Assignment #1 – Reactive Control

Update your COMP203 project from SVN to get the starter code and other files for this assignment.

**Group Challenge: Line Following**

Create a reactive Controller for a Hemisson that will cause it to follow the center brown line around the “Hemmison Barrel Race” course provided.

Your Controller must be called BarrelRacer and must be contained in a package named assignment1.barrel. Include in a comment at the top of your file:

- the names of all group members
- the date
- the name of the Hemisson that was used.

**Grading Rubric:**

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>___ / 1</td>
<td>Controller is named &amp; placed correctly</td>
</tr>
<tr>
<td>___ / 1</td>
<td>Code is well documented and easy to understand</td>
</tr>
<tr>
<td>___ / 5</td>
<td>Remains on center line without intersections</td>
</tr>
<tr>
<td>___ / 4</td>
<td>Correctly crosses intersections</td>
</tr>
<tr>
<td>___ / 3</td>
<td>Smoothness of motion</td>
</tr>
<tr>
<td>___ / 3</td>
<td>Speed</td>
</tr>
<tr>
<td>___ / 2</td>
<td>Finds and follows line from green area inside loop</td>
</tr>
<tr>
<td>___ / 1</td>
<td>Finds and follows line from green area outside loop</td>
</tr>
<tr>
<td>___ /</td>
<td>Bonus: Avoids obstacle on the path and continues on</td>
</tr>
<tr>
<td>___ / 20</td>
<td>Total</td>
</tr>
</tbody>
</table>
**Individual Challenges:**

1. **Obstacle Avoidance:**

Create a reactive controller for a simulated Hemisson (HemissonPlayer) robot that uses the approach of Braitenberg Vehicles 2(a,bc) and will move forward while avoiding obstacles. Your controller must be called BbergV2 and must be contained in a package named assignment1.vehicle.

To run your controller:
- **Launch the dLife ControlCenter:**
  - Open a Terminal
  - cd to dLifeDist/UserScripts
  - enter the command:
    - ./ControlCenter
- **Within the ControlCenter:**
  - Select HemissonPlayer as the robot
  - Select your BbergV2.class file as the controller
  - Click “Start Up”
    - A “Connect to HemissonPlayer” dialog box will appear.
  - Click the “Browse Button” and navigate to:
    - COMP203/src/assignment1/simulation/sims/ObstaclesRoom
  - Select the ObstaclesRoom.dlp file, which provides dLife with a description of the simulated environment and robot.
  - From the dropdown menu select “7000:hemi1 – Hemisson Player”.
  - Click “Connect”
    - After a moment a simulation window should open showing the robot in a room with a few obstacles.
  - Click “Run” to run your Controller.
  - Click “Stop” to stop running your Controller.
  - Click “Shut Down” to disconnect from the simulation.

**Helpful Hints:**
- When the simulation is running you can move the robot around using the mouse. Left click and drag moves the robot. Right click and drag rotates the robot.
- To see changes to your controller you do not need to restart the ControlCenter. You just have to “Shut Down” and then “Start Up” again.
- You can try your controller both with and without sensor noise by clicking the “Enable Noise” button in the “Connect to Hemisson Player” dialog.
- You can create your own simulations using the BuildHemissonSimulation.java class included in the assignment1.simulations package. You’ll just need to produce a map
and edit the paths and sizes in `BuildHemissonSimulation` and then run it. A map is just a simple graphics file (e.g. gif, jpg, png) with black areas representing obstacles/walls and white areas representing open space.

**Grading Rubric:**

___ / 1  Controller is named & placed correctly  
___ / 1  Code is well documented and easy to understand 
___ / 5  Is implemented using Braitenberg approach 
___ / 5  Moves forward when no obstacle is present 
___ / 5  Smoothly avoids obstacles to the front left and front right 
___ / 3  Smoothly avoids obstacles directly in front 
___ / 20  Total 

2. Wall Following:

Create a reactive Controller for a simulated Hemisson (`HemissonPlayer`) robot that will follow a wall with both left (inside) and right (outside) turns always keeping the wall on its right hand side. Your Controller must be called `WallHugger` and must be contained in a package named `assignment1.hugger`.

You can test your controller by running it using the `WallsRoom` simulation included in the `assignment1.simulations` and used similarly to the way you used the `ObstaclesRoom` simulation above.

**Grading Rubric:**

___ / 1  Controller is named & placed correctly  
___ / 1  Code is well documented and easy to understand 
___ / 5  Makes smooth left hand (inside) turns up to 90 degrees 
___ / 5  Makes smooth right hand (outside) turns up to 90 degrees 
___ / 3  Maintains an even distance from the wall being followed 
___ / 2  Speed 
___ / 1  Is able to make right hand (outside) turns up to 180 degrees 
___ / 1  Accomplishes behavior when wall is not initially on right 
___ / 1  Deals reasonably with tight spaces (e.g. narrow dead-end hallway) 
___ / 20  Total