1 Project 1 - 5 Points

CSCI 360, Spring 2017
Due Date: February 3rd, 2017, 11:59:59 PM PST

2 Introduction

Moving an agent from a starting state or location to a goal state or location using local and global sensors is one of the most fundamental tasks in Artificial Intelligence. In this project, you will simulate a mobile robot performing this task on a discrete 2D grid using a simple text-based simulator. Below is a screen shot of the text-based simulator you will make use of in this course. Each dot represents a location in the environment that is currently not occupied. The 0 represents the location of the robot in the environment, while the $ represents the target location. The goal is to use the actions and sensors of the robot to move it from its current location to the goal location.

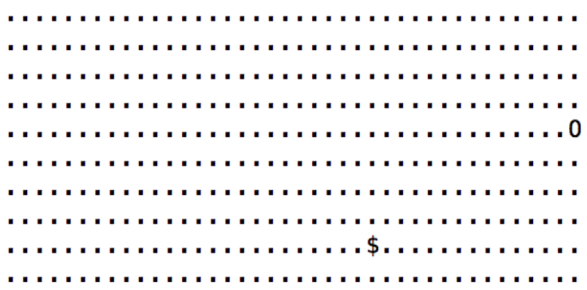

Figure 1: A screenshot of the text-based simulator. The environment is laid out in a discrete grid. Each period represents an unoccupied location in the environment. The robot's location is represented by the 0, while the target location is represented by $

3 Simulator Details

3.1 Wheelbot

In this course, you will be working primarily with a robot called Wheelbot, an abstract mobile robot encoded in the *Robot* class. At each simulation step, Wheelbot can read its sensors and then take one action, moving it one

space in a given direction. For the purposes of this assignment, Wheelbot has the following actions and sensors:

3.1.1 Wheelbot Actions

1. ACTION_UP: Wheelbot moves up one space in the grid: (to use, call $r1 \rightarrow setRobotAction(MOVE\_UP)$)
2. ACTION_DOWN: Wheelbot moves down one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_DOWN)$)
3. ACTION_LEFT: Wheelbot moves left one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_LEFT)$)
4. ACTION_RIGHT: Wheelbot moves right one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_RIGHT)$)
5. ACTION_UP_RIGHT: Wheelbot moves up and right diagonally one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_UP\_RIGHT)$)
6. ACTION_UP_LEFT: Wheelbot moves up and left diagonally one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_UP\_LEFT)$)
7. ACTION_DOWN_RIGHT: Wheelbot moves down and right diagonally one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_DOWN\_RIGHT)$)
8. ACTION_DOWN_LEFT: Wheelbot moves down and left diagonally one space in the grid (to use, call $r1 \rightarrow setRobotAction(MOVE\_DOWN\_LEFT)$)

r1 is an instance of Wheelbot, and it will be provided for you to use in the code you need to modify.

3.1.2 Wheelbot Sensors

1. Target Position Sensor: Returns the 2D position of the target. To use this sensor, call $sim1 \rightarrow getTarget()$, which returns a 2D vector. This is a global sensor, which returns the absolute target location regardless of Wheelbot's position.
2. Target Radiance Sensor: Returns the radiance of the target, which indicates whether or not the robot is at the target location. To use this sensor, call $sim1 \rightarrow getTargetRadiance()$. A value of 0 means that the robot is not at the target location. A value of 1 means that

the robot is at the target location. In future projects, we will extend this sensor to provide more interesting information to the robot.

sim1 is an instance of the simulation environment and will be provided for you in the code you need to modify.

4 Project Details/Requirements

This project will require you to modify the file Project1.cpp in the project source code. **In this file, you will add code to the function getOptimalAction(Simulator* sim1, Robot* r1) You are not to modify anything else in this file or any other file that is part of the simulator**. Feel free to look at Robot.h, which defines Wheelbot, Simulator.h, which defines the environment, and Vector2D.h, which provides 2D vectors and points. We will test your project by copying your code block into our simulation environment in the designated area and running it. Any changes you make outside of the designated area will not be saved or counted as part of your submission. The code that is currently in this function is placeholder code meant to illustrate how to control Wheelbot. This code should be removed and replaced with your own code. Your code must be standard C++ code. You will not need any headers, libraries, etc. outside of what is already provided in the Project1.cpp file.

The requirements of the code you replace it with are as follows:

1. Use the above sensors and actions to navigate Wheelbot from its random starting position to the target position.

2. Let a be the number of horizontal steps between the random starting position and the target position in terms of manhattan distance. Let b be the number of vertical steps between the random starting position and the target position in terms of manhattan distance. Your robot must take no more than $a + b$ steps for full credit.

3. Return (print to the screen) the radiance of the target using $sim1 \rightarrow getTargetRadiance()$. For grading purposes, the value of the radiance will be changed to some positive integer W. If the value of the radiance in your program is 1 when you test it, you can be sure you will get full credit.

For this project, all you need to submit is your modified Project1.cpp file. Submit it to Blackboard.