CS 436 Final Project

Introduction

White blood cells, also known as Leukocytes, are the muscle behind your immune system, locating and destroying bacteria, viruses, fungi, and other potentially hazardous foreign bodies in your circulatory system. Typically, these cells account for about 1% of your blood, but people with conditions such as HIV/AIDS, or recipients of cancer treatment, find themselves with lower counts of these vital cells. In general, a lower white blood cell count, also called “leukopenia,” correlates with higher risk of infection, illness, and eventually death.

When it comes to harmful foreign bodies, viruses in particular have a very insidious approach to infection. In a process known as the “lytic cycle,” viruses enter a host cell, such as a blood cell, commandeer that cell’s enzymes to produce new virus particles, then the replicas burst out of the host cell, moving on to infect more cells. In many cases, the process of breaking out of the host cell destroys it (called “lysis”), which is obviously not good for the organism that these cells are a part of.

Although the human immune system is exceptionally complex, observing processes like this, even in an exceptionally simplified context, can help us better understand why certain people are at risk of infection. This is why we chose to observe the effect of varying white blood cell counts (continuous from “low” to “high”) on virus exposure outcomes.

The Model

We constructed an agent-based model to observe a macroscopic phenomenon, that is, the correlation between low white blood cell count (leukopenia) and worse outcomes for viral infection. The agents are as follows:

Cells:

* Have position, color, angle, speed, radius, and ID
* Can move to a new position and locate its “neighbor” cells
* Blood Cells are a subtype of cells
  + Also have “isInfected” property
  + Can check neighbors to see if a virus is present
  + Can “infect,” i.e. become infected if a virus is close enough and a random probability is satisfied
  + Can divide if not infected and the current amount of blood cells is less than the limit
* Virus is a subtype of cells (even though they’re not biologically “cells”)
  + Can only do what a Cell can do
* Defender (i.e. White Blood Cells)
  + Can Check neighbors to see if a virus is present; if a virus is within a certain radius, it will change its angle to follow the virus; if it is within a tighter radius, it has a chance to destroy the virus based on a probability
  + Will spawn a new white blood cell at a random location if it encounters and destroys a virus

Each of the cells move independently based on their speed, angle, and position properties, recognizing the borders of the window as impassible. As described above, white blood cells will often follow viruses and will “call” a new white blood cell to a random location in the window when they encounter and kill a virus. Once a blood cell has become infected, it can no longer divide, will turn yellow, and, after five seconds, will rupture, spawning two viruses. The environment is as follows:

* The Window
  + The window is meant to represent a closed, simplified circulatory system, which is easily observable for our purposes
  + It is a grid with varying width and height based on the size of the window and a continuous x-y coordinate plane
  + It has global attributes that affect the behavior of the cells and simulation
    - Blood Cell Start Count
    - Virus Start Count
    - Defender (white blood cell) ratio (relies on blood cell start count)

On its own, the environment will not change or act on the agents, because the interaction among the viruses, white blood cells, and blood cells is our only concern in this model. The only way that the environment restricts the agents is making sure that they do not travel beyond the bounds of the window.

In order to represent the above in code, I started with a base “Cell” class that is inherited from by Defender (white blood cells), BloodCell, and Virus. The base class has methods and attributes necessary for locomotion, whereas each individual class is specialized for the desired function of the agent. There are global attributes that affect the interaction of these Cells in an object called “options,” containing such properties as their default radii, start counts, and infection/defense probabilities. Some of these properties can be modified through inputs in an options window that can be revealed when the simulation isn’t running.

Most of the interactions depend on simple mathematical functions that take into account their speed, angle, and current position to move them or determine their neighbors. Otherwise, most of the checks for interactions are achieved through simple conditionals. Overall, the logic implementation was very straightforward.

We chose to use JavaScript because of the inherent portability and rapid-prototyping capacity of the language as well as the JIT compilation that allows us to fix logic or runtime errors more precisely in a model that is constantly being updated. One of the main disadvantages was the fact that only a single thread is dedicated to processing the code, so the model had to be kept relatively minimal to still run smoothly. The variation between JavaScript engines based on browsers is inconsequential, because many of the same bottlenecks, if not additional ones, exist on every browser. Overall, even though JavaScript is not a fully-articulated object-oriented language, the relatively new class and inheritance structures made the code clean and manageable coming from a mainly object-oriented background.