Rahma Ahmed 9/21/17

Cancer Cell Random Motion

Background/Introduction:

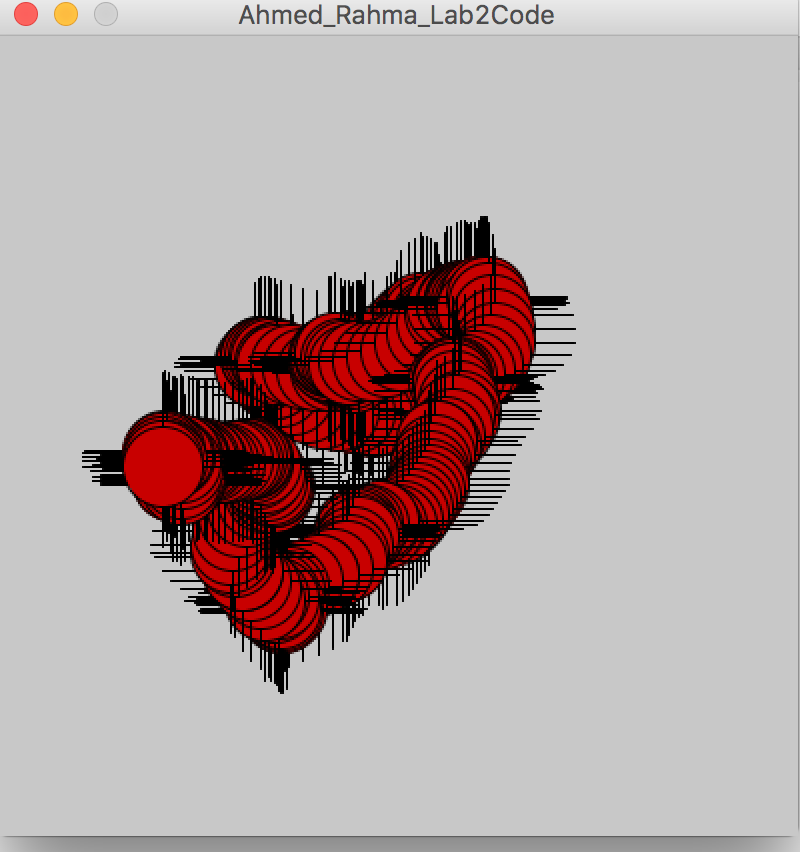
In this lab, I designed a cancerCell class that has two step functions, one based on Perlin noise and the other based on a random variable that determines the movement of x,y. A cancerCell is initialized and moves using the Perlin noise function (step()) sixty percent of the time and the random step function (randStep()) the other forty percent of the time. The cancerCell’s step size ranges from 0,10 whenever the mouse is pressed. The purpose of this lab is to get started on my cancerCell class which I will continue to build upon. This lab also gives me practice with perlin noise and random walkers.

Methods/Design:

My cancerCell class has six variables, xpos and ypos are the x and y positions of my cancerCell, tx and ty are the times (which correspond to the x axis on the time vs noise graph). Diameter is the diameter of the cancer cell and stepSize is the magnitude of the steps my cell takes. The cancer cell is made up of a red ellipse with four lines coming out of it. The cancer cell has two step function: step and randStep. Step uses ideas from Perlin noise and essentially increments time (tx, ty) by a chosen value. This time is then mapped to a x,y value within the dimensions of the display screen. This ensures that the cell always stays within the bounds of the display. My randStep function changes the x and y positions based on a random value (-1, 0, 1 that is multiplied by stepSize which as explained above is the magnitude of the steps). The display function displays the cancerCell as a red circle with four lines coming out of it. In the main sketch, a cancerCell called leuk is initialized. The setup function sets the size of the display and the background color. Draw has two conditionals, one that choose a random stepSize between 0 and 10 if the mouse is pressed and the other initializes a random value between 0 and 1 then it makes the cell move using the Perlin noise function (step()) sixty percent of the time and the random step function (randStep()) the other forty percent of the time.

Results:

My code initializes a cancer cell that moves using Perlin noise 60% of the time and using a random function the other 40% of the time. When the mouse is pressed, the step size varies. The cancerCell continues to move until the program is closed, otherwise it just keeps going.

Figure1: shows a moment in time of my cancerCell moving. The background is not set within the draw function so the path of the cell can be seen.

Conclusion:

My cancerCell can move using two different movement functions that take use stepSize (which changes when the mouse is pressed) to determine how far their steps will be with each movement.

Next steps:

In the future, I can make my randStep function include bounce for when the cancer cell goes off bounds. This aspect is included within the Perlin noise function but my random movement function is lacking in this aspect. I would also make it possible for my cell to stop movement based on certain conditions such as contact with a boundary.

Credit/Acknowledgements: I used our textbook, Nature of Code, to refresh my memory of the random walker class. I also worked with Lucy to talk through the various parts of my code.

Citation:

Shiffman, D. (2012). Nature of Code. I.1, I.2, I.3

