Alex Harry
Justin Keeling

Quiz 3

By using our skills from image processing, graph algorithms, as well as data structures, we believe we came up with a solution to finding the shortest path to each landmark, while following the given requirements.

The structure of our program entails multiple classes. One class, Map, is essentially the map which the user provides the robot by walking and marking points within the house. Using a subclass to map, as we call Coordinate, we are able to store the path using a list of connecting Coordinate instances to represent the connections to the robot. This class also contains integers such as x and z to pinpoint specifically in the map where the particular coordinate is.

Another class is needed to represent the "landmarks" (i.e. sink). We call this Destination. Destination has a string name and a Coordinate reference. A method call from the Movement class (mentioned later on) takes in a destination instance to find the best path.

The class to control robots decisions and actions is Movement. Movement contains the brunt of the methods. These methods perform the movements needed to achieve its route to destination. Referencing the slides "RoboticsSLAM," from March 29th, we use A* search algorithm. A* allows us to find the shortest possible path from an initial coordinate to the given goal. The use of A* provides the robot the initial shortest and safest path only using coordinates that are within the Coordinate instances. As this is computed into a tree and movements begin to be executed, we use robot vision to ensure that there are no obstacles between it and its next Coordinate instance it is traveling to. The Room class gives the robot a location in which a landmark is located or the room it is requested to go to. To optimize the path in which the robot will travel, we would implement an algorithm as follows:

> Once A* is computed, we have a variable named 'current' that is the robots current location. There is also a variable named 'next' which is the next location *within the path* that A* computed to reach. The Perception class allows the robot to make short cuts between points in the path if and only if there is no obstacle between the current and next coordinates within the frame. If this is achieved, then a variable named 'mutant' stores this path. Mutant has higher priority, thus the robot will choose the mutant path over the original. If the mutant path is not achievable, then the robot reverts to the original path. This optimizes the initial A* path by creating safe shortcuts to coordinates while still allowing the robot to revert to the A* safe path if necessary.

One class that optimizes the robots decisions (once A* is computed) is called Perception. Perception is similar to the functionality of modern day backup cameras in vehicles. Essentially, the robot has an understanding of other coordinates that may not be within the Coordinate instances. This requires us to use the robots camera FOV (during the development cycle) and calculate accurate measurements corresponding to the space in between coordinate locations. This leads to a ratio to what is also seen in a frame and where the coordinates fit into it. Once this is implemented, the robot can have more insight to the map other than the Map itself. Theoretically, we create a grid where each section of the grid corresponds to an x-z coordinate.

Alex Harry
Justin Keeling

As stated in the quiz slide, the robot does not start on the path. In order for the robot to find the path, we use the Perception class. The Perception class uses the grids within the frame to locate the coordinates of the path in relation to the Coordinate instances. If one is not found, then the robot will turn 90 degrees. If one is still not found once it has done a 360 degree turn, then it wanders. Wanders makes sure that the robot does not repeat the same path twice and ensures it does not run into any obstacles. In theory, the robot will eventually wander onto a known x and z coordinates of a Coordinate instance.

In conclusion, the data will be stored primarily in a undirected graph. The data is organized within the classes and graphs utilizing dictionaries and stacks. We will not throw out data except the previous unused mutation data, the initial data provided allows us with a general safe path to fall back on. We are adding data through mutations that keep track of the shortest safe paths that are found. Making sure that the path is safe and the updates to the Coordinate paths are made need to be pre-processed to the map saving. When somebody says “Go to Sink”, using the robots current position and final coordinates, the robot will try to find more efficient paths, utilizing its vision to make sure that the path is safe to travel. .