Assignment 3

Adam Luqman Hakim Bin Mohamad 40021709

COMP 426
Concordia University
20/10/2020

Techniques Used & Architecture

I took the code that was built in Assignment 1 in this assignment and improved it by replacing the multithreading achieved with CUDA task parallelization using the standard thread library. A CUDA task scheduler was initialised and a parallel generic

parallel algorithm was introduced instead of specifically creating threads to handle the parallelization of the cancer simulation. The CUDA tasks carried out the sequential updating of cell states in sequential.

The entire programme is stored in one (.cpp) C++ file. This file is composed of a main function, 5 GLUT-related helper functions, 3 general cell update helper functions, and one class with an overloaded parenthesis) (operator. The individual roles and groups are: figure 1.

Control flow diagram:

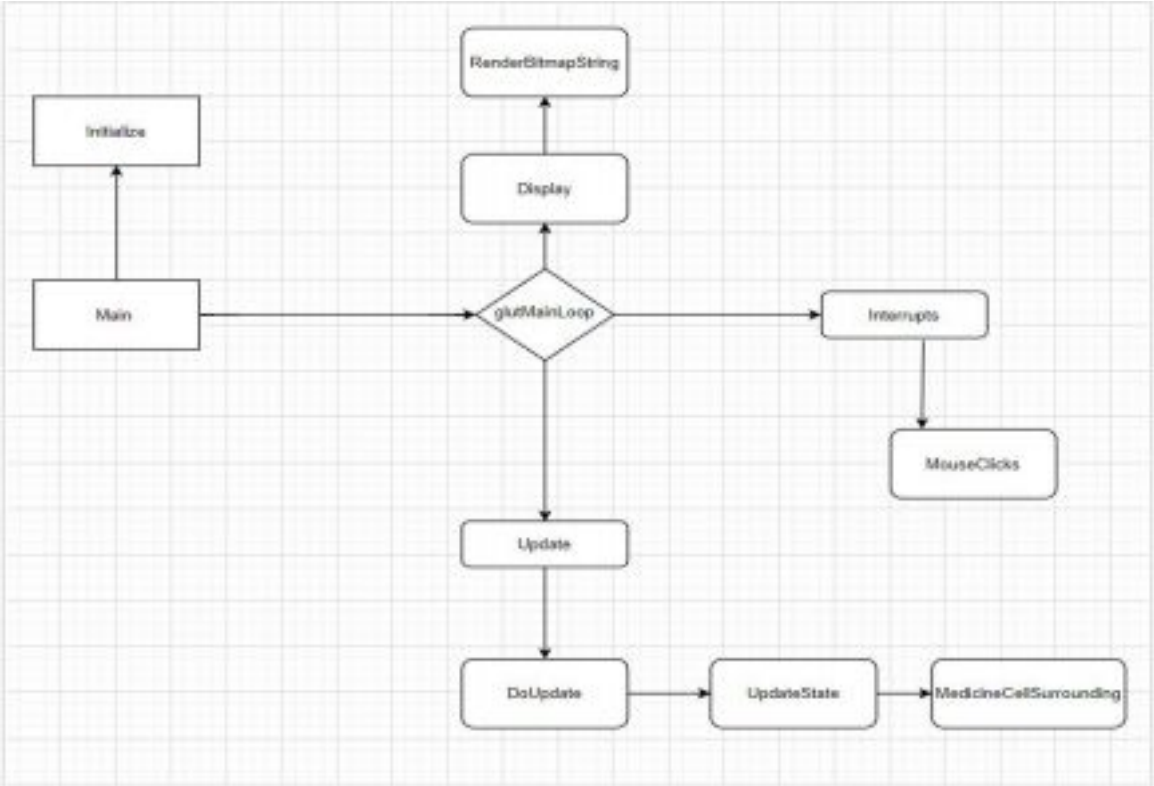

Figure :1

Most Important Part of Assignment

Replacing the regular thread library with CUDA tasks is the most significant aspect of this assignment. We can be sure, using CUDA tasks, that the software will use the required number of threads, scaled to the number of available cores. This means that the simulation is more parallelized and optimized now.

Most Difficult Part of Assignment

For me, using CUDA was the most challenging aspect of the task. I had absolutely no experience with CUDA. Also, to link the CUDA files to the code was challenging too because it needs to have the same bits for every aspect . So, I had to learn everything from scratch. It is an API that is very strong and has quite a few interesting features.

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: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 #3: 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 #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.

Reference

[1]. Home. (2017, June 07). Retrieved October 10, 2020, from http://www.opengl-tutorial.org/

[2].(n.d.). Retrieved October 10, 2020, from http://www.opengl-tutorial.org/beginners-tutorials/