

**ASSIGNMENT I - Report**

Enhanced Dynamic Robot Movement Simulation

**Course Title :** Artificial Intelligence

**Course Code :** CSE 366

**Semester :** Spring 2024

**Submitted by**

**Name :** [Tasmia Islam](mailto:2021-2-60-062@std.ewubd.edu)

**ID :** 2021-2-60-062

**Section :** 3

Department of Computer Science and Engineering

**Submitted to**

Mohammad Rifat Ahmmad Rashid

Associate professor

Department of Computer Science and Engineering

East West University

**Date of Submission :** 05 March, 2024

**Title :** Enhanced Dynamic Robot Movement Simulation

**Objective :** Design and implement an advanced simulation environment for a robot

navigating through a dynamically created grid. This project aims to deepen understanding of

basic programming concepts, object-oriented programming (OOP), algorithms for

navigation and pathfinding, task optimization, safety, and energy management strategies.

**Overview :** Develop a simulation for a robot moving through a grid-based environment

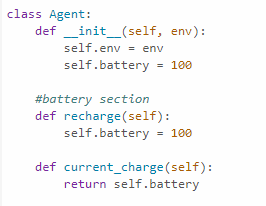
considering task optimization strategies and safety to ensure efficient travel, collision

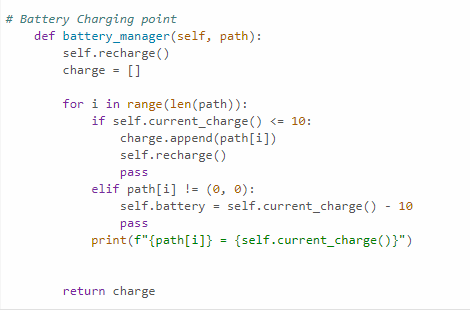
avoidance, and effective energy management. The simulation should manage the robot's

energy levels and battery status, incorporating the necessity of recharging.

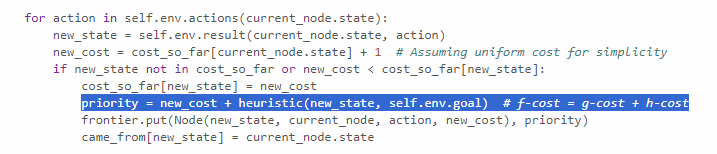
**Summary :**

1. PriorityQueue, Node, and Environment classes are unchanged.
2. But, in the Agent class changes are made; to show the charging level and the position of battery recharge. Initially battery charge is set to 100. As the robot is moving forward to the goal, for every step the charging level is decreased by 10. However, when the charging level is equal to 10 or less the robot recharges the battery and moves forward. This procedure is made in the battery\_manager function.

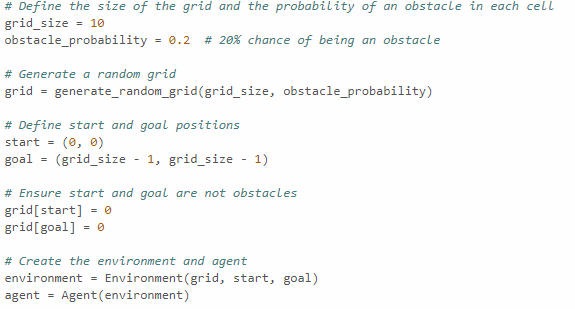




1. To find the optimal solution path two algorithms are used : UCS and A\* search ; here, the difference between these two algorithm is that in the A\* search algorithm h(x) heuristic cost is added with the new\_cost(). In the heuristic function Manhattan distance is used (Euclidean distance can be used).



1. The grid size is 10 X 10 and obstacle probability is 20% .



**Simulations :**

UCS :

This is the solution using Uniform Cost Search algorithm

(0, 0) = 100

(0, 0) = 100

(1, 0) = 90

(2, 0) = 80

(2, 1) = 70

(2, 2) = 60

(2, 3) = 50

(3, 3) = 40

(3, 4) = 30

(3, 5) = 20

(4, 5) = 10

(4, 6) = 100

(5, 6) = 90

(5, 7) = 80

(6, 7) = 70

(6, 8) = 60

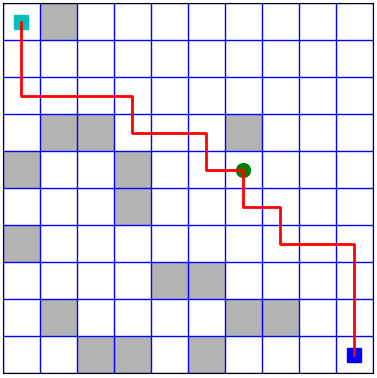
(6, 9) = 50

(7, 9) = 40

(8, 9) = 30

(9, 9) = 20

Charging Point : [(4, 6)]



A\* search :

This is the solution using A\* Search algorithm

(0, 0) = 100

(0, 0) = 100

(1, 0) = 90

(1, 1) = 80

(1, 2) = 70

(1, 3) = 60

(1, 4) = 50

(1, 5) = 40

(1, 6) = 30

(1, 7) = 20

(1, 8) = 10

(2, 8) = 100

(2, 9) = 90

(3, 9) = 80

(4, 9) = 70

(5, 9) = 60

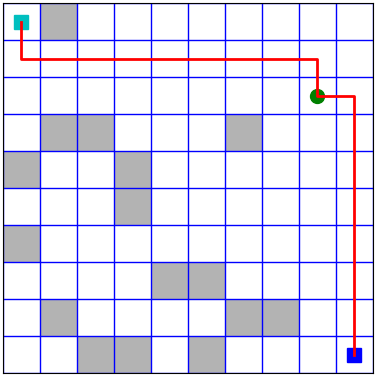
(6, 9) = 50

(7, 9) = 40

(8, 9) = 30

(9, 9) = 20

Charging Point : [(2, 8)]



To use Euclidean distance :

def heuristic(a, b):

(x1, y1) = a

(x2, y2) = b

return math.sqrt((x1 - x2)\*\*2 + (y1 - y2)\*\*2)