Project Plan

Xin Lin (863522)

19.03.2024

General plan

My target level for this project is Hard. The plan is to design and implement a program that stimulates the actions of a robot vacuum cleaner in a pre-defined environment. According to the course project requirement, my program should cover the following:

- an intuitive GUI that will show the robot's movement, the environment, boundaries, and obstacles.
- The robot will detect obstacles of varying shapes and move around seamlessly.
- Areas cleaned by the robot will be marked with certain visual indicators, allowing users to track the progress in real-time.
- The user can customize the map by placing boundaries and obstacles in the environment.
- Implementation of algorithms that reduce redundant movements, prioritize messy areas defined by the users, and complete the task in the least amount of time.
- Implementation of algorithms that include random movement, obstacle avoidance using sensors, and also a pathfinding algorithm.
- Allow multiple robots to be accommodated within the same environment.
- Implementation of collision avoidance mechanisms.

Moreover, some additional features will be included:

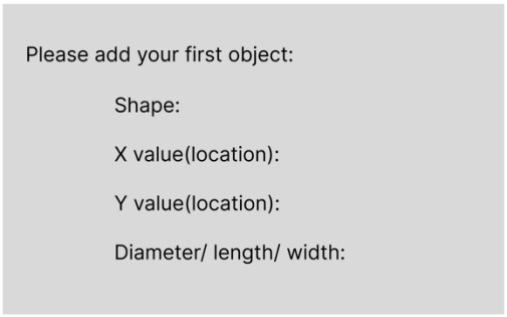
- The robot can detect how dirty is the ground area. If there is dirt left after cleaning, the robot will go back and clean until all dirt is vacuumed.
- When the battery level is lower than 5 percent, the robot will find the most efficient path to navigate itself back to the dock for charging and resume cleaning when it is recharged up to 80 percent.
- When the trash bin is full, the robot will find the most efficient path to navigate back to the dock. The robot will resume its tasks when the bin is reset to empty.
- The robot can detect living things by temperature. It will pause when a living thing is detected in the front, and resume when the living thing goes away.

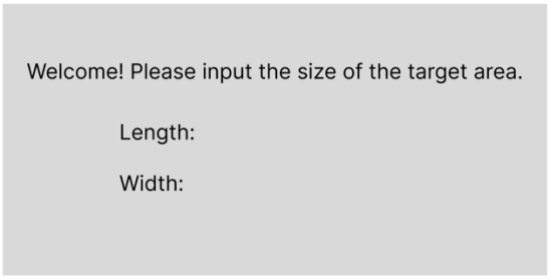
Use case and user interface

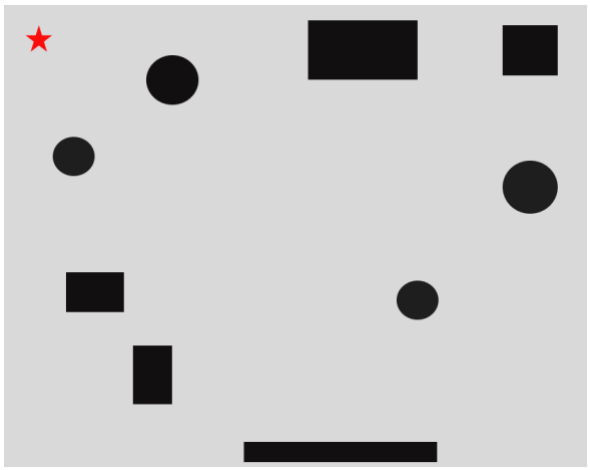
A common use case for this program would be testing the following:

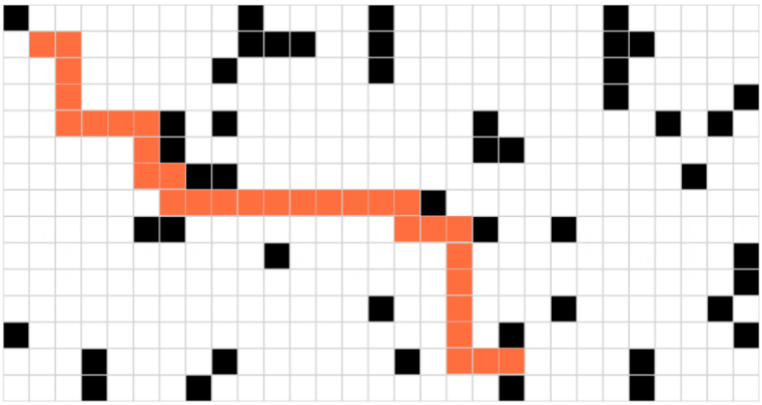
- Simulate how the cleaning robot functions in given environments.
- Compare the efficiency of different algorithms

The program will first ask the user to set up the simulated environment by adding obstacles. Afterwards, a window displaying the defined environment will pop up, just like the picture below.





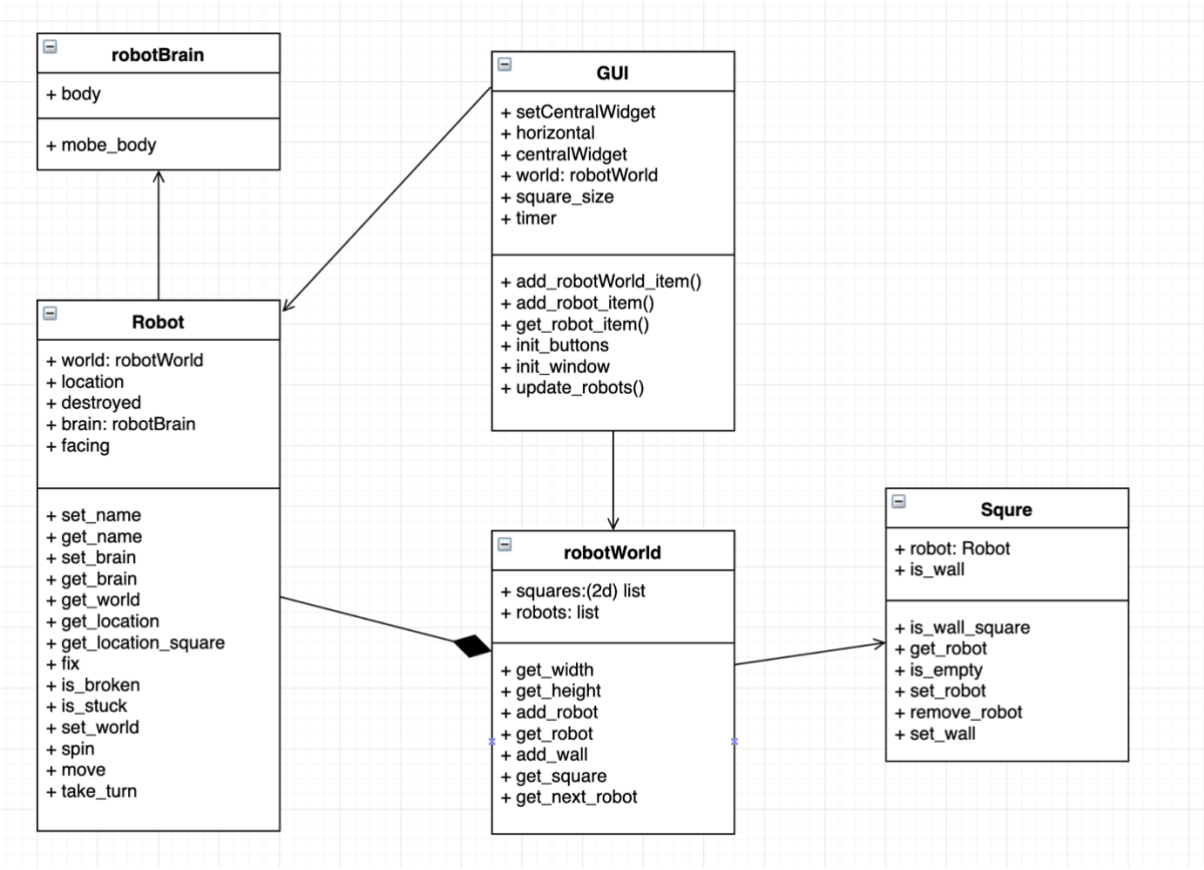




what parts of the UI this is associated with and how) and what parts of the program are activated in each phase ”behind the scenes” and perform the required tasks

Program structure

There will be 5 classes, as shown in the picture below. When I figure out how to implement different algorithms, there might be more classes needed. GUI class takes care of the UI. The Robot class presents robot objects, and the robotWorld class presents the environment. Square class presents square objects that consist of the environment grid.



Data structures

The main data structures will be lists. The environment will be set up as a two-dimensional list. Locations of robots, and obstacles will be presented in x and y coordinates.

Files and file formats

Each robot world grid with robot objects and obstacles can be stored in a text file format. Each row of squares that consists of the world grid can be presented by a two-dimensional array as follows:

0000000000
0000000000
0000000000
0000000000
0000000022
1022020222

Here each number would represent an object:

1. The robot object
2. Occupied square (by obstacle, wall, or another robot)

Algorithms

Path-finding algorithms such as minimum spanning trees, A-star to find the most efficient or shortest path between two points.

A-star algorithm starts by evaluating the starting point and calculating its "score". This score includes the distance from the starting point to the current node, as well as an estimate of the remaining distance to the destination using a heuristic function. Next, the algorithm explores the neighboring nodes of the current node and calculates their scores. It then selects the node with the lowest score and adds it to the list of visited nodes. The algorithm continues this process, evaluating nodes and adding them to the visited list until it reaches the destination node. (Rubio, 2023).

There will also be obstacle-avoidance algorithms such as the Bug or the Potential Field. I will look into details later.

Testing plan

The testing will be mainly done on robot movement, collision detection, and path-finding algorithms. I will also test if the file handling part function as expected.

Libraries

For now I think PyQt is enough.

Schedule

There are still many things confusing to me. I will start planning after the guidance meeting, so that I can estimate better how much time tasks will take.

Literature references

The 5 most powerful pathfinding algorithms < https://www.graphable.ai/blog/pathfinding-algorithms/>

How LiDAR Object detection works https://www.thinkautonomous.ai/blog/how-lidar-detection-works/

How rovots use sensors to avoid obstacles < https://www.linkedin.com/advice/0/how-can-robots-use-sensors-avoid-obstacles-skills-robotics>