■ ■   ■|
|    ■      |
|         ■|
|   ■ ■   ■|
|    ■   ■|
|   ■  ■    |
|           |
- - - - - - - - - -
```

Shortest Path:

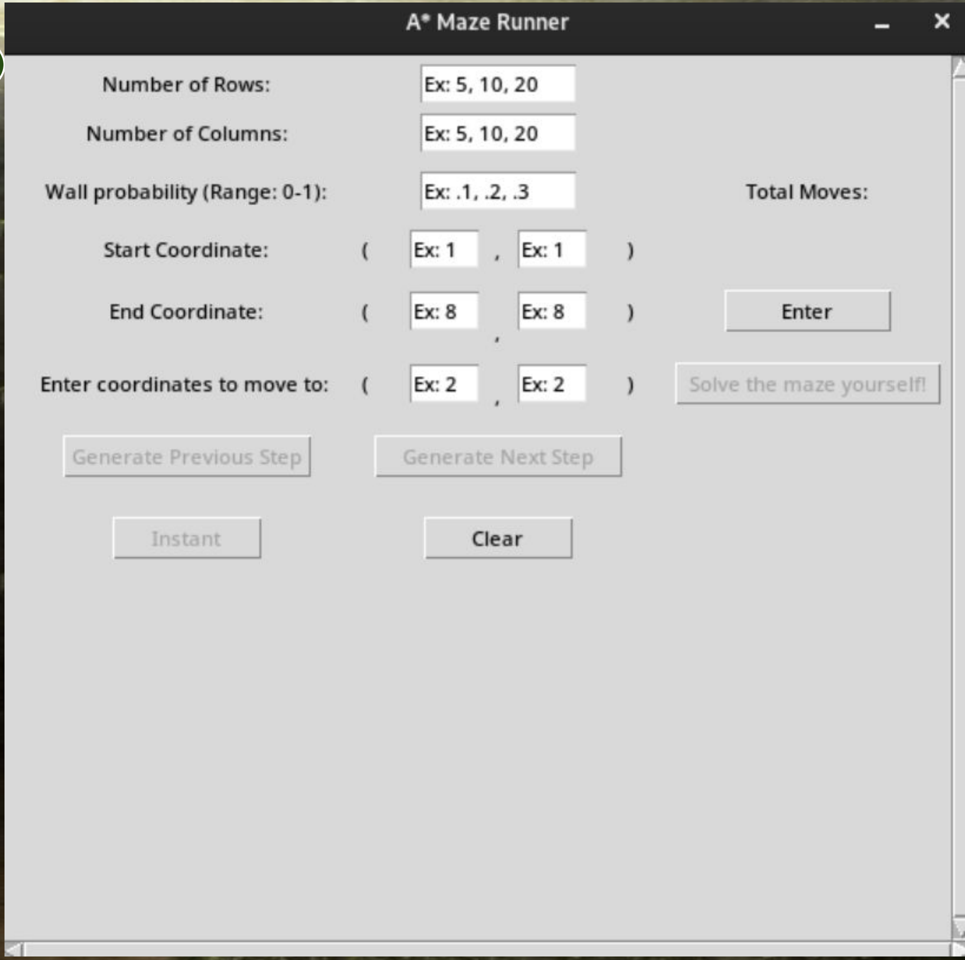
```
- - - - - - - - - -
|A B E H ■ ■   |
|C ■ F ■ ■   ■|
|D G J L ■     |
|I K M O R W B ■|
|N P Q ■ ■ X D ■|
|S T U V ■ Y E ■|
|Z Z ■ C ■ F G H|
|             I|
- - - - - - - - - -
```

Code Layout

AstarMazeRunner.py
(imports)
Astar_gen.py
(imports)
Scroll.py

AstarMazeRunner.py

Tkinter-based GUI for user interaction, maze generation logic, and
step-by-step path visualization feature.



A* Maze Runner

Number of Rows: Ex: 5, 10, 20

Number of Columns: Ex: 5, 10, 20

Wall probability (Range: 0-1): Ex: .1, .2, .3

Total Moves:

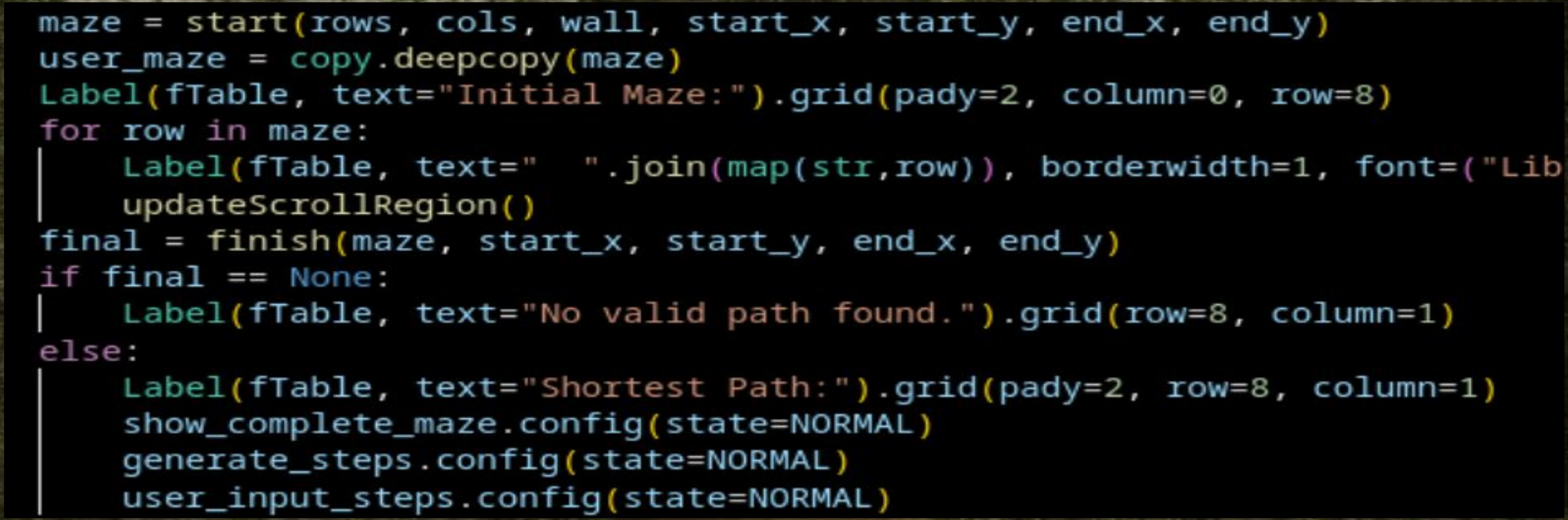
Start Coordinate: (Ex: 1 , Ex: 1)

End Coordinate: (Ex: 8 , Ex: 8)

Enter coordinates to move to: (Ex: 2 , Ex: 2)

AstarMazeRunner.py

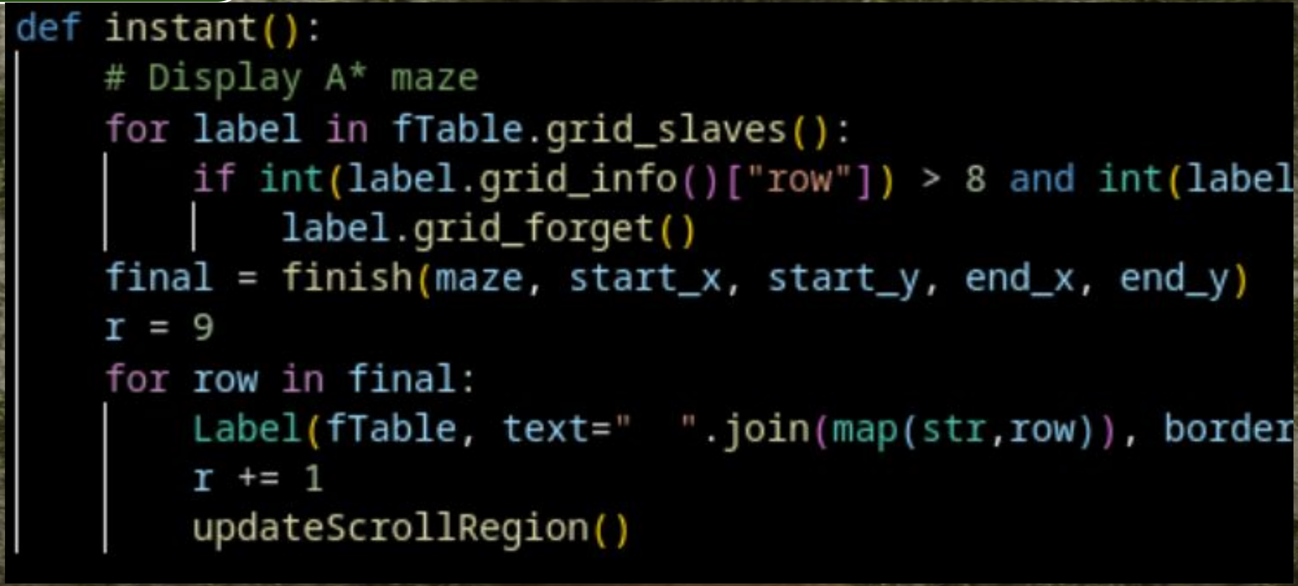
Enter Button



```
maze = start(rows, cols, wall, start_x, start_y, end_x, end_y)
user_maze = copy.deepcopy(maze)
Label(fTable, text="Initial Maze:").grid(pady=2, column=0, row=8)
for row in maze:
    Label(fTable, text="  ".join(map(str,row)), borderwidth=1, font=("Lib
    updateScrollRegion()
final = finish(maze, start_x, start_y, end_x, end_y)
if final == None:
    Label(fTable, text="No valid path found.").grid(row=8, column=1)
else:
    Label(fTable, text="Shortest Path:").grid(pady=2, row=8, column=1)
    show_complete_maze.config(state=NORMAL)
    generate_steps.config(state=NORMAL)
    user_input_steps.config(state=NORMAL)
```

AstarMazeRunner.py

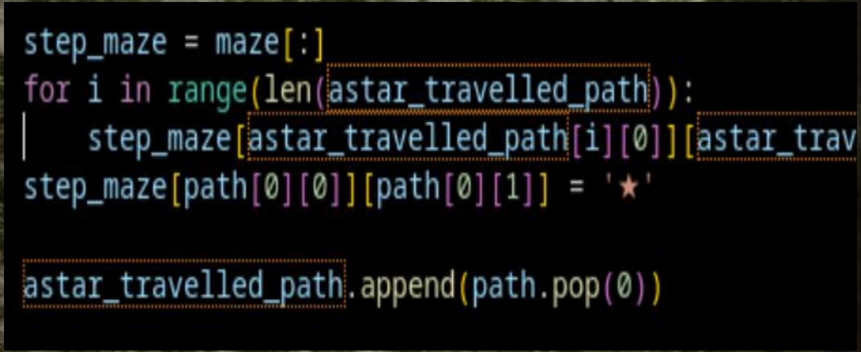
Instant Button



```
def instant():
    # Display A* maze
    for label in fTable.grid_slaves():
        if int(label.grid_info()["row"]) > 8 and int(label
            label.grid_forget()
    final = finish(maze, start_x, start_y, end_x, end_y)
    r = 9
    for row in final:
        Label(fTable, text="  ".join(map(str,row)), border
        r += 1
        updateScrollRegion()
```

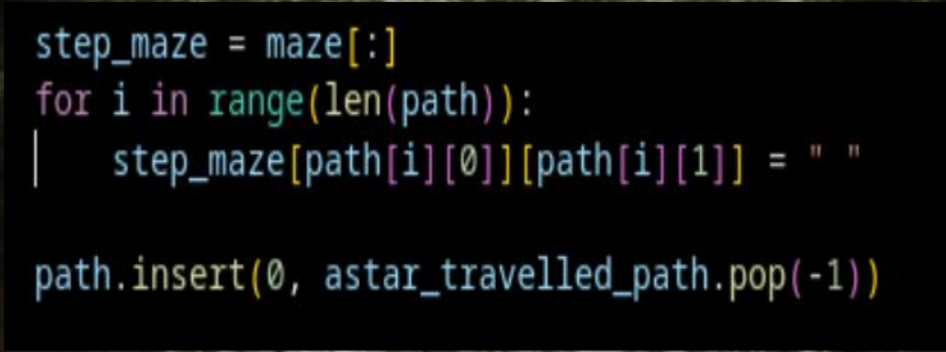
AstarMazeRunner.py

Next/Previous Button



```
step_maze = maze[:]
for i in range(len(astar_travelled_path)):
    step_maze[astar_travelled_path[i][0]][astar_trav
step_maze[path[0][0]][path[0][1]] = '★'

astar_travelled_path.append(path.pop(0))
```

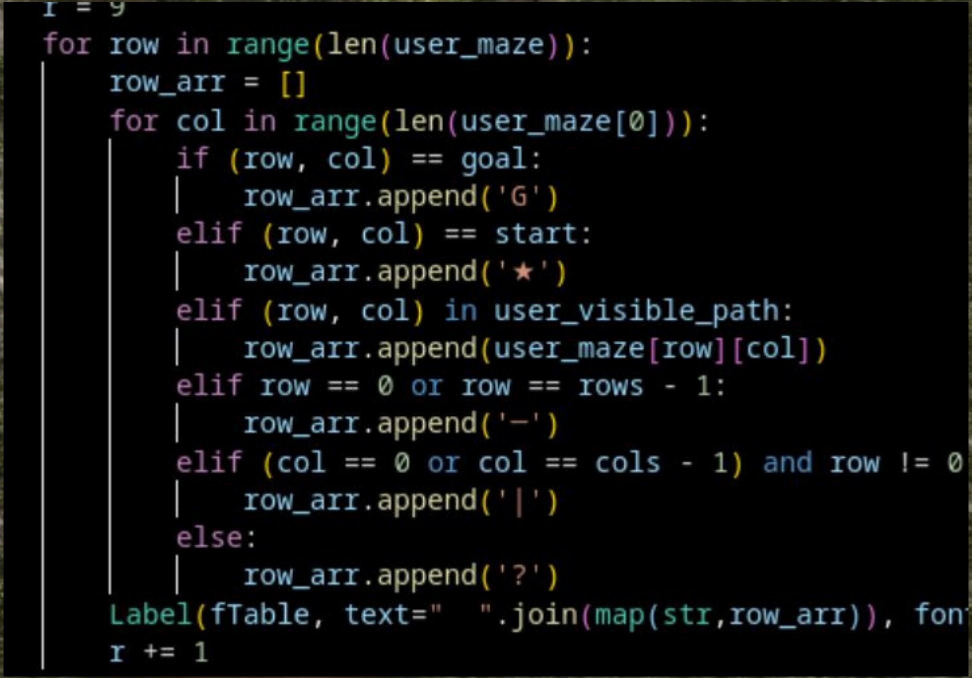


```
step_maze = maze[:]
for i in range(len(path)):
    step_maze[path[i][0]][path[i][1]] = " "

path.insert(0, astar_travelled_path.pop(-1))
```

AstarMazeRunner.py

User Solving Button



```
r = 9
for row in range(len(user_maze)):
    row_arr = []
    for col in range(len(user_maze[0])):
        if (row, col) == goal:
            row_arr.append('G')
        elif (row, col) == start:
            row_arr.append('★')
        elif (row, col) in user_visible_path:
            row_arr.append(user_maze[row][col])
        elif row == 0 or row == rows - 1:
            row_arr.append('—')
        elif (col == 0 or col == cols - 1) and row != 0
            row_arr.append('|')
        else:
            row_arr.append('?')
    Label(fTable, text="  ".join(map(str,row_arr)), fon
    r += 1
```

astar_gen.py

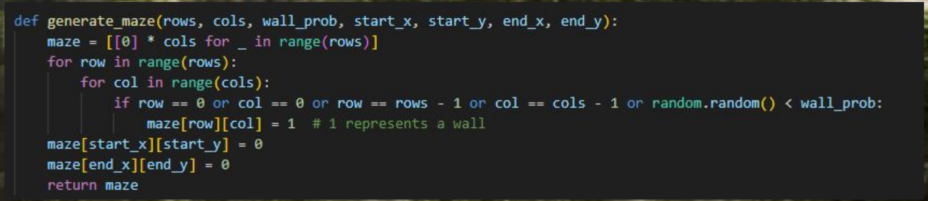
Implementation of A* pathfinding algorithm and maze generation logic

Functions

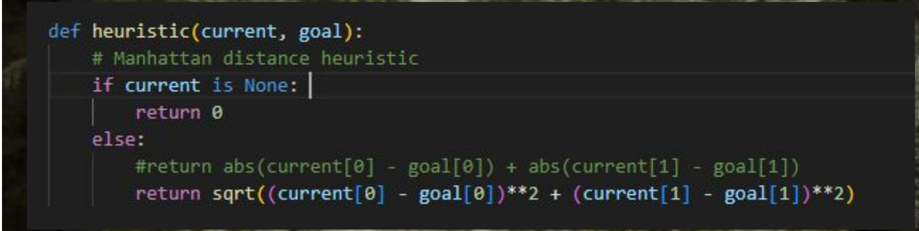
generate_maze : generates a 2D array maze based on user-specified parameters

heuristic : calculates estimated cost from the current node to the goal node.

astar_pathfind_gen : performs the A* pathfinding algorithm



```python
def generate_maze(rows, cols, wall_prob, start_x, start_y, end_x, end_y):
    maze = [[0] * cols for _ in range(rows)]
    for row in range(rows):
        for col in range(cols):
            if row == 0 or col == 0 or row == rows - 1 or col == cols - 1 or random.random() < wall_prob:
                maze[row][col] = 1  # 1 represents a wall
    maze[start_x][start_y] = 0
    maze[end_x][end_y] = 0
    return maze
```

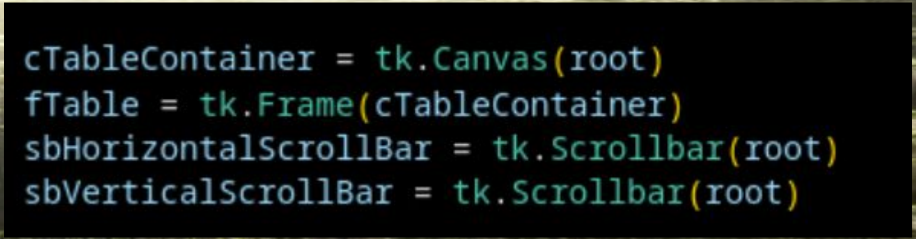


```python
def heuristic(current, goal):
    # Manhattan distance heuristic
    if current is None:
        return 0
    else:
        #return abs(current[0] - goal[0]) + abs(current[1] - goal[1])
        return sqrt((current[0] - goal[0])**2 + (current[1] - goal[1])**2)
```

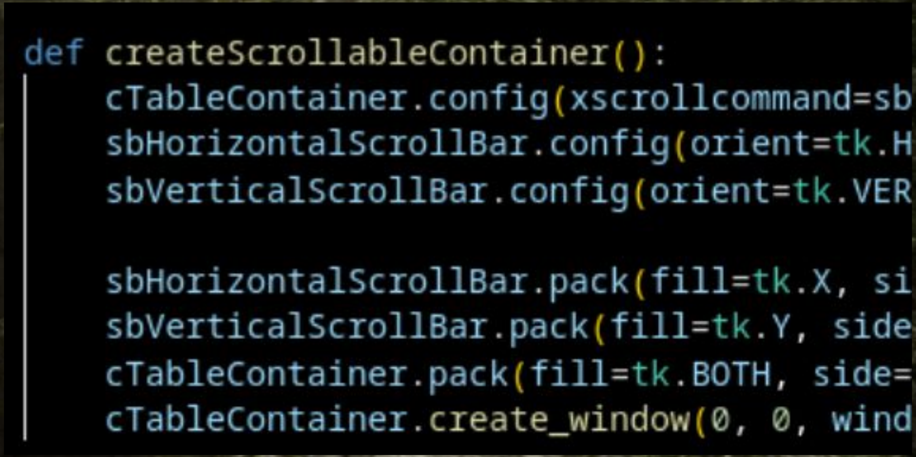
scroll.py

Implementation of a vertical and horizontal scroll bar

Allows for a dynamically changing GUI

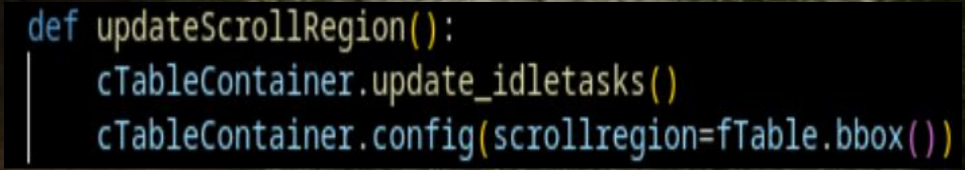


```
cTableContainer = tk.Canvas(root)
fTable = tk.Frame(cTableContainer)
sbHorizontalScrollBar = tk.Scrollbar(root)
sbVerticalScrollBar = tk.Scrollbar(root)
```



```
def createScrollableContainer():
    cTableContainer.config(xscrollcommand=sb
    sbHorizontalScrollBar.config(orient=tk.H
    sbVerticalScrollBar.config(orient=tk.VER

    sbHorizontalScrollBar.pack(fill=tk.X, si
    sbVerticalScrollBar.pack(fill=tk.Y, side
    cTableContainer.pack(fill=tk.BOTH, side=
    cTableContainer.create_window(0, 0, wind
```



```
def updateScrollRegion():
    cTableContainer.update_idletasks()
    cTableContainer.config(scrollregion=fTable.bbox())
```

Demo

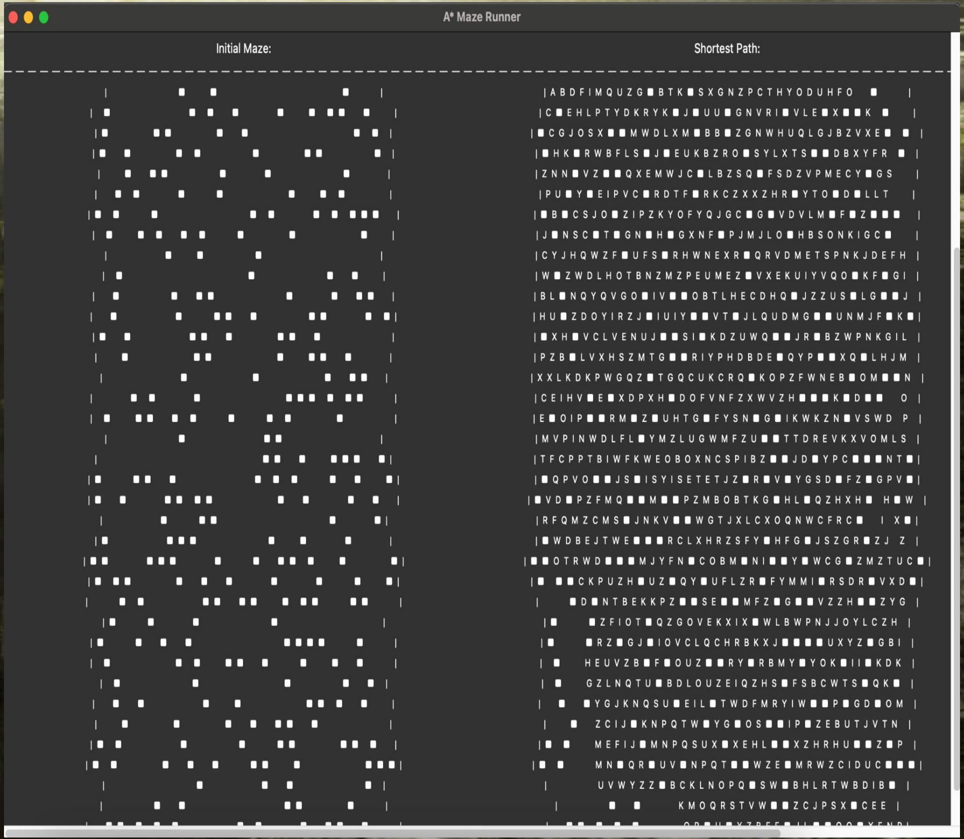
Future Work

- Scalability

- Switch out text characters for images

- Allow users to submit their custom maze

-More algorithms; BFS, DFS, Dijkstra's Algorithm



```
Initial Maze:
---------------------------------------------------------------------------
|                     ■       ■                                   ■     |
|   ■                         ■   ■     ■           ■       ■   ■ ■     ■   |
| ■             ■ ■             ■     ■                               ■   ■ |
| ■     ■             ■   ■               ■             ■ ■               ■ |
|       ■     ■ ■               ■           ■                     ■         |
|     ■   ■           ■         ■                   ■       ■   ■           |
| ■   ■         ■                           ■   ■           ■   ■   ■ ■ ■   |
|   ■       ■   ■     ■   ■         ■             ■                   ■     |
|                 ■       ■               ■                                 |
|   ■                                     ■                     ■     ■     |
|     ■               ■     ■ ■                     ■           ■     ■ ■   |
|     ■                 ■         ■ ■     ■               ■ ■           ■   ■ |
| ■     ■                 ■ ■     ■             ■ ■     ■                   |
|       ■                   ■ ■                   ■       ■ ■     ■         |
|                       ■                   ■                   ■     ■ ■   |
|           ■   ■           ■                         ■ ■ ■   ■   ■ ■     |
|   ■       ■ ■     ■   ■         ■         ■   ■             ■             |
|                     ■                       ■ ■                           |
|                                                 ■ ■     ■       ■ ■ ■     ■ |
| ■         ■ ■     ■                       ■   ■   ■         ■     ■       ■ |
| ■     ■           ■ ■   ■ ■                   ■     ■           ■     ■   |
|                 ■         ■ ■                                 ■       ■ |
| ■                 ■ ■ ■                     ■       ■           ■         |
| ■ ■           ■ ■ ■           ■         ■     ■ ■   ■       ■             ■ |
| ■   ■ ■             ■     ■     ■           ■           ■         ■       ■ |
|       ■   ■                 ■ ■     ■ ■       ■   ■ ■         ■ ■       |
| ■       ■           ■                     ■                             |
| ■       ■     ■     ■                           ■ ■ ■ ■         ■       |
|   ■                 ■   ■       ■ ■     ■         ■       ■     ■       |
|   ■                   ■                         ■               ■     ■ |
|   ■   ■                 ■       ■                   ■ ■   ■     ■     |
|     ■           ■             ■     ■     ■ ■     ■                   |
| ■   ■               ■               ■         ■ ■             ■ ■   ■   |
| ■   ■           ■     ■     ■         ■ ■       ■                   ■ ■ ■ |
|                       ■                 ■     ■                     ■ |
|           ■     ■                           ■ ■             ■       |
```

```
Shortest Path:
---------------------------------------------------------------------------
| A B D F I M Q U Z G ■ B T K ■ S X G N Z P C T H Y O D U H F O   ■     |
| C ■ E H L P T Y D K R Y K ■ J ■ U U ■ G N V R I ■ V L E ■ X ■ ■ K   ■   |
| ■ C G J O S X ■ ■ M W D L X M ■ B B ■ Z G N W H U Q L G J B Z V X E ■   ■ |
| ■ H K ■ R W B F L S ■ J ■ E U K B Z R O ■ S Y L X T S ■ ■ D B X Y F R   ■ |
| Z N N ■ V Z ■ ■ Q X E M W J C ■ L B Z S Q ■ F S D Z V P M E C Y ■ G S     |
| P U ■ Y ■ E I P V C ■ R D T F ■ R K C Z X X Z H R ■ Y T O ■ D ■ L L T     |
| ■ B ■ C S J O ■ Z I P Z K Y O F Y Q J G C ■ G ■ V D V L M ■ F ■ Z ■ ■ ■   |
| J ■ N S C ■ T ■ G N ■ H ■ G X N F ■ P J M J L O ■ H B S O N K I G C ■     |
| C Y J H Q W Z F ■ U F S ■ R H W N E X R ■ Q R V D M E T S P N K J D E F H |
| W ■ Z W D L H O T B N Z M Z P E U M E Z ■ V X E K U I Y V Q O ■ K F ■ G I |
| B L ■ N Q Y Q V G O ■ I V ■ ■ O B T L H E C D H Q ■ J Z Z U S ■ L G ■ ■ J |
| H U ■ Z D O Y I R Z J ■ I U I Y ■ ■ V T ■ J L Q U D M G ■ ■ U N M J F ■ K ■ |
| ■ X H ■ V C L V E N U J ■ ■ S I ■ K D Z U W Q ■ ■ J R ■ B Z W P N K G I L |
| P Z B ■ L V X H S Z M T G ■ ■ R I Y P H D B D E ■ Q Y P ■ ■ X Q ■ L H J M |
| X X L K D K P W G Q Z ■ T G Q C U K C R Q ■ K O P Z F W N E B ■ O M ■ ■ N |
| C E I H V ■ E ■ X D P X H ■ D O F V N F Z X W V Z H ■ ■ ■ K ■ D ■ ■   O |
| E ■ O I P ■ ■ R M ■ Z ■ U H T G ■ F Y S N ■ G ■ I K W K Z N ■ V S W D   P |
| M V P I N W D L F L ■ Y M Z L U G W M F Z U ■ ■ T T D R E V K X V O M L S |
| T F C P P T B I W F K W E O B O X N C S P I B Z ■ ■ J D ■ Y P C ■ ■ ■ N T ■ |
| ■ Q P V O ■ ■ J S ■ I S Y I S E T E T J Z ■ R ■ V ■ Y G S D ■ F Z ■ G P V ■ |
| ■ V D ■ P Z F M Q ■ ■ M ■ ■ P Z M B O B T K G ■ H L ■ Q Z H X H ■   H ■ W |
| R F Q M Z C M S ■ J N K V ■ ■ W G T J X L C X O Q N W C F R C ■   I X ■ |
| ■ W D B E J T W E ■ ■ ■ R C L X H R Z S F Y ■ H F G ■ J S Z G R ■ Z J   Z |
| ■ ■ O T R W D ■ ■ ■ M J Y F N ■ C O B M ■ N I ■ ■ Y ■ W C G ■ Z M Z T U C ■ |
| ■   ■ ■ C K P U Z H ■ U Z ■ Q Y ■ U F L Z R ■ F Y M M I ■ R S D R ■ V X D ■ |
|       ■ D ■ N T B E K K P Z ■ ■ S E ■ ■ M F Z ■ G ■ ■ V Z Z H ■ ■ Z Y G |
| ■       ■ Z F I O T ■ Q Z G O V E K X I X ■ W L B W P N J J O Y L C Z H |
| ■       ■ R Z ■ G J ■ I O V C L Q C H R B K X J ■ ■ ■ ■ U X Y Z ■ G B I |
|   ■     H E U V Z B ■ F ■ O U Z ■ ■ R Y ■ R B M Y ■ Y O K ■ I I ■ K D K |
|   ■     G Z L N Q T U ■ B D L O U Z E I Q Z H S ■ F S B C W T S ■ Q K ■ |
|   ■   ■ Y G J K N Q S U ■ E I L ■ T W D F M R Y I W ■ ■ P ■ G D ■ O M |
|     ■   Z C I J ■ K N P Q T W ■ Y G ■ O S ■ ■ I P ■ Z E B U T J V T N |
| ■   ■     M E F I J ■ M N P Q S U X ■ X E H L ■ ■ X Z H R H U ■ ■ Z ■ P |
| ■   ■       M N ■ Q R ■ U V ■ N P Q T ■ ■ W Z E ■ M R W Z C I D U C ■ ■ ■ |
|           U V W Y Z Z ■ B C K L N O P Q ■ S W ■ B H L R T W B D I B ■ |
|           ■     ■         K M O Q R S T V W ■ ■ Z C J P S X ■ C E E |
```

References

[1] “Tkinter Course - Create Graphic User Interfaces in Python Tutorial.” YouTube, YouTube, 19 Nov. 2019, www.youtube.com/watch?v=YXPyB4XeYLA.

[2] Joehutch. How to Add Scrollbars to a Dynamic GUI in TKinter · Joe Hutchinson. (n.d.). https://www.joehutch.com/posts/tkinter-dynamic-scroll-area

[3] Barnouti, N. , Al-Dabbagh, S. and Sahib Naser, M. (2016) Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm. Journal of Computer and Communications, 4, 15-25. doi: 10.4236/jcc.2016.411002.

[4] Tjiharjadi, Semuil, Marvin Wijaya, and Erwin Setiawan. "Optimization maze robot using A* and flood fill algorithm." International Journal of Mechanical Engineering and Robotics Research 6.5 (2017): 366-372.