Name:____________________________________ Period ___ Date:________

Lab 70-3 Electric Forces using Python (simple game). V2.1

Important: Some of these steps should be done only once. Don't do them again when you continue the lab after logging out and logging back in again.

- Get a LabJack "game controller" board. Connect it via USB to your laptop.
- Firefox > halverscience.net > Python Coding > Python Program Files > ezu3.py Save and move to your my_python folder (Do once)
- Firefox > halverscience.net > Python Coding > Python Program Files > ezu3test.py Save it and move it to your my_python folder (Do once)
- Run Terminal

  cd Desktop (Do every time after you log in.)

  cd my_python (Do every time after you log in.)

  python ezu3test.py (You should get a stream of data coming over the terminal.)
- **Expand the terminal window** (make it wide) to see clearly what happens.
- Verify that the sliders and game buttons work OK.
- Firefox > halverscience.net > Physics - Halverson > Python for Physics > electric_charge_game0.py Save the file and move it to my_python (Do once.) Run it in the terminal to see that it works. (It's basically the black hole code, slightly modified. I renamed the black hole to BIG_Q and the ship is now small_q)
- cp electric_charge_game0.py electric_charge_game1.py (This makes a copy and now you will modify the copy) (Do once)
- edit electric_charge_game1.py (Do every time after you log in.)
- edit ezu3test.py
- Copy from ezeu3test.py the lines from "import sys" to "u3setup(d,['ain'....." (lines5 to 18) to the start of your main program. (Put it just above the line that says "tk = Tk()")
- Copy from eze3test.py the two lines that read the slider voltages. They say

```
V0=ain0(d)      #Read analog inputs
 V1=ain1(d)      #Analog inputs range from 0 to 2.4 Volts
```

  and paste them just after the line that says "while keep_looping:"
- We are going to have the sliders control the location of the black hole, except that now the black hole will become a positive charge. The code to do that will be:

```
BIG_Q_x = V0/2.2 * window_width
BIG_Q_y = V1/2.2 * window_height
```

  ----- Insert these two lines after the previous two lines (V0=... and V1=...)

This works because the sliders produce a voltage that ranges from 0 to 2.2 Volts. So if V0 is zero, then the hole_x value will be zero and it will go to the left side of the window. If v0 is 2.2 Volts, then the hole_x value will be equal to window_width and it will go the the right side.

1. When you have the hole moving under the control of the sliders, get a **stamp** for credit.

At this point there is a lot of unnecessary code in your program. Anything related to the BIG_Q acceleration and velocity is no longer needed. I recommend that you clean it out.

2. Modify the code so that the small charge (the square) is repelled by the large charge (the round thing). Get a **stamp**.

3. Modify the code so that when the small charge hits a wall, it bounces off with 1/2 its original velocity.

As an example, here is code for when it hits the right side.

```
if small_q_x >= window_width:    # >= means “greater than or equal to”
   small_q_x=window_width     #Move the small_q to the edge (prevent it from getting trapped)
   small_q_vx = -small_q_vx*0.5    #Keep 1/2 of the original velocity
```

There are FOUR walls, so you will need FOUR “if small_q ....” type statements. Keep in mind that the bottom wall is at y=0, the left wall is at x=0, the right wall is at x=window_width and the top is at y=window_height.

------These if statement should be added near the end of the loop. A good place put put them would be just before the “locate” statements.

When the bouncing works, get a **stamp**.

Try using the repulsion of the charge to push it into the upper right corner of the window.

Thinking question:

How would you modify the code so that it prints out “YOU WIN” when you get the charge in the upper right corner?