COMP 426

Assignment 1

Design Documentation

Christopher Mukherjee

6291929

October 6, 2014

Techniques Used & Architecture

The 2D area of 1024 x 768 cells is represented by a global array in my assignment. The reason I used an array is because the size of the problem was already known. This means that resizing (which would be a problem for an array) would never be needed. Therefore, all I would need to do is change the values of specific cells, which is easy to do with an array.

In order to manage the state changing of the cells, each time the update function is called, four threads are created. Each of the four threads manages one quadrant of the total cell area (as seen in Figure 1). The threads are then joined in order to ensure that all threads are able to finish their computations before continuing.

|  |  |
| --- | --- |
| Figure 1 | |
| Thread  1 | Thread  2 |
| Thread  3 | Thread  4 |

Finally, recursion is used in the “HealSurroundingMedicine” function. This function is used to heal all the surrounding medicine cells when a cancer cell is healed. Starting with the cancer cell that is being healed, the surrounding cells are all checked to see if they are medicine cells. If they are, then they are turned into healthy cells and their surrounding cells are checked. With recursion, I can easily ensure that all the surrounding medicine cells can be healed when a cancer cell is healed.

Test Scenarios

Test Scenario #1: Directly healing a single cancer cell

User Input: Click on a red cancer cell.

Expected Result: Medicine is injected into the cancer cell and is instantly absorbed. The cancer cell is healed and turns green.

Test Scenario #2: Injecting medicine into a healthy cell

User Input: Click on a green healthy cell.

Expected Result: Medicine is injected into the healthy cell and the cell becomes yellow. The medicine is not absorbed and moves radially outwards by one cell position in each direction. Note that all the surrounding cells turn into yellow medicine cells regardless of their previous state.

Test Scenario #3: Indirectly healing a single cancer cell

User Input: Click on the cells around a single cancer cell. Be sure to click at a distance of two (2) cells away due to the radial expansion of the medicine. It should take three (3) clicks in order for the cancer cell to be sufficiently surrounded by medicine.

Expected Result: The cancer cell is cured and turns into a green healthy cell due to the high number of surrounding yellow medicine cells. The surrounding yellow medicine cells also become green healthy cells.

Test Scenario #4: Injecting medicine into a healthy cell at the edge of the area

User Input: Click on a green healthy cell that is at the edge of the area (either on the left, right, top, or bottom edges, or in a corner).

Expected Result: Medicine is injected into the healthy cell and the cell becomes yellow. The medicine is not absorbed and moves radially outwards by one cell position in each possible direction, being sure to not expand outside of the boundaries of the area. Note that all the affected surrounding cells turn into yellow medicine cells regardless of their previous state.

Test Scenario #5: A healthy cell turns into a cancer cell

User Input: Click on a red cancer cell that is surrounded by more than the majority (≥ 6) red cancer cells.

Expected Result: Medicine is injected into the cancer cell and is instantly absorbed. The cancer cell is temporarily healed and turns green. However, the cell quickly turns back into a red cancer cell due to the high number of surrounding red cancer cells.

Test Scenario #6: Simulation starts with ≥ 25% cancer cells randomly placed.

User Input: Restart the simulation several times (checking expected result each time).

Expected Result: The simulation starts with approximately the same amount of cancer cells each time (being ≈25%). The cells appear in different locations each time, due to the fact that they are randomly placed.

Most Important Part of Assignment

In my opinion, the most important part of the assignment is the use of multiple threads on a single core in order to attempt to increase the program’s efficiency. After completing the assignment, it can be discovered that using multiple threads on a single core does not always actually achieve an improvement in efficiency. To give an example, the computers in the multicore lab have the capability of running up to two threads in parallel per core. Thus, on these machines, using multiple threads on a single core will achieve some level of increase in efficiency. However, when running on a single-core or multicore machine that does not have the capability of running parallel threads on a single core, the threads will have to be executed sequentially. Thus, on these machines, having multiple threads running on a single core will not result in any increase in efficiency. In conclusion, on the machines that I have readily available for testing, having four threads that run in parallel is not possible when using only a single core.

Most Difficult Part of Assignment

For me, the most difficult part of the assignment was using OpenGL. I had previously had absolutely no experience using OpenGL, so I had to learn everything from scratch. However, OpenGL is a very well documented language so it was quite easy to find resources online to solve all the problems I faced. In the end, I was happy to have had this opportunity to experiment with OpenGL. It is a very powerful language and has quite a few interesting features.