Rahma Ahmed 10/26/17

Metastasizing cancer cells

**Background/Introduction:**

Cancer cells can spread to other parts of the body through a process called metastasis. In this lab, I implemented a StemCell that is an extension of the given CellWalker class. These are two kinds of cancerous cells that metastasize differently. A Tumor class is also used to create an arrayList of cellWalkers and allow them to run and reproduce. Different cell lines will have a variation of symprob (probability of a stem cell reproducing a cell) and prolif (probability a cell will reproduce) values. The two cell lines that we will test are KBard which has a symprob = 0.1 and a prolif = 0.02 and NBard which has symprob = 0.2 and prolif = 0.01. It seems like KBard will last the longest because its cells have a higher chance of reproducing and therefore the most efficient cell type.

Methods/Design:

The CellWalker class has four variables, two PVectors for position and velocity and two floats for telomeres and lifespan. The constructor defines the initial location of the cell. It takes in a PVector that is for the position, sets velocity to (0,0), chooses a random value between 0 and 50 for lifespan and chooses a random number between 0 and 10 for telomeres. The display function draws the cell body, the nucleus, and the receptors. Isdead returns true if the cell’s lifespan is less than 0.0 and false otherwise. The reproduce function makes and returns a new cellWalker with the same number of telomeres as its parent cell’s number. Step makes the cell move with a 30% chance of it moving towards a source and randomly the other 70 % of the time. Lifespan decreases with every iteration of the step function. Position is then constrained to the width and height of the display. Finally, the run function updates the position of the cell through the step function and displays the cell.

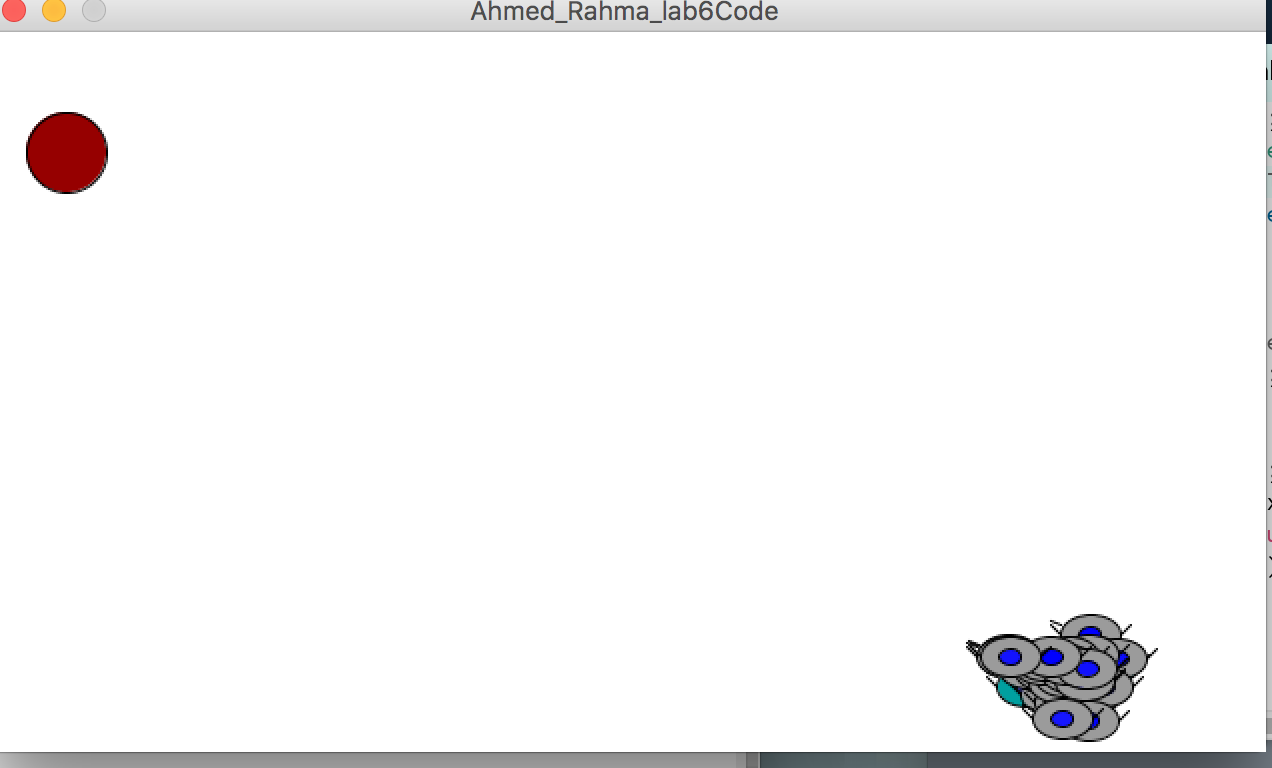
The StemCell class is a child of the CellWalker class and therefore inherits all of its methods. This class starts off with initializing a value called symprob which is the probability of a StemCell reproducing another StemCell. The constructor of stemCell runs the constructor of the superclass CellWalker, then initializes lifespan and telomeres to 255. It has its own display function which is essentially exactly like the parent class’s method but the cell has a different color. Lastly, it has a reproduce function that creates a stemCell *symprob* percent of the time and CellWalker otherwise.

The tumor cell is essentially a arraylist of type CellWalker. The iterator is first imported at the start of the file. The CellWalker arraylist is then initialized and a float, prolif, is initialized and used to determine the possibility of a cell reproducing. The constructor adds one StemCell and one CellWalker to the arraylist. Is dead returns true if the arraylist size is zero or less and false if it is greater than zero. Run creates a new arraylist called bornArr for all of the new cells reproduced and an iterator to go through the entire arraylist. A while loop is used to get the next cell in the list and run it. If the current CellWalker has telomeres, it has a prolif probability of reproducing a cell that is added to the bornArr list. These cells will be displayed at the next call of tumors.run(). The run function then decreases the number of telomeres of the current cell by one. It then removes all dead cells and adds all cell in bornArr to the array list as mentioned above.

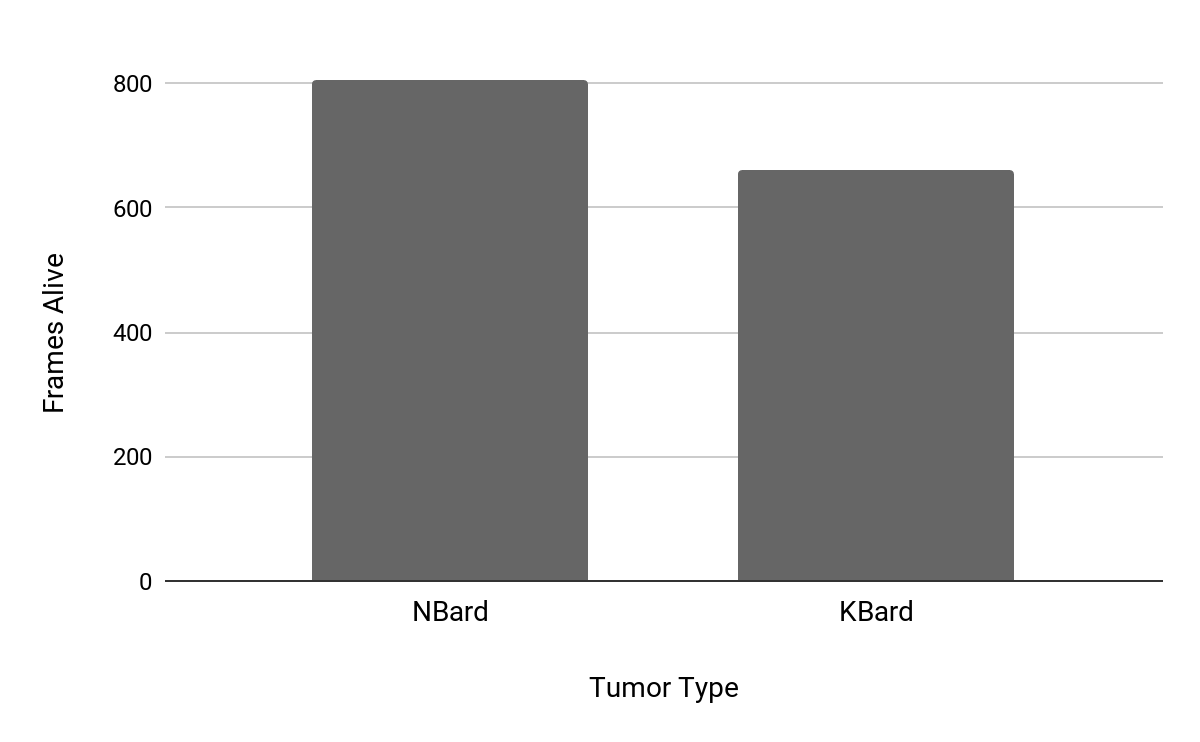
The main sketch initializes a Tumor c, and a PVector for source. The setup function initializes the size of the display to 640, 360 and initializes the value of tumor c with input PVector(width, height) and source at 40, 60. The draw function sets the background of the display, calls the run function on the tumor object and draws the source. Draw also has a frame count and a stop function that stops the program when all the arraylist in celltumor is empty.

Results:

My hypothesis was incorrect. It turned out that the higher symprob lasted longer, This makes sense now because if a stemCell produces another stem cell, it has the opportunity to produce another stem cell whose telomeres are independent of the parent cell. This then increases the chances that the child cell reproducing and continuously growing the population of the tumor’s cells.



**Figure 1:** Image of the cells as they move towards the source of nutrients shown in red and reproduce.



**Table 1:** Shows the average number of frames each tumor was able to stay alive for three trials.

Conclusion:

My hypothesis was incorrect, symprob’s value has a much more dramatic effect on the tumor’s overall lifespan.

Next steps:

In the future, it would make more sense to have StemCell’s display function use the characteristic of inheritance which helps decrease repetitive code between the parent and child class.

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.

https://www.cancer.gov/types/metastatic-cancer