Design Competition 2014 Workshop 1: Robot Simulation

Milestone 1 due by Wednesday, 2/5/14, at 5pm, by email or demonstration to Nick

Task: Use the iRobot Create MATLAB-based simulator to solve "maze1" and "maze2"

Download:

1. The MATLAB Toolbox for the iRobot Create: http://verifiablerobotics.com/CreateMATLABsimulator/createsimulator.html
2. DC14Workshop1.zip, containing DCexampleControl, maze1, and maze2

How does this thing work?

- Unzip the toolbox, put the files from DC14Workshop1.zip in …\iRobotCreateSimulatorToolbox\Example Files
- Open SimulatorGUI.m and run it
- Click 'Load Map' and select "maze1" from \Example Files\
- Use 'Set Position' to move the robot to the top left corner of the maze, facing down
- Use the 'Manual Controls' to get to the bottom right of the maze. Check out the 'Sensors' live readings

Now what?

- Use 'Set Position' to move the robot back to the top left corner of the maze, facing down
- Under 'Autonomous', click 'Start', and select 'DCexampleControl' from \Example Files\
- 'DCexampleControl' will run the robot around, demonstrating:
    - How to make a simulation run for a limited amount of time
    - How to set the forward velocity and angular velocity of the robot
    - How to read the bump sensor

Dumb dead reckoning

- The simplest way to solve the maze is to drive until you hit a wall, back up a little, turn 90 degrees, and continue, until you get to the end of the maze. "Hard code" an array with turning instructions, because you already know the maze and know you need to turn left, left, right, right, and so on. Every time you hit a wall, look at the array to know what turn you should take.
- Dead reckoning is a method of using velocity and time to determine your position. If you know your heading and velocity, you can predict where you are by integrating (or waiting a certain amount of) time. In simulation, this might work well. But in the real world, noise, slipping and other factors make dead reckoning unreliable.
- Try to edit the 'DCexampleControl' example to use dead reckoning to solve the mazes. Hard code an array with 1's for left turn and 0's for right turn. In the main loop, go straight until you hit a wall. Back up a little, then, using how many hits have occurred since you started as the index to the array, turn left or right for a certain amount of time that seems to do 90 degrees reliably. See if you can finish a maze.

Better dead reckoning

- It is pretty hard to turn exactly 90 degrees with time as your only sensor. If the robot had encoders on the wheels, or a compass, you could "turn until you read 90 degrees" relative to your starting angle.
- The simulated robot has a measure of angle. Use it to replace the time-based turn in your previous code to see if you can do any better than before.

That's it! Don't spend more than a few hours on this. See Nick if you have any problems.

In Workshop 2 we will learn how to use the Makerbot, and continue to improve the maze solving robot with better sensors and algorithms.