Rahma Ahmed 10/12/17

Cancer Cell Migration Assay

Background/Introduction:

In this lab, I incorporated extracellular matrices that affect migration of cancer cells. There are three different ECMs (ECMa, ECMb, and ECMc) each with its own friction and a force which effects directed motion. Looking at my code, if I choose to use ECMc rather than a or b, the cells will take the longest to reach the blood vessel. I believe this will happen because c has the largest friction coefficient and the random force that each ECM applies will have a much smaller effect on the cells than the force of friction since the random force is only applied 1% of the time. With this same reasoning, ECMa will be the fastest because it has the smallest coefficient for friction and ECMb will be in the middle in terms of time to arrival to source. This lab helps with visualizing how changing different variables can have an effect on cancer cells.

Methods/Design:

Similar to previous implementations of cancerCell, 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. Mass is the mass of the cell which is used in measurements of forces. 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 which is just an ellipse. This class has a constructor which initializes all the variables. There is an update function which updates the velocity and position of the cell based on changes that happen to acceleration elsewhere. The display function displays the cancerCell as a circle 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. The applyForce function has parameter force of type PVector and is a basic application of newton’s second law of motion. It takes into account the mass of the cell (using force = force/mass) when calculating the force that is added to acceleration. Drag has a parameter of type attractor. It applies drag that is caused when a cell enters an attractor (blood vessel) using the attractor’s drag coefficient as well as the velocity of the cell to calculate the force of drag.

Attractor is a simple class that’s used for making an object that attracts cells to itself. It has five variables, a position vector for the location of the cell, two floats for diameter and mass of the attractor, a float for drag coefficient and a float for the gravitation constant. The constructor gives values to all the variables/ the display function displays the attractor as a red ellipse (only half the ellipse shows on the display because the center is at the edge of the display). The attract function does the calculation for attracting a cell towards the attractor.

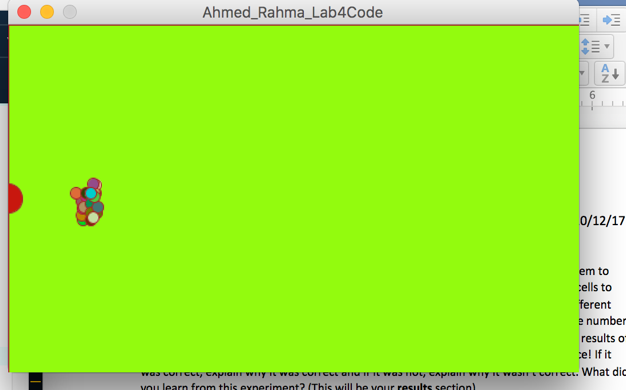
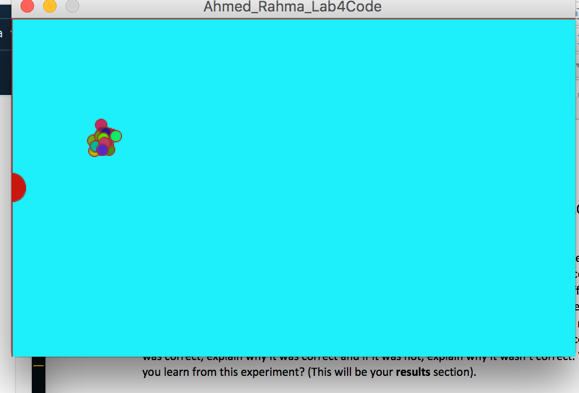
In the main sketch, an array of cancerCells called leuks, an attractor names blood and a random float are all initialized. The setup function sets the size of the display, initializes the cell array to be of size 25, populates the array with cells, and initializes the attractor. The three ECM functions (ECMa, ECMb, and ECMc) each have a rectangle the size of the whole display with a different color (a is blue, b is green, and c is purple. All three have friction that only differs by the friction coefficient in each function and they have a random vector that is applied as a force on the cell 1% of the time affecting the cell’s direction of movement. Draw sets the background to a grey color, chooses a random number that determines the probability of a random vector being applied. It then calls one of the ECM functions and displays the attractor. A for loop is then used in draw to display the cells in the array, add the attractive force of the attractor (blood) to them, and update their position. Also within the loop is a conditional which causes cells to experience drag once they are inside the blood vessel (attractor).

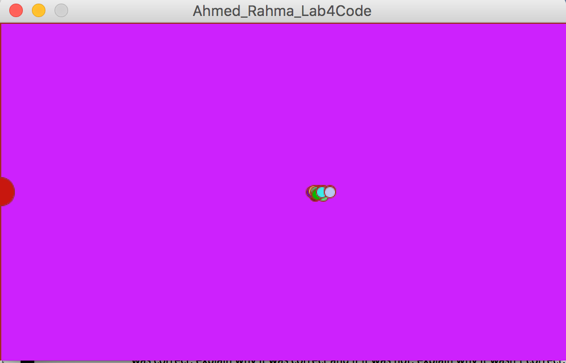
Results:

My hypothesis was supported when I ran the code because with ECMc, the cells took the longest to arrive at the attractor and with ECMa it was the fastest. Though the random force that each of the extracellular matrices applied did have an effect on time it took for the cells to reach the attractor, I tried setting the random forces to the same value and still got the same results. This showed me that the



Figure1: Graph of the frame count for each of the extracellular matrices



Figure 2: The 25 cancerCells moving towards the attractor (red semicircle at the left of the display) through the different ECMs. Blue display is ECMa, green display is ECMb, and purple display is ECMc.

Conclusion:

My hypothesis was correct cells are fastest when in ECMa and slowest when in ECMc.

Next steps:

In the future, it would make more sense to have my ECM as a class. I could also have different graphs that show the effects of only friction changing and nit friction and the random force. It would be more accurate to get an average number of frames instead of just using the value from two or three runs.

Credit/Acknowledgements:

I used our textbook, Nature of Code, and I also worked with Lucy and we talked through the various parts of the project.

Citation:

Shiffman, D. (2012). Nature of Code. Chapter 2 & 3

