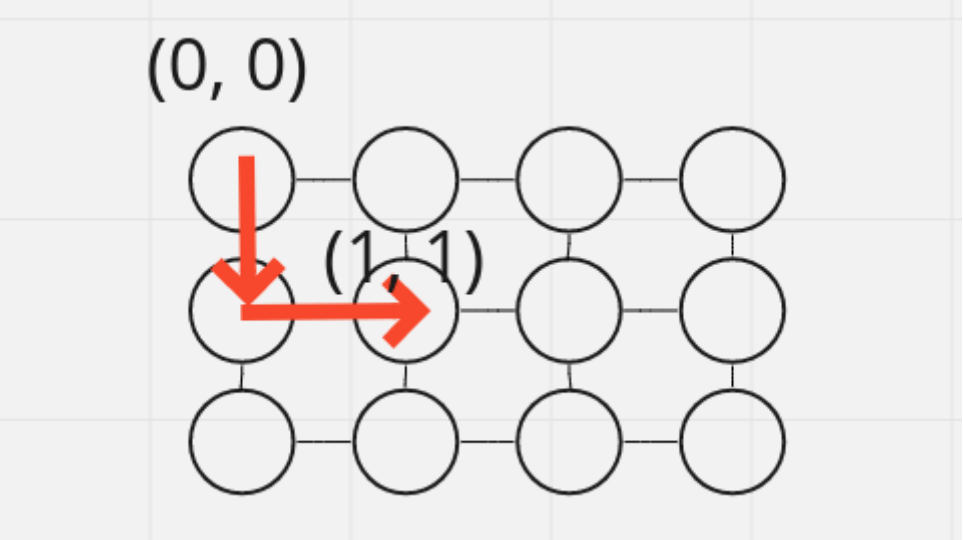
Lab 1

In this lab we have tried to solve the more complex version of the assignment, namely task2.

In this manner we could fulfill task1 and task2 at the same time. Our agent type is a model-based agent and makes decisions based on a sequence of previous percepts. These precepts update the internal state of the agent about the environment. The more the agent explores and observes the environment the more clear and accurate the state becomes.

For exploration of the environment we used Breadth-first-Search (BFS). The agent makes a decision and chooses an action based on the next square to be visited according to the BFS algorithm. For finding the shortest path to these frontier squares we use BFS one more time. In this manner the vacuum cleaner knows the correct path to the next square. We have defined a variable called “goal_square” that tells the agent to react rationally in order to reach the goal square. For instance if we are in square (0, 0) and the next square to be visited is (1, 1) the path might be {(0,0), (0,1),(1,1)} (see picture below). In order to achieve this goal we need to turn the agent's direction to the south (if it is currently not) and then move to south and turn the direction to the east and move forward. When this goal is achieved we update the goal_square again by popping from the queue and repeating the same process in order to visit that square. The end state is when the queue is empty and the goal is to go back to position (1,1) which is home.



We store some data about the partially known environment to build a graph so then we can run BFS on it to find the shortest path between two arbitrary squares. BFS also requires that we keep track of visited squares and therefore we used a list to store all visited squares.
I think there is a more effective way of finding a path between two squares. In our implementation we calculate the shortest path between current square to goal square in each iteration in order to find the next action. This calculation may cost extra time which makes it not necessarily the best approach. If we could somehow store a sequence of actions in a list when we calculated the shortest path between square A and square B we could run those actions without needing to calculate BFS again.

We used BFS because we think that it explores the environment in a more systematic and strategic manner than the DFS algorithm.
In conclusion this model-based agent is better than reactive agent because it has used a BFS algorithm to search the environment and has an internal state which may help the agent for the next cleanup session. Because of using BFS our agent solution is completed which means it scans all squares and cleans up all dirt.
In addition the reactive agent is not able to find the home position which in the model-based agent case, it is able to do so which was made possible again by BFS algorithm. One downside of our agent is that it needs to perform many actions in order to accomplish the goal (cleaning the environment and going back to the home position) but if we evaluate the the performance of these two agents according to the lab performance measurement our agent should outperform the reactive agent.