INTELLIGENT ROBOTICS ASSIGNMENT-2 REPORT

GROUP MEMBERS-

ARIJIT SAMAL (19051) and KUNAL SHAW (19353)

GitHub links:

ARIJIT SAMAL- https://github.com/1-ARIjitS/INTELLIGENT_ROBOTICS/tree/master

KUNAL SHAW- https://github.com/Kunals909/Intelligent_robotics

Objective:

We have been given a environment where there are 4 circular obstacles and two robots with velocity 0.5m/s and sensing radius of 2 m. These two purple robots follow the carrot chasing algorithm until the path ends. we have to reach the goal from the start location by designing an algorithm.

Robot used:

We use the differential drive robot E-puck for start and 2 purple robots (robot 1 and robot 2 as shown in the figure below).

Approach :

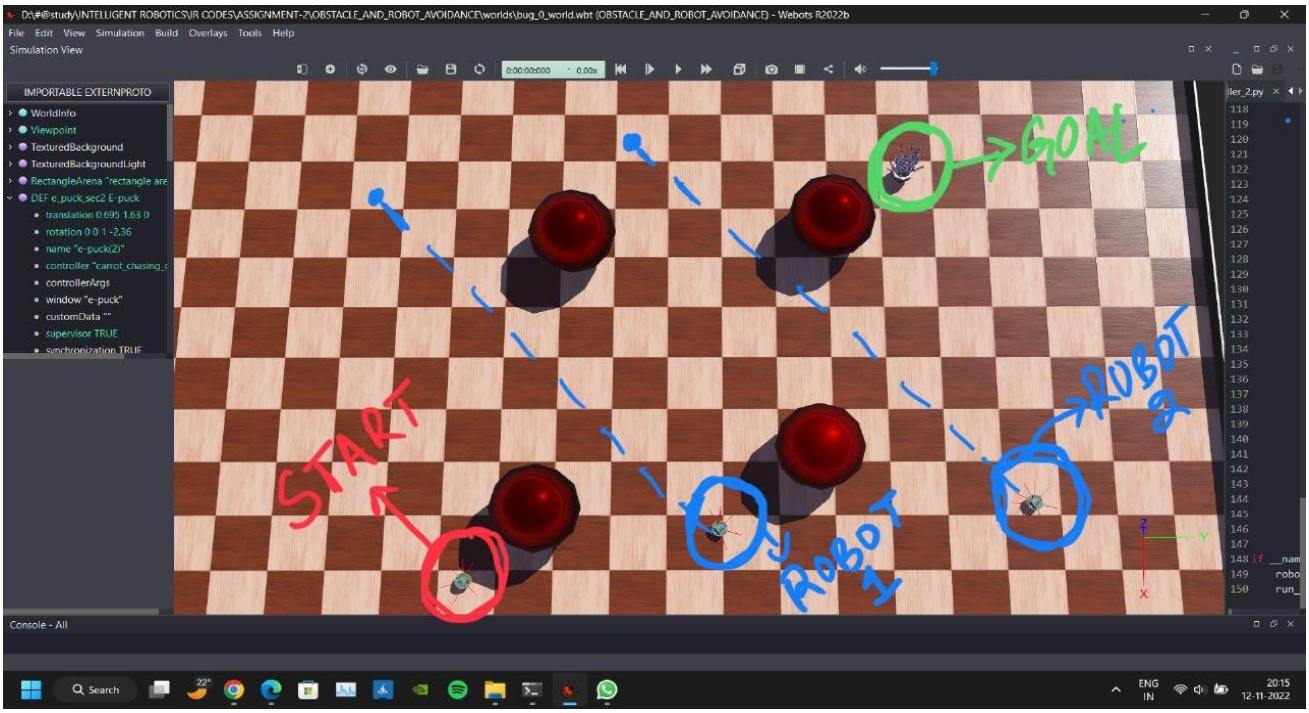
We have come up with a modified version of left following of bug-0 algorithm. Here like the bug0 algorithm we turn left when we sense an obstacle using the distance sensor and move towards the goal when we are not in front of an obstacle.

The modification that we are using here is the sensing radius of the purple robots to avoid collision between the purple robots and the source robot. These two purple robots follow the carrot chasing algorithm on the mentioned path until it ends.

We use two ways to sense the sensing radius. First, we use the current position of the purple robots to calculate the equation of the circle with radius 2 metres and then use it to avoid collision by calculating the distance of the source robot from the circle. Second, we use the Euclidean distance between the source robot and purple robot to avoid the collision.

If we sense the purple robots we use 2 methods explained below to avoid the collision. In First method we turn right as soon as we sense the purple robot and align back towards the goal as soon as we leave the sensing radius. In the Second method we slow down the robot and let the purple robot move out of the sensing radius, when the purple robot have passed we move towards the goal with the original speed.

The bug0 algorithm with following modifications mentioned above is successful in reaching the goal and the final screenshots after reaching the goal are attached in both the methods.

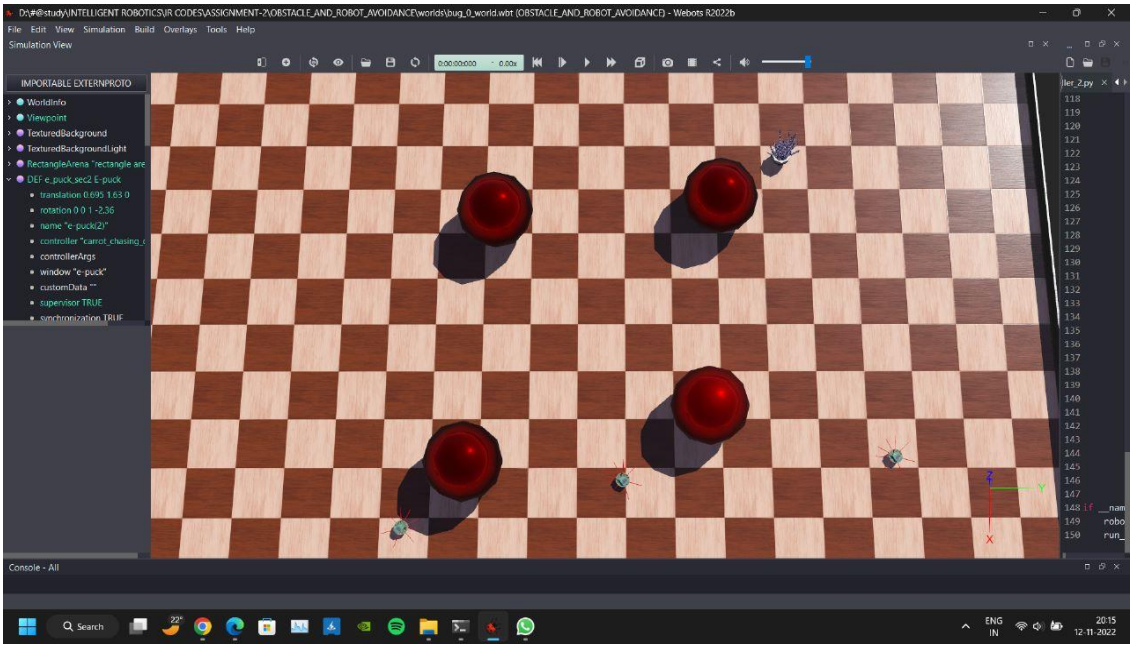


Methods

Method 1(Robot turns around purple robot)

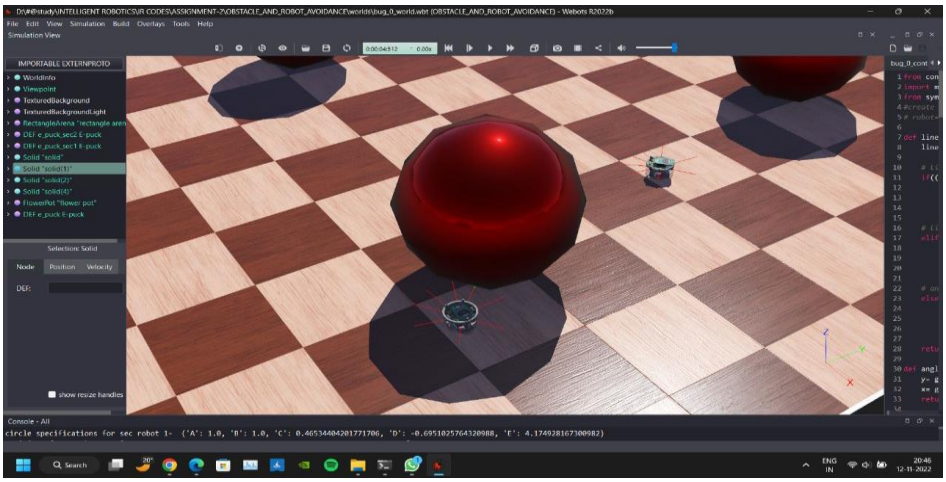
Step 1:

At first robot distance sensor do not sense anything (all the sensing values are less then 80), so the left and right motor moves with max speed in the original direction , so it follows bug-0 algorithm of “towards goal” to move towards the goal.



Step 2: Robot approaches to static Obstacle:

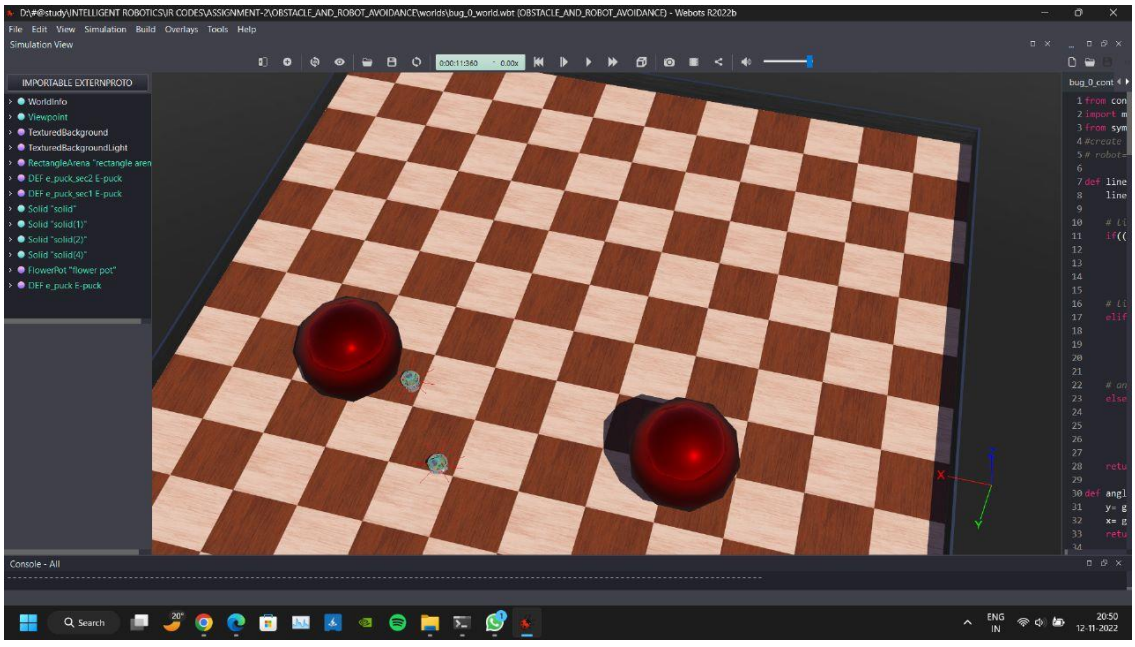
When the robot sensor Ps7 , Ps0 and Ps6 have value sensing value greater than 80, it means there is a obstacle in the front, so it will follow front wall facing algorithm and move towards left.



Step 3:Wall following step:

As the robot move left, the sensor PS-6 and PS-7 have value greater 150, which indicate obstacle is at right. So, the robot will continue to follow the wall of the obstacle.

Step 4:.Robot come near sensing radius:



Robot continuously calculate its distance from the purple robot, to remain away from its sensing radius . To calculate the distance we apply two different approach-

a)Using circle distance method: If coordinate of the robot is inside or on the circle formed by the sensing radius (if the points has radius greater than 2m)of purple robot, then it is avoiding collision and radius of the circle formed by robot coordinate less than 2.

Code-

```
def sensing_radius(source):
    # sensing radius is of 2 m
    x1= source[0]
    y1= source[1]

    circle= {}
    circle['A']= 1.0
    circle['B']= 1.0
    circle['C']= -2*x1
    circle['D']= -2*y1
    circle['E']= (x1**2 + y1**2 - 4)

    #equation of the circle is Ax^2 + By^2 + Cx + Dy + E = 0
    return circle
```

Equation of circle: $Ax^2 + By^2 + Cx + Dy + E = 0$

Here, A=B=1

C=-2*x1 , D=-2*y1 , $E=x1^2 + y1^2 - 4$

X1 and y1 represents the current position of position of the purple robot.

```
line specifications-  {'a': 1.982837089687639, 'b': 1.967730333679993, 'c': -0.34029324453536447}
circle specifications for sec robot 1-  {'A': 1.0, 'B': 1.0, 'C': -1.6248824186921815, 'D': -0.46443442627100745, 'E': -3.2860144472798414}
position of sec robot 1-  [0.8124412093460908, 0.23221721313550373, -2.0865284227631134e-05]
circle specifications for sec robot 2-  {'A': 1.0, 'B': 1.0, 'C': -1.3865896279699965, 'D': -3.249929258155766, 'E': -0.8788322551472851}
position of sec robot 2-  [0.6932948139849983, 1.624964629077883, -2.0865284227174902e-05]
```

Here, we have value for circles for sensing radius of purple robot-1 and purple robot-2 at a particular time. We also show here the line equation for the source robot from source to target.

b) Using Euclidean distance method: if the distance between the robot coordinate and purple robot coordinate. When the distance greater than 2 , we are safe and when it is less than or equal to 2, it is the sensing range.

Equation of line

Code-

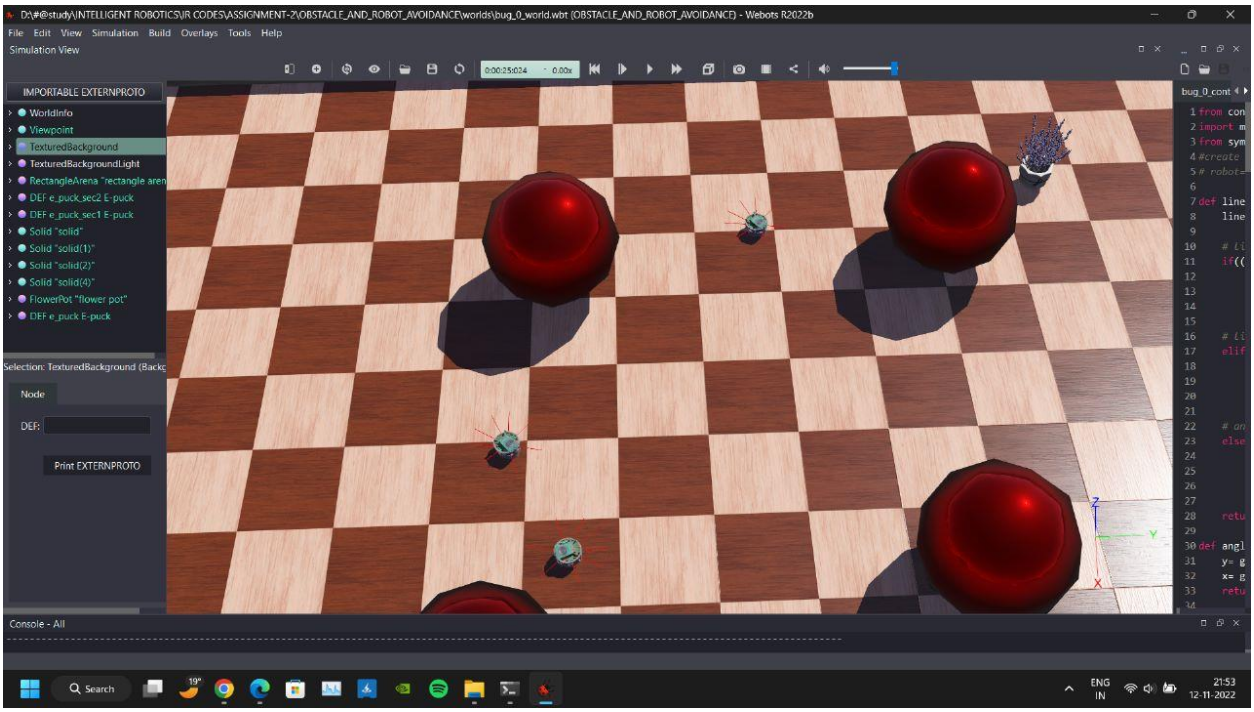
```
def distance(source, goal):
    x= goal[0]-source[0]
    y= goal[1]-source[1]
    dist= (x**2 + y**2)**0.5
    return dist
```

```
distance between source and robot 1-  1.1396082968609 08
distance between source and robot 2-  2.5327246195085573
```

An example of between source and robot 1 and distance between source and robot 2 at an arbitrary time

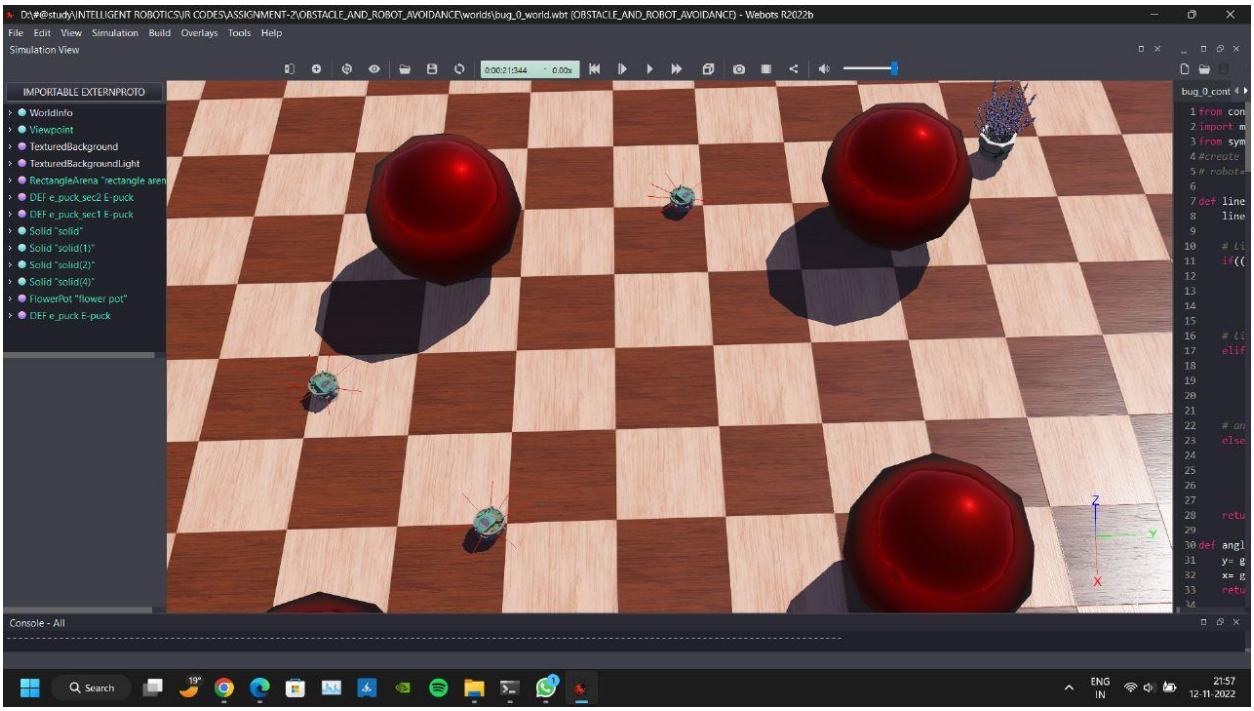
Step 5: Collision avoidance:

When the robot just come near the sensing radius it, change its direction of trajectory. The robot right motor moves in opposite directions and left motor moves with max speed, so that the robot moves in left direction. As the robot starts to move in new direction, the purple robot continue to move as per as carrot chasing algorithm.



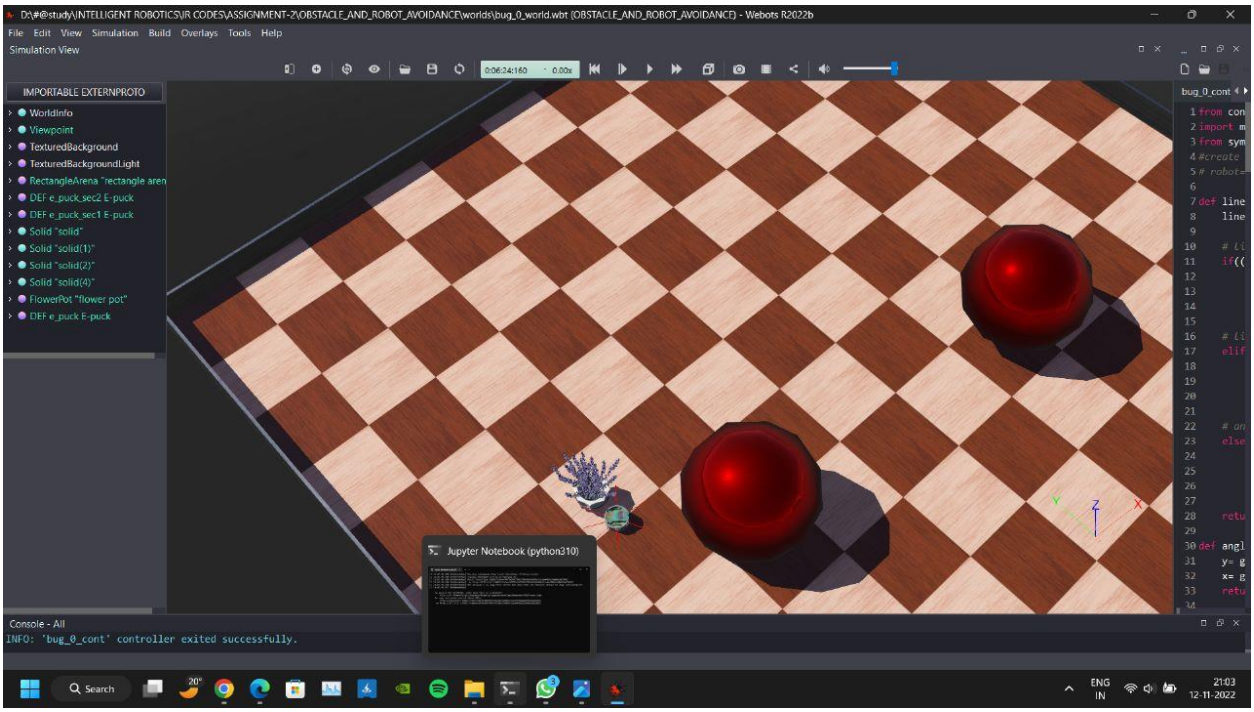
Similarly our robot avoid all the three remaining static obstacle and purple robot-2.

The purple robot stops as they reach their final position in their path as per as their carrot chasing algorithm.



Step 6: Reaching the goal:

The robot continue to move towards the goal as there are no more obstacle. Finally it reach the goal coordinate and our navigation stops.



Method 2:

Key idea- Slowing down the robot speed when the source robot senses any purple robot to avoid collision

All the steps before reaching sensing radius is same as above method.

Step-1: Same

Step-2: Same

Step-3:Same

Step-4:Near sensing radius: Same

Step-5:Collision Avoidance :

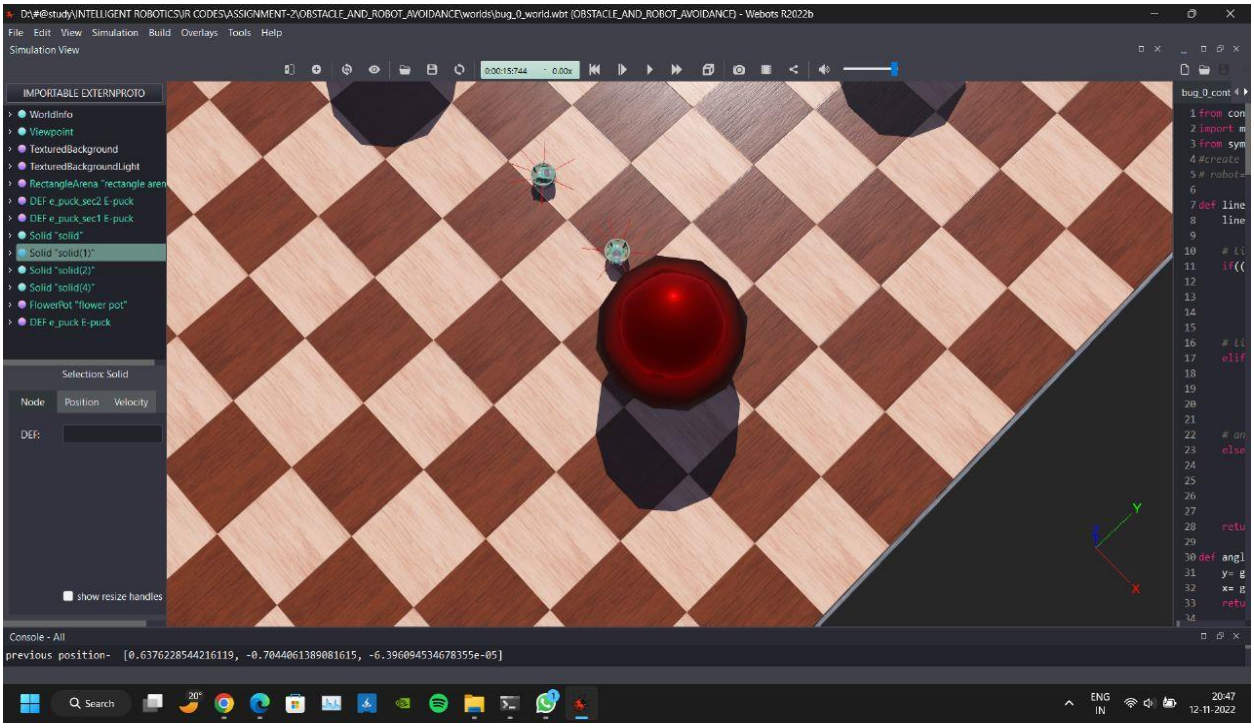
In this method as the robot calculate its distance from purple robot , it follows below steps:

a)The motor of the robot continues to move in same direction but the speed reduced to 0.3 times max speed. The purple robot continues to move in set path

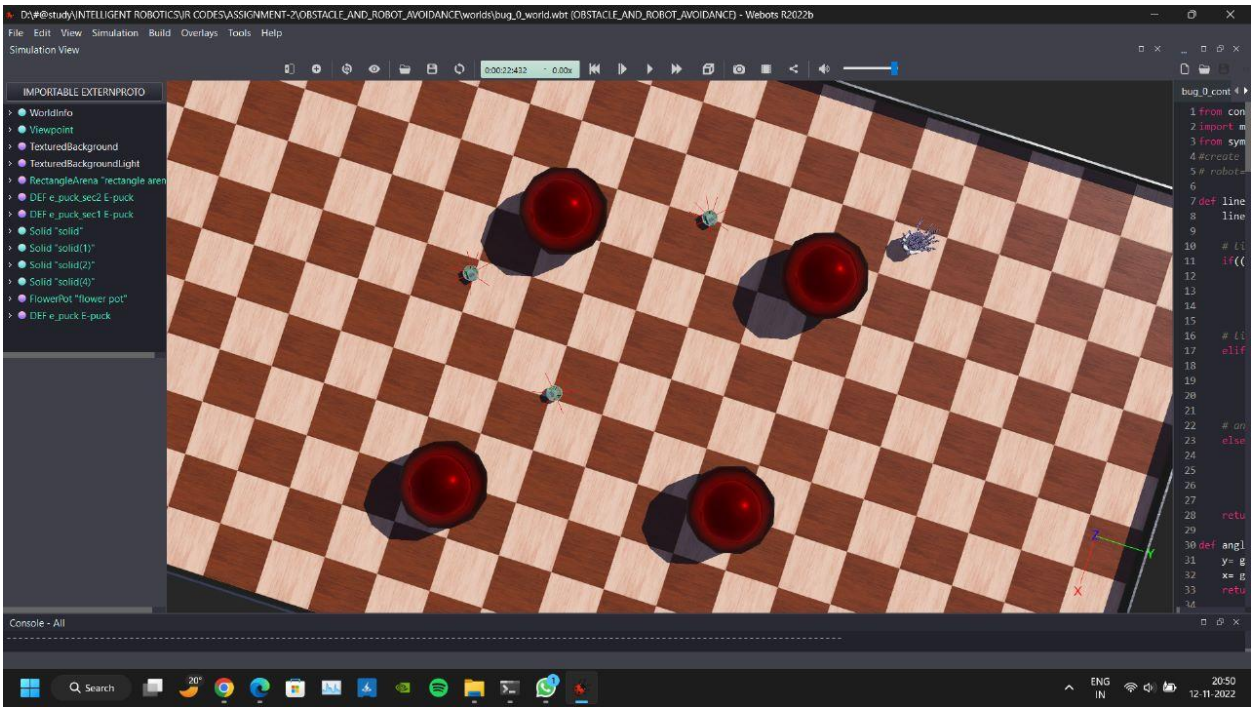
b) As the speed is reduced, the collision is avoided .

c) Due to reduced speed, our robot could easily pass purple robot.

d) When our robot do not sense the purple robot, it regain its original speed in the same direction and continue to move towards goal as per as bug-0 algorithm



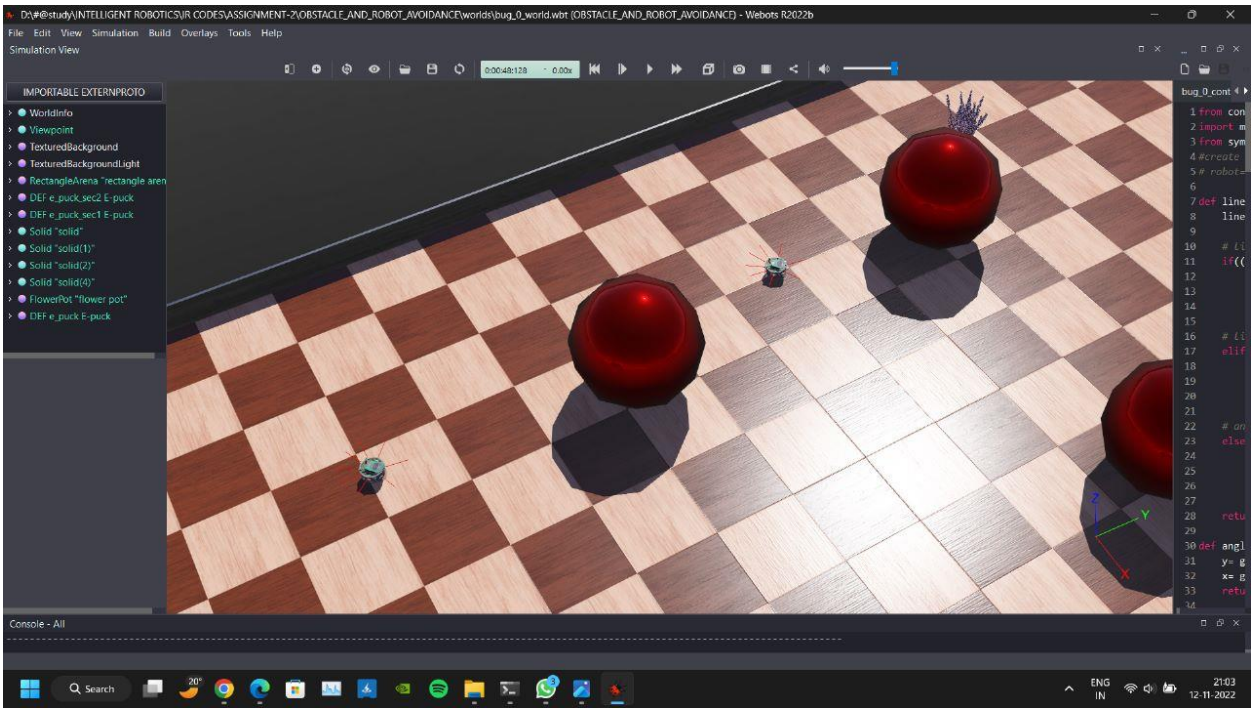
Slowing down when reached near sensing radius



Moving with max speed as the robot leaves the sensing radius of purple robot.

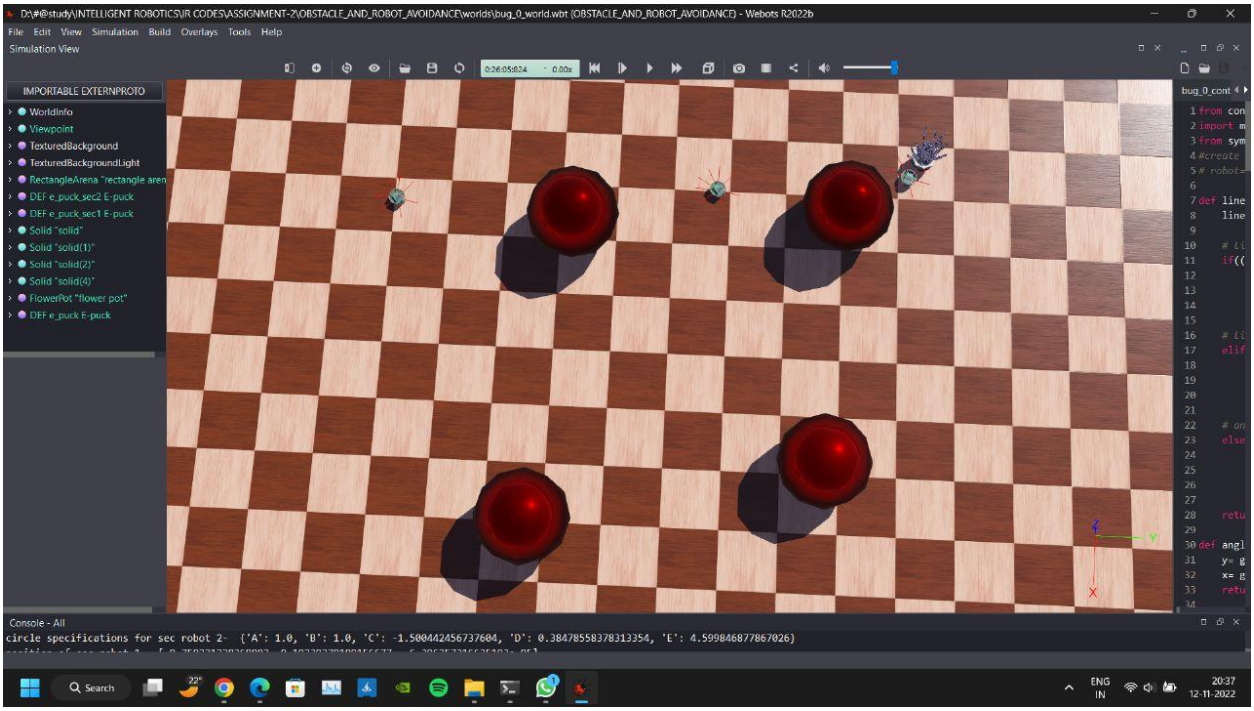
Step 6: Reaching the goal

The purple robots keeps on moving in the fix path as per as carrot chasing algorithm. It finally stops as it reach towards the end of its path.



Step 4 and step 5 are repeated till no more obstacle are present in the path towards the goal.

Finally the robot reach to the goal



Terminal outputs-

```
sensor: ps0 , value: 66.83767316804558
sensor: ps1 , value: 60.77796243616181
sensor: ps2 , value: 57.39907490458216
sensor: ps3 , value: 68.63013710310624
sensor: ps4 , value: 58.89512668185592
sensor: ps5 , value: 69.27001688675018
sensor: ps6 , value: 71.09819316326478
sensor: ps7 , value: 59.88680879021298
line specifications- {'a': 1.982837089687639, 'b': 1.967730333679993, 'c': -0.34029324453536447}
circle specifications for sec robot 1- {'A': 1.0, 'B': 1.0, 'C': -1.6248824186921815, 'D': -0.46443442627100745, 'E': -3.2860144472798414}
position of sec robot 1- [0.8124412093460908, 0.23221721313550373, -2.0865284227631134e-05]
circle specifications for sec robot 2- {'A': 1.0, 'B': 1.0, 'C': -1.3865896279699965, 'D': -3.249929258155766, 'E': -0.8788322551472851}
position of sec robot 2- [0.6932948139849983, 1.624964629077883, -2.0865284227174902e-05]
current angle- -0.7861878970119169
target angle- 2.352370562954506
distance between source and robot 1- 1.139608296860908
distance between source and robot 2- 2.5327246195085573
current position- [1.047730333679993, -0.8828370896876391, -2.0865284227452457e-05]
rotation- [0.02031584920137971, -0.008380663101127638, 0.9997584862141518, 2.3555743302500547]
previous position- [1.0491499999516134, -0.8842590000483106, -2.5517746189968396e-05]
----------
```

Challenges Faced-

During the challenges faced are as follows-

1. How to send the data from one robot to another so that all the robots have knowledge of each other's location. It is required to know the sensing radius of the purple robots to avoid collisions. We fixed that by accessing all the robots common location using the supervisor in the main controller we wrote for the source robot.

2. We also faced challenges while deciding the speed of the robots for turning the robot right when it is within the sensing radius and tending towards a collision.

3. We also faced difficulties in slowing down the robot which later resulted in the faulty detection of obstacles.