**Robots and the Mind: Homework**

As homework over the weekend, try to make sure your robot avoids obstacles just like we showed in class. The source code for the program we created in class is listed below for your convenience. Create a new class called **Avoider** (it has to have this exact name) and replace its content by the source code below (copy and paste). It should compile and work reasonably well as long as your motors and sensors are connected to the proper input/output ports. You might have to adjust the factor causing a 90 degree turn depending on your robot’s configuration.

Once you get that to work, try the following modifications:

1. **Easy:** Robot should detect and avoid obstacles sooner, i.e. when they are 25 cm or closer.
2. **Easy:** Robot should buzz when it detects an obstacle in addition to avoiding it.
3. **Medium:** Instead of just turning, the robot should buzz, drive backwards a short distance, then turn 90 degrees.
4. **Fun/Medium:** Add a touch sensor to your robot, facing up (say). When you touch it, your robot will react somehow (it could beep, or turn back a short distance, or turn right or left, or whatever you like). Of course it should still avoid obstacles just as before. Note that for a touch sensor, *data*[0] equals 1 if the sensor is touched and 0 otherwise.
5. **Tricky:** Robot should slow down if it detects an obstacle between 10 and 30 cm ahead. If it is closer than 10, it should avoid it and return to its original speed. This is tricky because you are now need a different obstacle detection scheme. Previously you created a function (detectObstacle) that returned one of *two* options: either yes (true) or no (false). Now you need to change this because you need to distinguish between three different possibilities. So, think about it but don’t worry if you can’t make it work, we will discuss this on Monday.

Note that you do not need to turn this in, but we will talk about it on Monday. Please send me email if you have any questions. I highly recommend you try it, including at least attempting the tricky problem. On Monday we will discuss how to create the program we need to solve our challenge, and *that* program will be due either next Wed or Mon, depending on how we are doing.

**Source code listing:**

**import** lejos.hardware.Button;

**import** lejos.hardware.Sound;

**import** lejos.hardware.motor.EV3LargeRegulatedMotor;

**import** lejos.hardware.port.MotorPort;

**import** lejos.hardware.port.SensorPort;

**import** lejos.hardware.sensor.EV3UltrasonicSensor;

**import** lejos.robotics.SampleProvider;

**public** **class** Avoider

{

// the fields for the left and right motors

**public** **static** EV3LargeRegulatedMotor *leftMotor* = **new** EV3LargeRegulatedMotor(MotorPort.***A***);

**public** **static** EV3LargeRegulatedMotor *rightMotor* = **new** EV3LargeRegulatedMotor(MotorPort.***D***);

// the fields I need for a sensor according to the "sensor framework"

**public** **static** EV3UltrasonicSensor *distanceSensor* = **new** EV3UltrasonicSensor(SensorPort.***S1***);

**public** **static** SampleProvider *distanceProvider* = *distanceSensor*.getDistanceMode();

**public** **static** **float**[] *data* = **new** **float**[*distanceProvider*.sampleSize()];

// the standard method to get the distance from the sensor

**public** **static** **double** getDistance()

{

*distanceProvider*.fetchSample(*data*, 0);

**return** *data*[0];

}

// Method to drive the robot forward

**public** **static** **void** driveForward()

{

*leftMotor*.forward();

*rightMotor*.forward();

}

// Method to avoid the obstacle by simply turning by 90 degrees

**public** **static** **void** avoidObstacle()

{

**int** degrees = (**int**)(2.7\*90);

*leftMotor*.rotate(degrees, **true**);

*rightMotor*.rotate(-degrees);

}

// Returns true if it detects an object less than 15 cm ahead, false if path is clear

**public** **static** **boolean** detectObstacle()

{

**if** (*getDistance*() < 0.15)

{

**return** **true**;

}

**else**

{

**return** **false**;

}

}

// The main method. Should \*always\* be simple and basically just call

// the other methods (in the right order) to do all the work.

**public** **static** **void** main(String[] args)

{

// enter a loop that is stopped by touching the ENTER button

**while** (Button.***ENTER***.isUp())

{

*driveForward*();

**if** (*detectObstacle*())

{

*avoidObstacle*();

}

}

// done (someone pressed the button) - beeping just for fun

Sound.*beep*();

}

}