Task 3:

Main decision loop:

Our decision loop first checks if the agent is on a position with dirt on it, if it is it will suck the dirt.

If there is no dirt it will check if there are any actions that should be completed in the actionsQueue and if there is a point in the path. If there is none in both cases it will get the path and store it in the path state. After that it will be checked if the path is equal to null, which would mean that there are no more points to explore and the agent will shut down.

If there is something in the path state but nothing in the actionsQueue, the agent's position and the last point of the path state will be compared for deciding in which direction the agent has to move to get to the next point. For example: if the agents x-position is smaller than the next point's x-position, the agent has to move eastwards. According to this direction the needed actions to move to the next point will be added to the queue.

If the actionsQueue is not empty it will execute the next action from the queue.

Data stored in the agent:

- actionsQueue: A queue of integers representing actions, in executeAction() it is used to determine which action should be executed
- path: An arraylist of points, calculated by the Breadth-First Search, the last point in the list is the next point on the path to an unexplored point.

Why Breadth-First Search and alternatives

BFS enables us to get an optimal path to a goal-point which is perfect for this example of the vacuum cleaner. It assumes that all the links cost the same. Using Dijkstra instead of Breadth-First, we could have also considered different link costs, if the agent has to turn, as turns are also taken in account for performance.