Assignment 2:

(1) Kinematic Feedback Control, and (2) Braitenburg's "Explorer" Vehicle

---

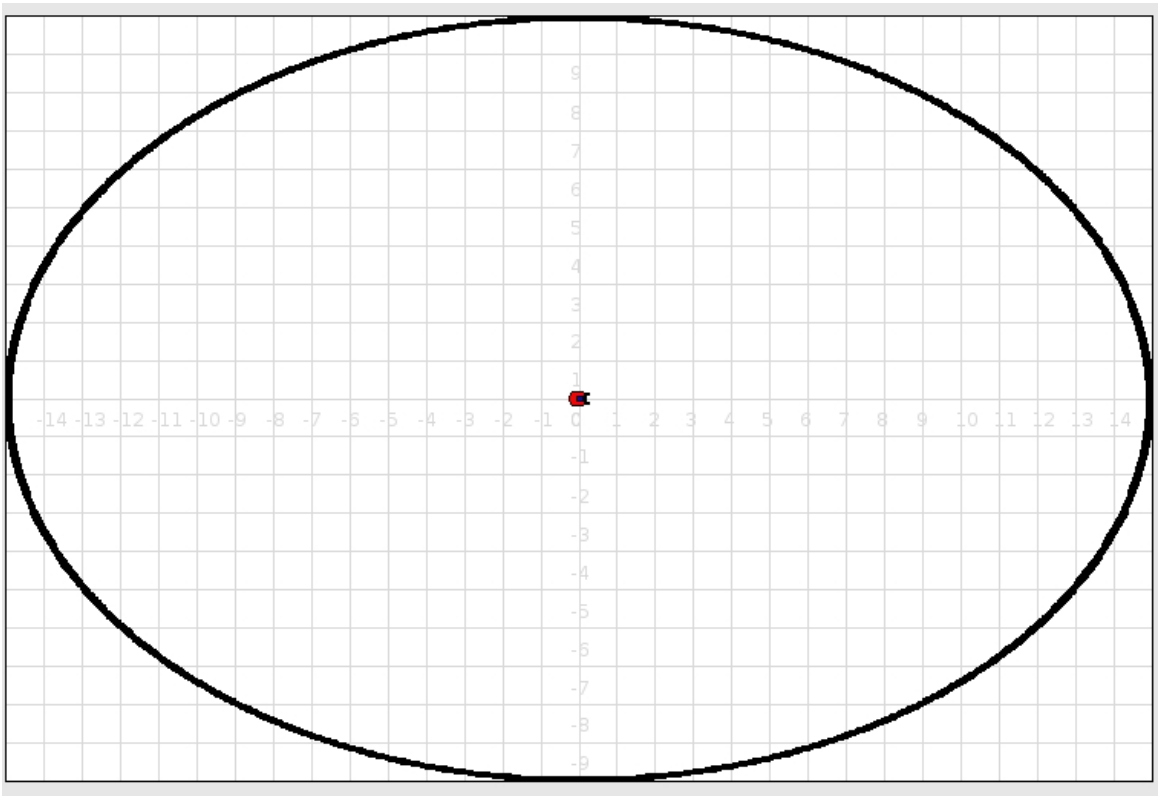
Assigned: Tuesday, Sept. 6
Due: Tuesday, Sept. 20 at 14:10:00
(any homework turned in after the due date/time will be considered late, and will receive a grade of '0')

Step 0: Download the supporting files **HW2-files_3.0.tar.gz** (available on the class website http://web.eecs.utk.edu/~parker/Courses/CS494-529-fall11/Homeworks/HW2-files_3.0.tar.gz and on BlackBoard).

Part 1: Implementing Kinematic Feedback Control

In this part of the homework, you will implement the feedback control approach for a differential drive vehicle, as discussed in Section 3.6.2 of the handout on kinematic control. You will illustrate this control by driving your robot through a series of goal poses (specified below). Your vehicle should stop at each pose, then move on to the following pose, until the last specified pose is reached.

Your robot should be the differential drive Pioneer 2DX robot (same as in mini-assignment #1). In the supporting files, use the .cfg and .world files defined for this part of the project (called HW2-Part1.cfg and HW2-Part1.world). This setup uses the "rink" environment and starts the robot at (0, 0, 0). Here is how your starting simulation should appear:



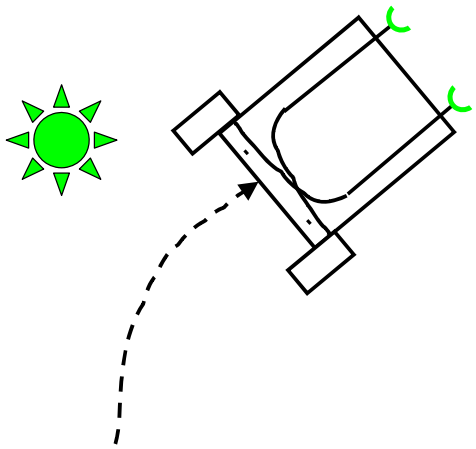
To test out your kinematic control, move your robot through the following series of goal poses; remember, the robot should stop at each pose:

(0,0,0) → (5,3,0) → (2,7,270) → (-7,4,0) → (-7, -3, 0) → (0, -3, 90) → (7, -5, 180) → (0, 0, 0)

What to submit for Part 1:

- A screenshot of the robot trace moving through each goal pose, from the starting position to the final goal pose.

Part 2: Braitenburg's EXPLORER Vehicle (see lecture notes on Aug. 25th, and handout from Sept. 6). In this programming exercise, you will implement the equivalent of Braitenburg's EXPLORER vehicle, which is attracted to a light, but is always exploring (see page 12 of Braitenburg handout). The abstract Braitenburg EXPLORER behavior is illustrated here:



In this exercise, the "light" will be a "fiducial" in Player/Stage. (Note: a *fiducial* is a special marker that can be easily detected with some appropriate sensor. For example, the fiducial could be a reflective barcode that can be detected easily by a laser. Or, it could be a distinctive visual feature that is easily detected by a camera. For this exercise, we don't really care exactly what the fiducial looks like, or what sensor is used to detect it. We'll just use the capabilities provided for this purpose in Player/Stage).

In order for the robot to detect the fiducial, it needs a "fiducialfinder", which is a sensor that detects fiducials (hence the name!). In fact, for this Braitenburg vehicle, the robot actually needs two fiducials, in order to differentially detect the strength of the light. Here, the fiducialfinder returns the range to the sensed fiducial. For this assignment, you should convert this range measure to an equivalent (arbitrary) light intensity. Note that the intensity of light falls off as the square of the distance from the target, whereas range is linear. So, to properly create the Braitenburg vehicle behavior, you'll need to map the range to a light value that falls off as the square of the distance.

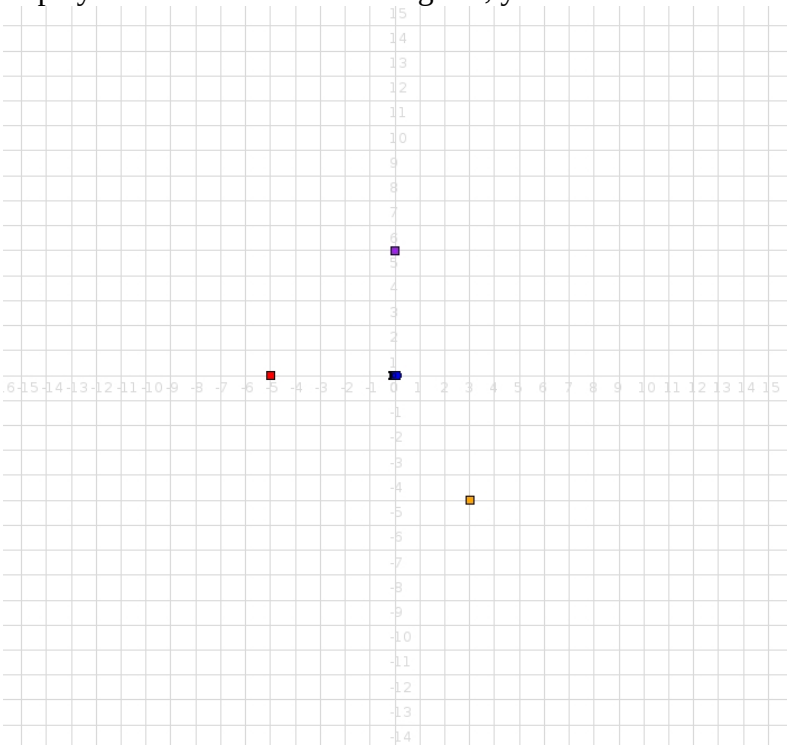
To help you out in setting everything up, we have provided a world model and a configuration file for your use in this part of the homework, called HW2-Part2.world and HW2-Part2.cfg,

available in HW2-files_3.0.tar.gz. Check these files closely to be sure you understand how these new definitions for fiducials and fiducialfinders are added. You are free to make changes to the parameters or beacon locations, as needed to illustrate your robot's EXPLORER behavior.

(FYI: Here's the online documentation for the world file, which includes comments on parameters specific to the fiducials: http://playerstage.sourceforge.net/doc/Stage-2.0.0/group__model__fiducial.html, so that you can understand what the various parameters mean).

In this exercise, we'll use the same "rink" bitmap used in part 2 above. However, so that you don't have to deal with obstacle avoidance issues, we've expanded the size of the environment, so that walls are far away (i.e., you can't actually see the sides of the rink). For this exercise, you should only focus on the Braitenburg "exploration" behavior; you don't have to write an obstacle avoider. A starting point for your code is provided to you in HW2-files_3.0.tar.gz, in the example code "fidRand-2.cc".

If you start up robot-player with the HW2-Part2.cfg file, you'll see a world like this:



What to submit for Part 2:

- Add to the pdf file a screenshot that shows your robot performing the EXPLORER behavior, with a sufficient length of robot trace to make it obvious what behavior the robot is generating. (You'll also be submitting your files that create this behavior; see below.)

SUBMITTING YOUR HOMEWORK:

Place all your files in a single directory. These files should include the following (or their equivalent, if you are using a programming language other than C++):

- Your screenshots and robot output as requested above, in a single pdf file called "yourlastname-HW2.pdf")

- For Part 1:
  - Your configuration file, called "HW2-Part1.cfg"
  - Your world file, called "HW2-Part1.world"
  - Your makefile, called "makefile" or "Makefile" (which is also sufficient for Part 2)
  - Your robot control code, called "yourlastname-HW2-Part1.cc"
  - A README-Part1 file giving the command line arguments to run your code (it may simply be "./yourlastname-HW2-Part1", but if you require command line arguments, please specify them here).

- For Part 2:
  - Your configuration file, called "HW2-Part2.cfg"
  - Your world file, called "HW2-Part2.world"
  - Your makefile, called "makefile" or "Makefile" (which is also sufficient for Part 1)
  -
  - Your robot control code, called "yourlastname-HW2-Part2.cc".
  - A README-Part2 file giving the command line arguments to run your code (it may simply be "./yourlastname-HW2-Part2", but if you require command line arguments, please specify them here).

Remove all other unnecessary/irrelevant files. Tar up all these files into a tarball, compress it, and submit to the BlackBoard website.