RoboMind Challenges

Line Following

Make robots navigate by itself

Difficulty: ★★☆ (Medium), Expected duration: Couple of days

Description

In this activity you will use RoboMind, a robot simulation environment, to solve several "line follower" tasks. Robotics competitions often include line follower challenges, where your robot must follow a line on the floor. This activity is a good introduction to the programming tasks one might face as part of a robotics club or team, preparing for competition!

In this activity, you will:

- Become familiar with RoboMind, a robot simulation environment
- Program the robot to solve several different line following tasks
- Develop an understanding of algorithms, and turn your algorithms into code for your robot

Note: there is programming involved in completing this activity. The RoboMind programming language is very simple, however this activity does not include a programming tutorial. If you don't have any programming experience, you might want to try the Getting Started activity first, as it includes step-by-step instructions for programming the RoboMind robot.

Make sure you have the latest software

You need the RoboMind software to complete this activity. If you have done other RoboMind activities, you should already have the software installed on your system. If not, download and install the software that is available on www.robomind.net.

Introduction

In this exercise, you will use the RoboMind robot simulation environment to program a robot to follow lines painted on the floor. Line following is a common feature of some types of robots; for example, mail-delivering robots in an office building often follow lines to get from one office to the next. Because line following is a real-world robotics application, line following challenges appear often in robotics club competitions.

By completing this exercise, you will gain experience with programming a robot to follow lines, and also build skills in algorithm development. Algorithms are a description of the steps one takes to solve a problem; in this exercise, you will create algorithms for your robot to solve each of the maps, and convert those algorithms to code for your robot.

Task 1: Stay on the White Line!

Your first task is simple. Your robot is at one end of a continuous white line, and needs to walk to the other end:

RoboMind Line Follower Challenge, map 1

The first thing you will need to do is open the map that comes with this document: rm-lf-task1.map (rm-lf-task1 stands for RoboMind, Line Follower, Task 1). Pay attention to where you save the map, as you will need to find it when you want to open it in RoboMind.

Once you have saved the map, startup RoboMind, go to the File > Open map menu item, browse to the map, and open it. You will need to be able to download, save, and load maps into RoboMind to do the tasks in this activity.

Once you have the map open in RoboMind, you are ready to begin the task of writing the code for the robot to walk to the end of the white line. If you're not familiar with RoboMind, the dark blue pane in the left part of the main window is the program editing area. You can type your program directly into the editor, and using the menus, save the program to a file.

Once you start coding, note that your robot's initial position has it heading in the wrong direction (north). The first thing you need to do is get your robot heading the right way. By looking at the map you'll see that the white line is to the left (west), and you could get your robot heading in the right direction by simply using the RoboMind `west(n)` instruction; however, see if you can come up with a more general purpose way to get headed in the right direction, by rotating the robot until the white line is in front of it. Once you get your robot heading in the right direction, start moving, and turn when necessary to stay on the white line. Be careful not to bump into the walls, if this were a real robot you wouldn't want it crashing into things! Note that when you get to the end of the white line, there is a beacon in front of the robot. Go ahead and have the robot pick up the beacon as a reward for a job well done.

If you're having trouble with this task, take a look at some of the "See" functions that allow the robot to determine what is around it. Loops and/or If-structures will be needed to control your robot's motion and turning, and also to figure out when to stop.

When complete, save your solution code. In RoboMind, select File > Save as..., and then save the code to a file. Remember where you saved the file, as you will need to submit it when you have finished all of the tasks. Since there are several tasks, you might consider creating a folder where you will save all of your RoboMind scripts for this activity.

Task 2: Debris on the Track

This task is very similar to the first one, however you'll notice that there are a few places where the white line is broken and replaced by a black spot:

RoboMind Line Follower Challenge, map 2

Going back to our example of a mail delivery robot, perhaps some clumsy office worker spilled a cup of coffee on the track, or somebody dropped something and it's covering the line. Can you make your robot smart enough to handle these interruptions in the white line?

To get started, download, save, and load map 2: rm-lf-task2.map

First thing to note, the robot is not at the head of the track (just a little extra added bonus!). The first thing you need to do is get to the start, which is also marked with a black dot. Have your robot turn until there is no obstacle in front of it, and then head forward until the robot is on the dot. Again, you can get there by just doing a `left()` and then a `forward(7)` , but it's a lot cooler if you use the loops and turn repeatedly until there is no obstacle in front, and then go `forward(1)` repeatedly until the robot is on top of the black spot.

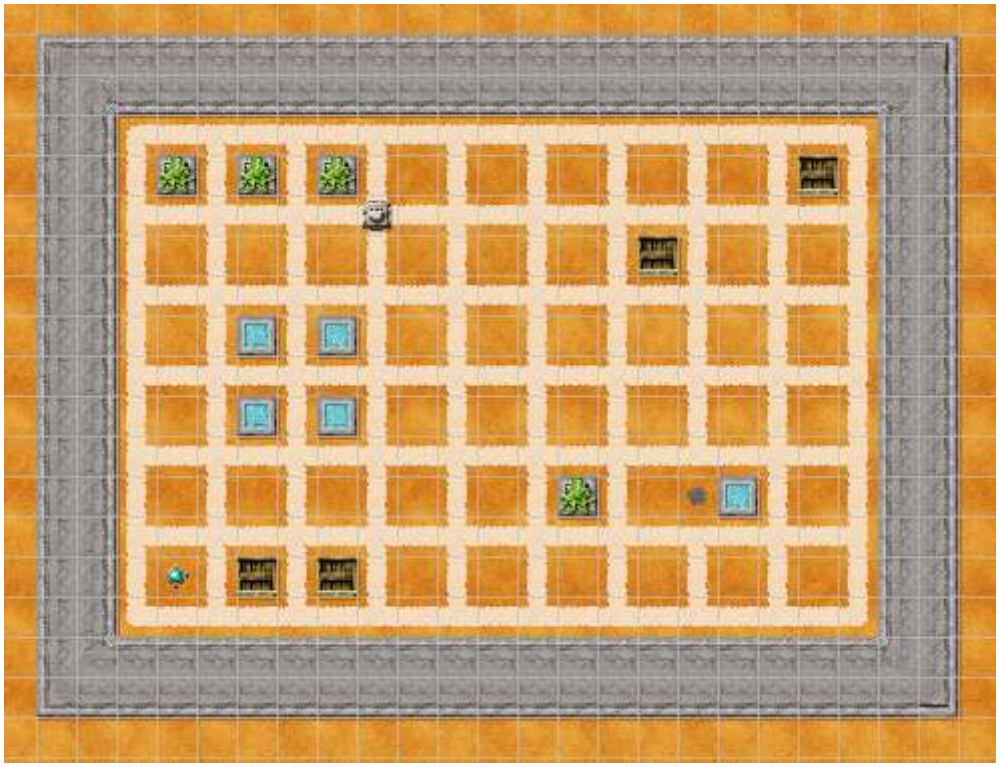
Once your robot reaches the spot, it needs to orient itself so that the white line is in front of it. Make sure you follow that little hook part at the start of the path; no fair just continuing west and missing the first part of the white line!

The code for this task will be very similar to the code for the first task. You just need to add some code at the start to find the black dot, and then handle the black dots that the robot may encounter when walking the white line. Pay particular attention to the black dot in the top-left corner, that one may be tricky, as the robot has to turn to find the white line again. You should probably just start with a copy of your script from task 1, and make changes to it to solve task 2.

When finished, save your script. BE SURE YOU DON'T CLOBBER YOUR SCRIPT FOR TASK 1! Use Save as... to save this script with a different name.

Task 3: Walking a Grid

Now the map gets a bit more complex. The entire map is a grid of white lines, with a black spot at one of the points on the grid. Your task is to walk around, staying on the white lines, and park the robot on the black spot. Have a look:



RoboMind Line Follower Challenge, map 3

Open map 3 to get started: rm-lf-task1.map

Remember earlier we mentioned algorithms? This task is a great example of where you need to develop an algorithm (figure out a solution to the problem) before you start writing your code.

There are (at least) a couple of ways to attack this maze. The first one is to make sure you cover the entire grid. You know that if you walk every white line, you will eventually find the black spot. A second way to look for the black spot is to just wander around randomly; walk on the white line, and every time you get to a decision point (an intersection), make a random choice which way to go. RoboMind has a function called flipCoin() as part of its basic instructions; half of the time it returns true, half of the time it returns false. You could use this function as part of your code to make a random choice.

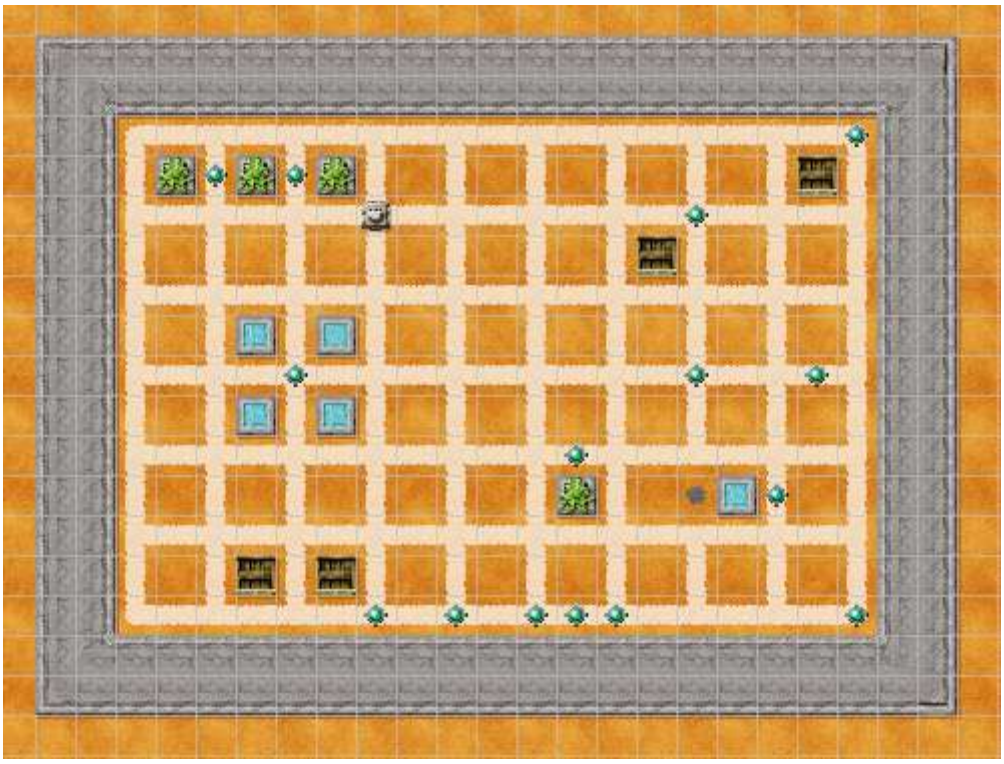
If you aren't using procedures in your code, this might be a good time to look into them. Procedures are a programming construct where you take a list of instructions and group them together, giving them a name that describes what they do. For example, to paint a square with your robot, you would put the brush down, move, turn, move, turn, ... until you had painted all 4 sides of the square. Every time you wanted to paint a square, you'd need to copy this handful of instructions. With a procedure, you'd write all of those instructions inside of a procedure body, give the procedure a descriptive name (like `paintSquare()`), and then whenever you wanted to paint a square, you'd just use the single instruction `paintSquare()`. A procedure might be useful to walk a row or column of the grid, or to make a 3-state (left, right, forward) random choice function using multiple calls to RoboMind's 2-state (true/false) `flipCoin`.

By the way, using "dead reckoning" to find the black spot (looking at the map, counting the squares, and then moving east and then south the correct number of squares) is cheating. Write your code so that your robot can find the black spot no matter where it is placed on the grid.

When completed, pat yourself on the back, because that was hard. Save your work, and move on to the next task.

Task 4: Walking a Mine Field!

The map looks similar to task 3, except this time, there are beacons on the grid. The robot can't walk through the beacons, and should be careful not to bump into them. It's going to be a bit more difficult to get to the black spot this time:



RoboMind Line Follower Challenge, map 4

Open map 4 (rm-lf-task4.map) to prepare for this task.

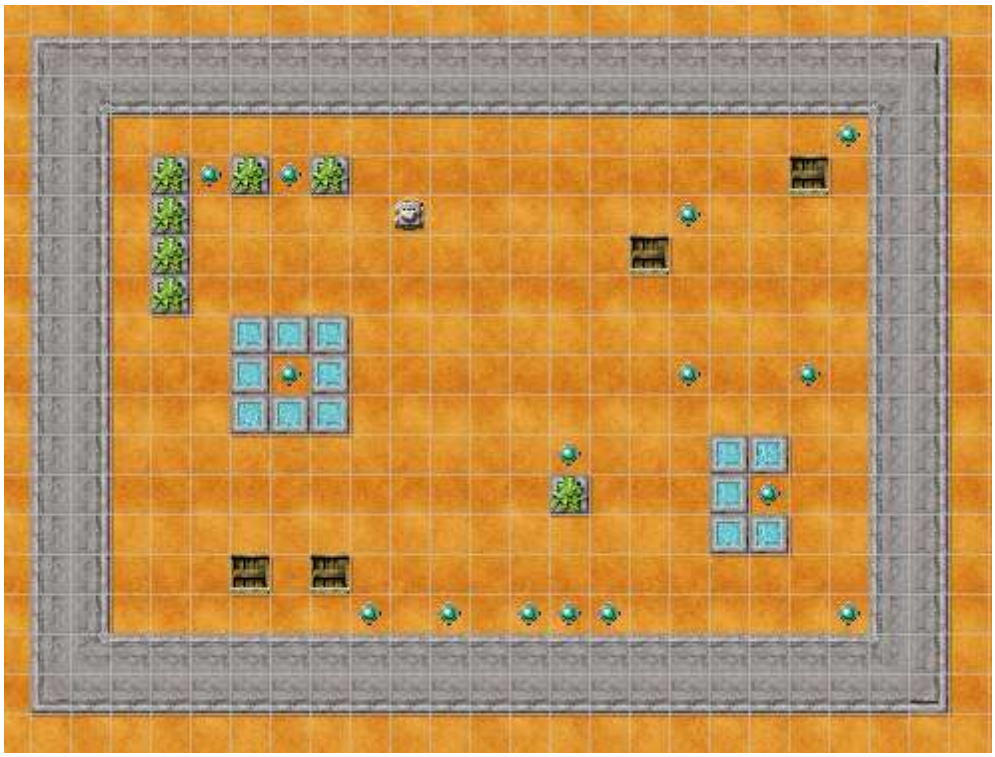
Since this map looks so much like the previous one, you should probably start with a copy of your solution to map 3 and just add to it to address the new problem. This time there are beacons on the grid, and they have to be moved to allow your robot to move around freely. When you find a beacon, pick it up, and put it down somewhere else. The robot can only hold 1 beacon at a time, so you can't just collect them all while you walk around. You can put the beacons anywhere you want, and you don't have to put them down right away, your robot can carry a beacon around as much as you want.

When finished, save your script. If you started from your solution to task 3, make sure you were working with a copy, and you don't clobber your solution to task 3!

Task 5: Hide 'n Seek

OPTIONAL CHALLENGE!

This one is optional. If you enjoyed the first 4 tasks and want a little more, here's something a little different:



RoboMind Line Follower Challenge, map 5

Open map 5 (rm-lf-task5.map) to get started.

The objective is similar to some of the earlier tasks, park the robot on the black spot. You see the black spot on the map, right? Trust me, there is a black spot on the map. It may just be hiding.

No special instructions this time, by now you're an expert. Think about what you want to do, develop an algorithm, code it, and then turn your robot loose and see if it can find and park on the black spot.

Save your work. Congratulations, you've complete the activity!