day93 micro:bit accelerometer
Tuesday 5/3/22 (6th period) Wednesday 5/4/22 (7th period)

Today we're going to dive deeper into micro:bit's accelerometer functionality.

Last assignment we accessed gestures, that is, recognizable patterns that the accelerometer detects. Today we're going to dive deeper into the world of specific positions using x and y coordinates, that is, measurements of how slanted the board is to the left/right and front/back.

A very simple program that accesses the x and y information is here:

```
from microbit import *
while True:
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    if button_a.was_pressed():
        display.scroll(x)
    if button_b.was_pressed():
        display.scroll(y)
```

The above code reads the x and y values from the accelerometer in an infinite loop. To see the x value at any moment press the A button. To see the y value press the B button. The micro:bit documentation says that the x and y values will be +/- 2000, but in my testing I found it perfectly functional to assume that the values would be +/- 1000.

Now comes some Image magic. You can create an empty, blank image by putting this line BEFORE the while True loop:

```
i = Image()
```

You can then set any pixel (any single light) on or off using the following a few commands:

```
i.set_pixel(col,row,9)
display.show(i)
sleep(500)
```

In the above code **col** needs to be an integer from 0 to 4 for the column and **row** is 0 to 4 for the row. The 9 is the brightness (0 is off, 9 is full brightness). The sleep is just to make sure you see the light before whatever happens next.

To turn that light off you would do these commands:

```
i.set_pixel(col,row,0)
display.show(i)
```

The 0 turns that light off (9 is on full, 0 is off).

(continued on next page)

Today's task:

Write a program that lights up dots positioned using the x and y coordinates. Turn it into a blackout game, where you tilt the board to move the dot around until all of the pixels are light up.

To do this you will need to map the coordinates as follows:

```
x or y smaller than -600, set to 0
x or y bigger than -600 and smaller than -200, set to 1
x or y bigger than -200 and smaller than 200, set to 2
x or y bigger than 200 and smaller than 600, set to 3
x or y bigger than 600, set to 4
```

You can do this with an if/elif statement as follows:

```
if x<-600:
    col=0
elif x<-200:
    col=1
elif x<200:
    col=2
elif x<600:
    col=3
else:
    col=4
```

The reason this works is that Python will only execute one of the if/elif statements, in fact, the first one that is true for any given x value. So when x is -500 it sets col to 1.

Make a parallel structure for y and a variable **row** and you end up with two variables, row and col that are between 0 and 4 inclusive which you can use with the i.set_pixel(col,row,9) call to light up a single pixel. Sleep for 500 milliseconds after each dot gets turned on.

Put this all together and you'll get a game where you try to light up all of the pixels on the board, one at a time until they are all lit.

Make it so pressing the A button wipes the board. You can do this with an if statement and then a line like this:

```
i = Image()
display.show(i)
```

This just resets the image named 'i' to be a new blank image.

Before you try to do today's program, remove the other code from page one that showed the x and y values when you pressed A and B (that was just to help you understand how things work.)

For extra credit, figure out when the whole grid is lit up (I used the i.get_pixel(x,y) call between turns on all of the LEDs to see if they were all on or not) and scroll "you won!" and then start over.

Alternate extra credit: start with all pixels lit up and have the moving pixel turn OFF pixels. Or do that after they are all turned on… or think of something else really cool and call me over to see it.