Authors: Michael Wood, Jack Gallagher

Project Description:

Our project depicts a simplified version of a robot nurse delivering medication to requested locations in a hospital. The code uses either the A* or Dijkstra's algorithm to find the most optimal path from the robot's current position to the next requested delivery location. The hospital floor plan is represented by a 36x30 matrix with cells of a variety of different values to label and color walls, wards, and outside of the building. The code accepts input details from a text file, which include the algorithm it uses, its starting location, and a number of delivery locations. The delivery locations are placed into a priority queue based on their cell values and corresponding ward priority. Once all input values are entered correctly, the code uses the given algorithm to find the optimal path from the starting location to the first delivery location in the priority queue. If there are multiple delivery locations, the first delivery location becomes the new starting location. The algorithm finds the optimal path to the new delivery location from there. If there is a delivery location that cannot be reached by the robot, then that location will be labeled as 'Failed'. The robot will then go to the next available delivery location instead, which will be labeled as 'Completed' if reached. Once all available paths have been found, the program outputs a colored map of the matrix that represents the hospital floor plan. The robot's movement is represented in the GUI through colored circles that move between cells. Once the robot stops, the program outputs all completed and failed delivery locations.

Project Functionalities:

- Taking input values from a text file (inputfile.txt)
    - Takes the algorithm, starting location, and a certain number delivery locations

- Format:

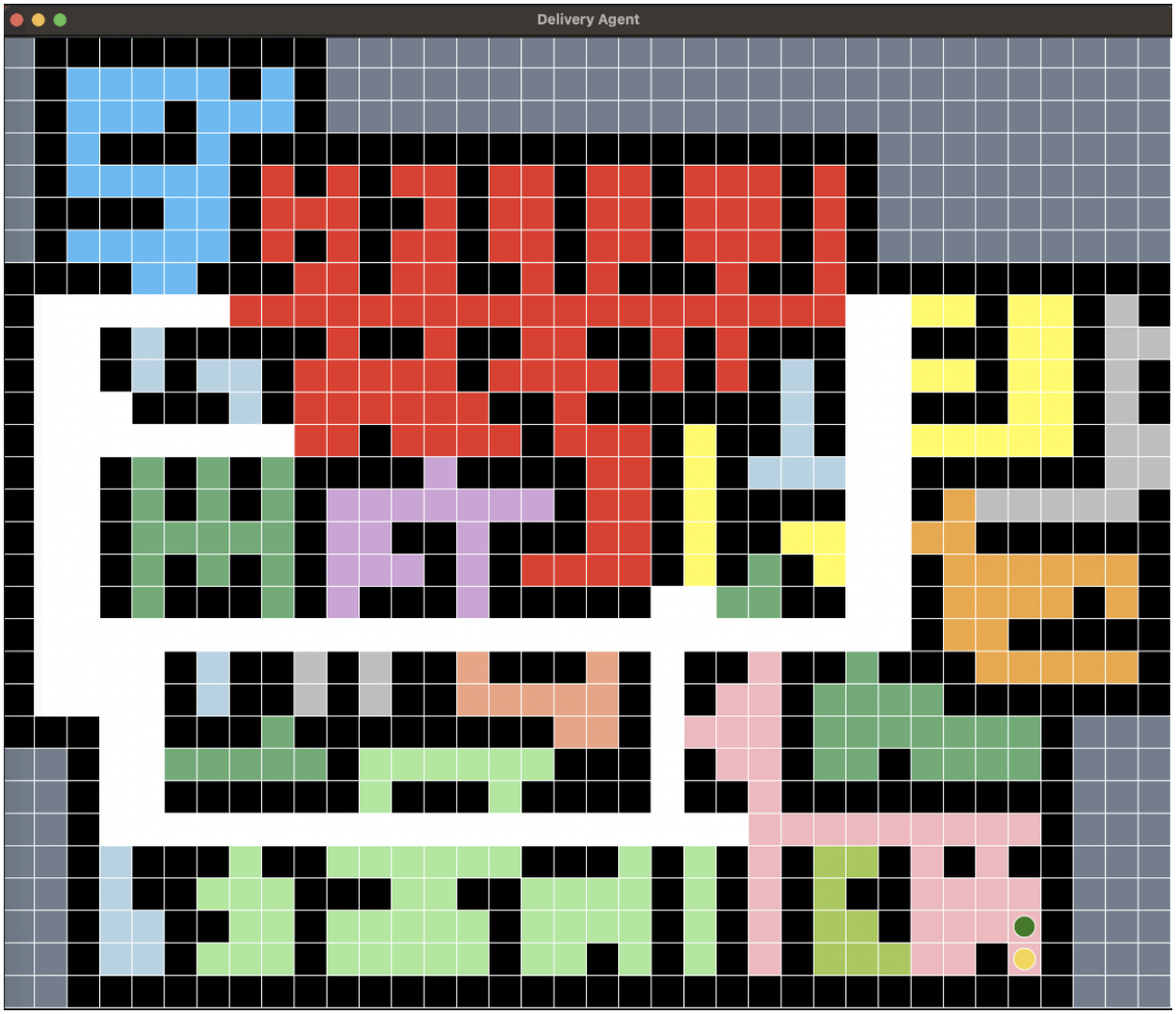
Delivery algorithm: A* or Dijkstra's
Start location: Some location XX
Delivery locations: Location_1, Location_2, ...

- Using A* or Dijkstra's algorithm to find the optimal path towards all delivery locations
  - Goal cost for any movement (N, E, S, W) is 1, no diagonal moves
  - Heuristic cost used for A* is Manhattan distance
- Uses a priority queue for all delivery locations based on ward priority
  - Ward priority used:

Priority 5: ICU, ER, Oncology, Burn Ward
Priority 4: Surgical Ward, Maternity Ward
Priority 3: Hematology, Pediatric Ward
Priority 2: Medical Ward, General Ward
Priority 1: Admissions, Isolation Ward

- Priority of delivery locations changes based on the robot's initial starting location
  - For example, if the robot starts off in the General Ward and there are multiple delivery locations in the General Ward, the robot will deliver to those locations first before delivering to the high priority locations.
- Draw a map of the hospital floor plan based on values in a matrix

- Each value in the matrix is given a certain color,
- Represented in the GUI:

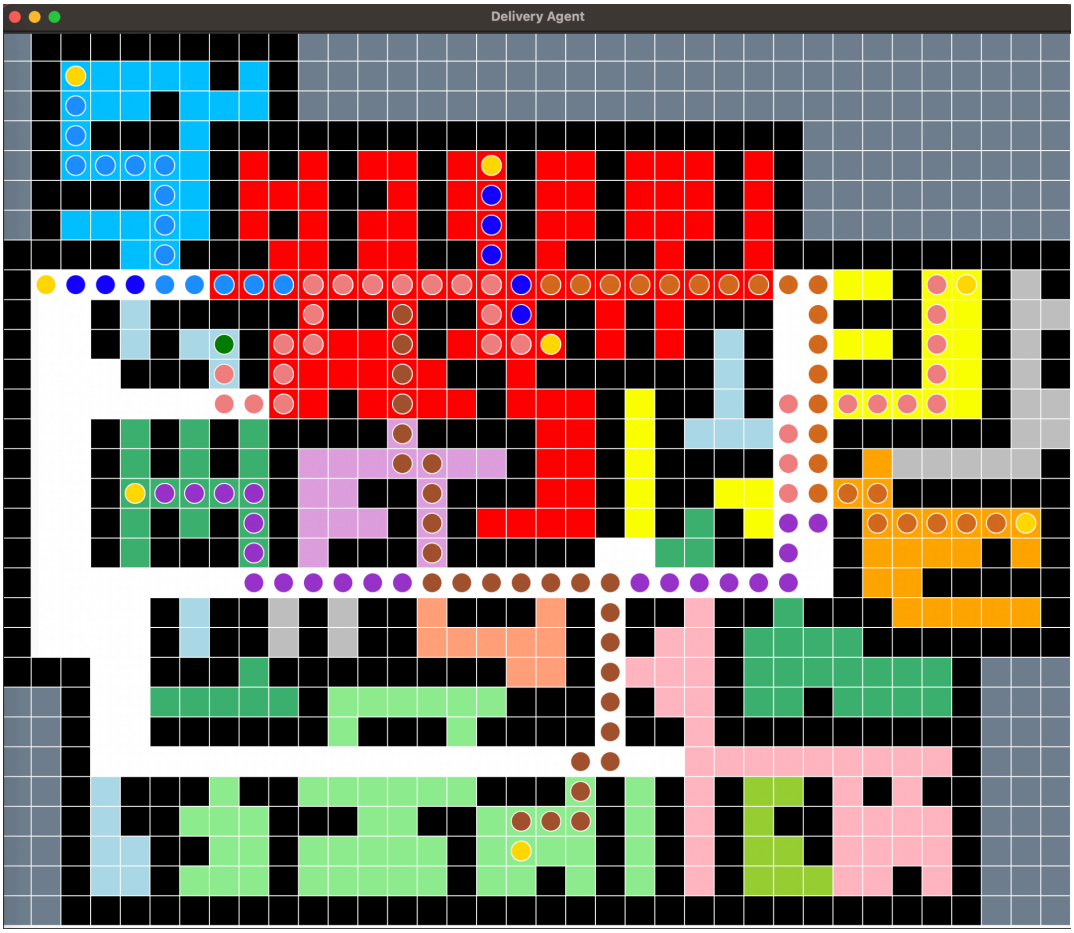


- Draw found paths one step at a time with a delay between paths
- Changes colors for every new path found
  - Uses up to 6 different colors for different paths, switches to a different color each time a new optimal path is found
- Output all delivery goals with a ‘Completed’ or ‘Failed’ label

- Example: Delivery locations: (4,16), (10, 18), (8,32), (0,0), (15,4) (21,5), (10,7), (27,17), (1,2), (16,34)
- Output of program:

```
Completed Goals:
(8, 32)
(15, 4)
(16, 34)
(1, 2)
(27, 17)
(4, 16)
(10, 18)
(10, 7)
Failed Goals:
(0, 0)
(21, 5)
```

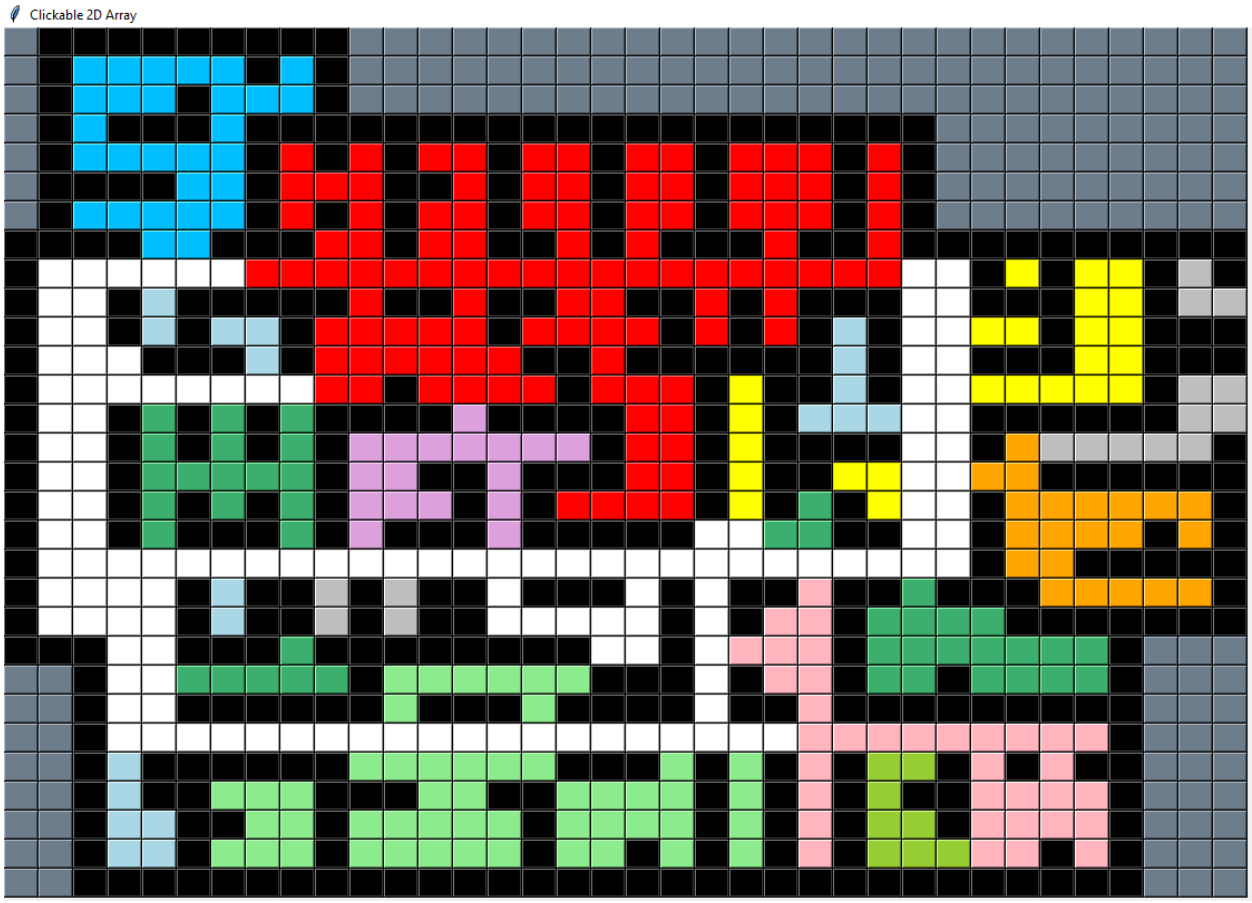
- Result of GUI (Starting location: (8,1), Delivery algorithm: A*)



- Clickable 2d array as well as user inputs for easy use
    - Prompts user to type whether they want A* or Dijkstra, as well as the number of destinations for the delivery locations

```
Choose algorithm - 0 for A*, 1 for Dijkstra (note: anything else will be A*): 0
You chose A* algorithm.
Enter the number of destinations: 4
```

- Once Prompted a clickable 2d array pops up similar to the one that will trace out the path allowing the user to select first a starting point, and then the number of delivery points depending on how many they requested.



- Tells the user in the console what to select and each point selected for what area.

```
Starting Point Selected:
(4, 30)
Select Destination Point(s):
```

```
Clicked on cell (1, 25)
Select Destination Point(s):
Clicked on cell (7, 23)
Select Destination Point(s):
Clicked on cell (12, 22)
Select Destination Point(s):
Clicked on cell (16, 23)
(4, 30)
(1,25),(7,23),(12,22),(16,23)
Data written to inputfile.txt
```

```
Delivery algorithm: A*
Start location: (4, 30)
Delivery locations: (1,25),(7,23),(12,22),(16,23)
```

Statement of Ranking:

Member 1: Michael Wood

My teammate and I agree that I handled **50%** of the overall project. My specific tasks included:

Task 1: I used the code in convert.py to get the initial matrix for the hospital floor plan.

Task 2: I updated the matrix to establish the location of each cell. To do this, I used different values to define which cells would be a certain ward, a wall, or outside of the hospital. This would be properly displayed in the output of the map with colors tied to these values.

Task 3: I helped ensure that the code accepted input values from an input text file. This is based on the given input format provided in the assignment description on Canvas.

Task 4: I updated the function that was used to find a given optimal path if available to accept both the A* and Dijkstra's algorithms.

Task 5: I updated the code and added more functions to find and draw multiple paths to multiple delivery locations.

Task 6: I updated the drawing methods to have different colors for the starting and ending cells of each path.

Task 7: I updated the drawing methods to switch to a different color every time a new path was found. I established up to six different colors for the optimal paths found.

Member 2: Jack Gallagher

My teammate and I agree that I handled **50%** of the overall project. My specific tasks included:

Task 1: I created iteration for the path drawing, adding in after statements to insert pauses

Task 2: Helped with setting up the A* and Dijkstra algorithms as well as setting up if statements to determine which algorithm is being utilized

Task 3: Set up file reading so that inputfile.txt would check for proper formatting and would return the values to variables used throughout the project

Task 4: Created a clickable 2d array that writes out to inputfile.txt to give the user an easier ui when creating starting and delivering points.

Task 5: Performed error checking for the clickable 2d array file that makes sure it limits the clicks to amount of destinations request, requires you to choose either A* or Dijkstra and doesn't allow you to insert an invalid format into inputfile.txt

Task 6: Set up initial classes for cell and maze games, creating base variables that would be needed for the project.

Task 7: Set up the classes for Dijkstra's heuristics and A* star heuristic