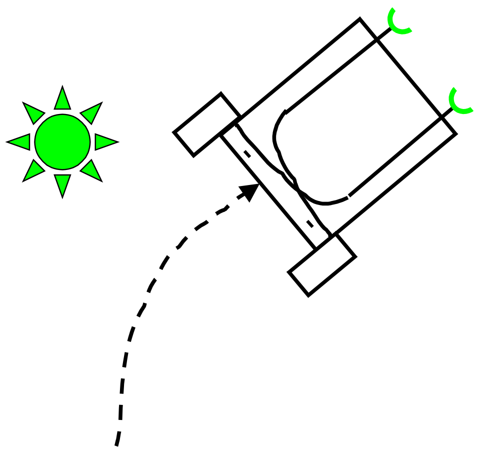
# Homework 5:
*Braitenburg's Vehicles*

**Assigned: Thursday, Oct. 2**
**Due: Tuesday, Oct. 14 at the beginning of class (11:10AM)**

## Part 1: [Everyone] Braitenburg's EXPLORER Vehicle (see lecture notes on Aug. 26th, and additional handout with this homework).

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).

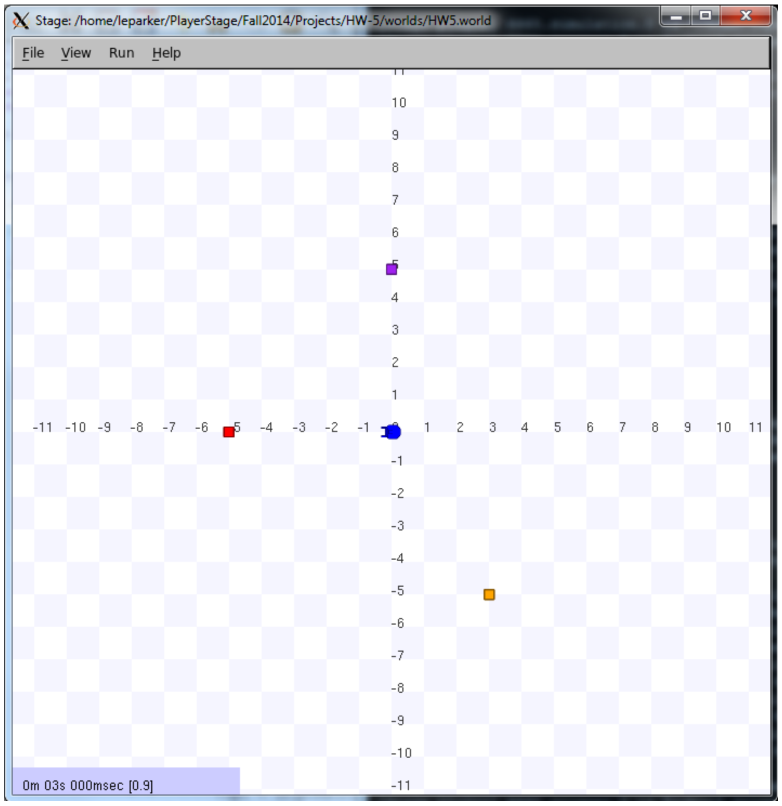
In order for the robot to detect the fiducial, it needs a "fiducialfinder", which is a sensor that detects fiducials (hence the name!). In fact, for this Braitenburg vehicle, the robot actually needs two fiducials, in order to differentially detect the strength of the light. Here, the fiducialfinder 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 HW5.world and HW5.cfg, as well as some basic code to get you started. These files are available in HW5-files.tar. 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.

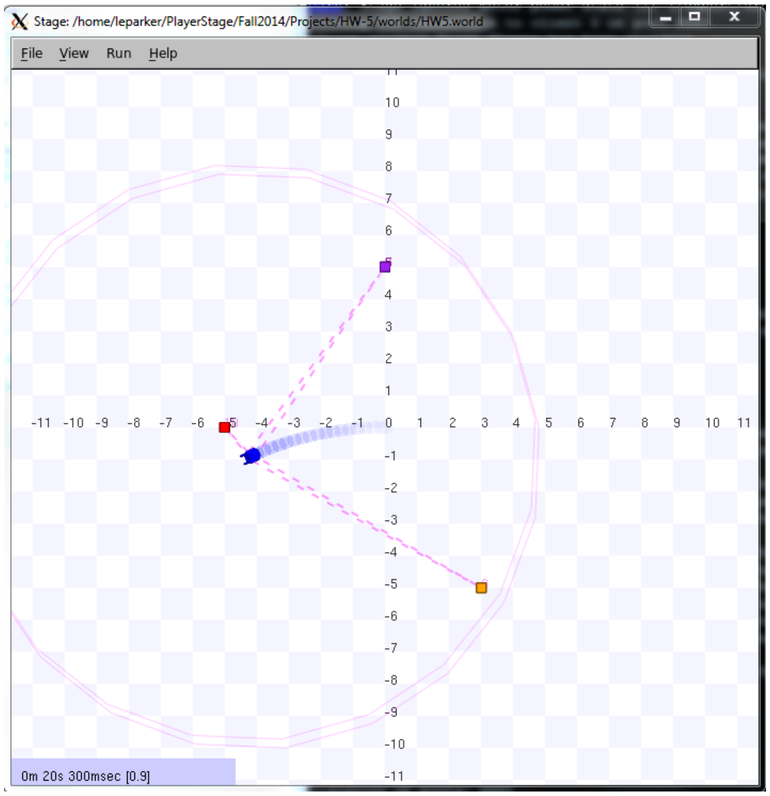
FYI: Here's the online documentation for the world file, which includes comments on parameters specific to the fiducials: http://playerstage.sourceforge.net/doc/Stage-2.0.0/group__model__fiducial.html, so that you can understand what the various parameters mean. Here's the online documentation for the FiducialProxy: http://playerstage.sourceforge.net/doc/Player-1.6.5/player-html/classFiducialProxy.php.

In this exercise, we'll use the same "rink" bitmap used in HW 4. 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.

If you start up player with the HW5.cfg file, you'll see a world like this:


Stage: /home/leparker/PlayerStage/Fall2014/Projects/HW-5/worlds/HW5.world
File View Run Help
0m 03s 000msec [0.9]

A starting point for your code is provided to you in HW5-files.tar, in the example code "fiducial-code.cc". When running this code in Stage, you can also select "Fiducials" and "Fiducial FOV" in the View->Filter data menu, then turn on the "data" flag in Stage (in the View drop-down menu); this will show you the fiducial sensor readings of the robot, and the range of the fiducial sensing of the robot. A sample screenshot when running fiducial-code with the fiducial sensor data displayed is given here:


Stage: /home/leparker/PlayerStage/Fall2014/Projects/HW-5/worlds/HW5.world
File View Run Help
0m 20s 300msec [0.9]

The sample fiducial-code.cc prints out the fiducial readings from each of the two fiducial sensors on the robot. Note that these readings give the (*x*, *y*) positions of the detected fiducials in the robot's frame of reference.

In the code that you write, you should aim for your robot to remain within the field of view of the fiducials, so that it doesn't run off into infinity.

***What to generate:***

- Generate screenshots that show 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.)
- Prepare a ½-page writeup that explains your function that maps the fiducial sensing to the robot's motor control.
- Both the screenshots and the writeup should be included in the same pdf file.

***What to turn in:***

See instructions at the end of this document.

**Part 2: [Graduate Students] Non-Linear Braitenburg Vehicle:**

Create a second Braitenburg vehicle that has a non-linear (perhaps non-smooth) connection between the sensors and motors, as explained in the "Vehicle 4" section of the handout (starting on pg. 15). Your resulting vehicle should have an interesting behavior of some sort, such as shown in Figure 7 (although you are free to create any sort of vehicle with interesting behavior). You should aim for your robot to remain within the field of view of the fiducials, so that it doesn't run off into infinity.

For this part, you will be turning in (1) screenshot(s) of your robot's behavior, (2) a ½-page writeup of how you generated your robot's non-linear behavior, (3) robot control code that generates the resulting behavior. (The screenshots and writeup should be included in the same pdf file.)

## [Everyone] Turning in your project

***Undergrads:***

Place all your files for Part 1 in a single directory. (It's OK to have subdirectories if needed.) 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 for Part 1, in a single pdf file called "yourlastname-HW5.pdf")

- Include all the files needed for running your code:
    - Your configuration file, called "HW5.cfg"
    - Your world file, called "HW5.world"
    - Your makefile, called "makefile" or "Makefile" (in case you changed the standard one)
    - Your robot control code, called "yourlastname-HW5.cc".
    - A README file giving the command line arguments to run your code (it may simply be "./yourlastname-HW5", 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, or create a zip file, compress if needed, and submit to the BlackBoard website.

***Graduate students:***

Create 2 subdirectories; one for your Part 1 materials (call it "Part1"), and a second subdirectory for your Part 2 materials (call it "Part2"). The Part 1 files should include the same materials as mentioned above for the undergrads (in the "Part1" subdirectory). For Part 2 (in the "Part2" subdirectory), include the following files:

- Your screenshots and robot output as requested for Part 2, in a single pdf file called "yourlastname-HW5-Part2.pdf")

- Include all the files needed for running your code:
  - Your configuration file, called "HW5-Part2.cfg"
  - Your world file, called "HW5-Part2.world"
  - Your makefile, called "makefile" or "Makefile" (in case you changed the standard one)
  - Your robot control code, called "yourlastname-HW5-Part2.cc".
  - A README file giving the command line arguments to run your code (it may simply be "./yourlastname-HW5-Part2", but if you require command line arguments, please specify them here).

Remove all other unnecessary/irrelevant files. Tar up all these files (for both Parts 1 and 2) into a single tarball, or create a zip file, compress if needed, and submit to the BlackBoard website.