CMSC 143: Introduction to Object-Oriented Programming with Robots

Lab 3: Personalized `gamepad()`

Due September 24, 2016

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 create your own `myGamepad()`. You can make the buttons do whatever you wish, for instance, report the battery voltage, save a picture to a file, or play different musical notes. For moving the robot with the gamepad the `move(translate, rotate)` function could be handy.

Be creative.

Lab Report

Submit an electronic copy (using moodle) and a hard copy (during the first class meeting following the due date) of your lab. Your program should have your name, email, assignment description, the date, and collaboration statement at the top of the file as a comment. Your electronic submission should be a zip file that expands to a folder with a single file:

```
cmsc143-lab3-LASTNAME-FIRSTNAME/
      lab4.py
```

Learning Objectives

- Create a personalized gamepad
- Use while loops
- Use conditionals
- Understand Flow-of-Execution

Some Gamepad Details

For the first part of the assignment, become familiar with `getGamepad()`; figure out how the different buttons work and the order they are reported.

The `getGamepad()` function waits for a button to be pressed and then returns the status of the controller (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): `[-0.99996, 0.0]`. In this case, the x-axis of the directional pad is pressed left. To retrieve an item from a list you can use the index operator. For example, if we have the result of `getGamepad()` in a variable named `status`, then `status['button']` grabs a list representing the values of the buttons and `status['axis']` grabs a list representing the values of the axes.

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:

    status = getGamepad()
```

```
    print ("The entire status of the gamepad looks like", status)

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

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

Another function named `getGamepadNow()` immediately returns the status of the gamepad without waiting for a button press. However, this will read a button press over and over again, as long as the button is pressed. This function is useful for driving the robot with the directional pad.

myGamepad()

Once you are comfortable with how `getGamepad()` and `getGamepadNow()` work you should implement your own `myGamepad()` function. You can make the buttons do whatever you wish, for instance, increase or decrease the beeping frequency, or report the light sensor values. Or maybe make pairs of buttons do some action.

There are two requirements:

1. the north most button should stop your function (i.e. `keepGoing = False`);
2. use all of the other buttons.

Your program should start something like:

```
def myGamepad():
    ''' HOW TO USE MY GAMEPAD
         up button:  EXPLANATION
       down button:  EXPLANATION
                  ...
    '''

    keepGoing = True

    while keepGoing:

       status = getGamepad()         # or status = getGamepadNow()
       buttons = status['button']
       axes = status['axis']
```