CMSC 143: Object-Oriented Programming with Robots

Lab 4: Robo-Cockroach
Due March 8, 2014

In this lab, we will create an autonomous robot creature; we'll turn the scribbler into a robot cockroach. Your goal is to create a robot program that will run as long as possible without any intervention — to make the robot autonomous. You should use the behavior-based approach outlined in the first ten pages of chapter 7 of the textbook. You can add as many levels of behavior as you like, but at the very least your cockroach should:

1. Scurry about randomly (using `randomNumber()` which returns random number between 0–1).
2. Avoid running into things (using the IR sensors: `getIR()` or `getObstacle()`, or `getStall()`).
3. Run away from light (using the light sensors: `getLight()`).
4. Allow a user to drive the cockroach with the gamepad (see next section).

Each robot behavior should be implemented as a separate function. That way we are able to add and remove each level of behavior easily (i.e. you should not create one loop with a bunch of if-statements). You should develop your program one behavior at a time. After each level is completed, you should write a paragraph (as a multi-line comment) describing how it works and how well it works. For example, from Chapter 7:

```
from Myro import *

cruiseSpeed = 0.75
turnSpeed = 0.5

def cruise():
    return True, cruiseSpeed, 0

def arbitrate(behaviors):
    for b in behaviors:
        activated, t, r = b()
        if activated:
            return t, r

    return 0, 0

setForwardness('scribbler-forward')

behaviors = [cruise]

for t in timer (20):
    t, r = arbitrate(behaviors)
    move(t, r)

stop()
```

The Gamepad

The `gamepad()` function (from the lab #1) makes all the gamepad buttons do something interesting (e.g., move the robot, beep, speak, or take a picture). In this lab, you will use the gamepad to control the robot. For moving the robot with the gamepad, Myro's `move(translate, rotate)` function could be handy.

First, become familiar with `getGamepadNow()`, figure out how the different buttons work and the order they are reported. The `getGamepadNow()` returns the status of the gamepad (represented as something called a dictionary). If you ask for the `'button'` entry you get a list of eight items: `[0, 0, 1, 0, 0, 0, 0, 0]` each item corresponding to one of the buttons. In this case, the third button is pressed. Similarly, the `'axis'` entry returns the status of the directional pad (as a list with two items): `[-0.99824, 0.0]`.

What are the differences between the directional pad on the left and the buttons on the right? Experiment with the following code, explain in a comment what you find.

```
while True:

    buttons = getGamepadNow("button")
    print ("The buttons:", buttons)
    print ("The first button reads", buttons[0])

    axes = getGamepadNow("axis")
    print ("The x-axis reads", axes[0], "and the y-axis reads", axes[1])

    wait(.5)
```

Learning Objectives

- ○ Program Robot Behaviors ○ Employ Incremental Development ○ Use Lists

Deliverables

`cmsc143_lab4_LASTNAME_FIRSTNAME.py` – Your cockroach program.