Rahma Ahmed 9/21/17

Cancer Cell Directed Motion

Background/Introduction:

In this lab, I improved my cancerCell class by incorporating vectors for the position, velocity, and acceleration of the cell. I also included a food source that the cell has a certain probability to move towards. The purpose of this lab is to improve the complexity of my code by incorporating factors that make it more interesting. I made an array of ten cells which move independently and have a 30 percent chance of moving randomly, a 45 percent chance of moving towards a food source and a 25 percent chance of being repelled by a site of medication.

Methods/Design:

The cancerCell class has seven variables. Position is a vector that represents the x, y value for the position of the cancerCell. Velocity and acceleration are also vectors that represent the velocity and acceleration of the cell. Topspeed is a float that limits the velocity so that the cell’s speed does not continue to increase for as long as the program is running. Diameter is the diameter of the cancer cell and stepSize is the magnitude of the steps my cell takes. Rand is a color that is randomly chosen for the cell. The cancer cell is made up of an ellipse with four lines coming out of it. This class has a constructor which initializes all the variables. There are then three different step functions. The randStep function, which is essentially the same from last lab, 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 foodStep function calculates the vector between the food source and the cell and makes the cell accelerate towards the food by adding this vector (which is the normalized acceleration after subtraction) to the velocity. The medicineStep does the opposite of foodStep, it repels the cell by negating the value of acceleration. The display function displays the cancerCell as a circle with four lines coming out of it using the x and y values of the position vector to set the position of the cell and the color variable, rand, to set the color.

In the main sketch, an array of cancerCells called leuks is initialized, the x and y values of the positions and the width/height (foodDiam/medicineDiam) of the food (a blue square) and medicine (a green square) sources are initialized. The setup function sets the size of the display, the background color, and sets the values for all the variables listed. Draw first draws the food and medicine sources. It then has a loop that goes through each of the individual cancerCells in the array, displays them, and has a few conditionals that determine what is to be done with them. The first conditional chooses a random stepSize between 0 and 10, when mouse is pressed, that effects the magnitude of movement if the randStep function is used. Next a conditional determines, using a random number between 0 and 1, the probability of each of the step functions being called. There is a 30 percent chance of the cell moving using randStep, a 45 percent chance of moving using foodStep, and a 25 percent chance of using medicineStep.

Results:

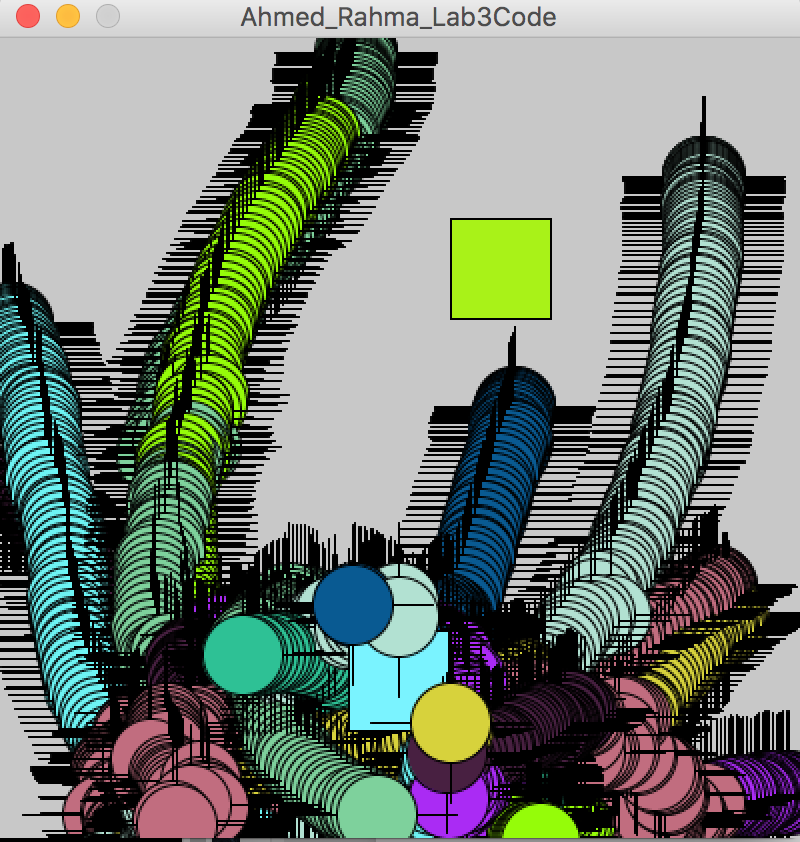
My code initializes a cancerCell array with each cell having properties that mostly use vectors. Whenever the mouse is pressed, each individual cell’s stepSize changes but this only affects the magnitude when randStep is the function used for movement. There is a 30% chance of the cells moving randomly, a 45% chance of moving towards a food source and a 25% chance of being repelled by a site of medication. The cells can acceleration but the velocity is limited so the cells do not continue to accelerate to a point where it is unclear how they are moving.

Figure1: shows a moment in time of my cancerCells moving. The cells are mostly around the blue square which is the food and they are far from the green square which is the medicine. The background is not set within the draw function so the path of the cell can be seen.

Conclusion:

My cancerCells have three different ways of moving, depending on a determined probability of a function being chosen, and they acceleration with each step.

Next steps:

In the future, I can make my randStep function have an efficient bounce that makes the cells move back into the display screen. Although the cells will eventually come back within the bounds of the display screen since they will be attracted to the food source, a bounce function will make it so that they do not go off the screen to begin with. I could also do something based on when the cells collide with one another so that they bounce off one another rather than overlap.

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

Citation:

Shiffman, D. (2012). Nature of Code. Chapter 1 & 2

