Game of Life Project Report

This project reads an input .ppm file and writes output .ppm files iteratively using the ruleset from John Conway’s Game of Life. The Game of Life ruleset is used to evaluate the ‘live’ or ‘dead’ status of a given pixel and its neighbors. In the most general scenario, pixels with RGB values all equating to 255 are ‘live’ while pixels with all zeroes are ‘dead’. The ruleset for the Game of Life is as follows:

For ‘live’ pixels:

* Each pixel with zero or one live neighbors dies (isolation)
* Each pixel with four or more live neighbors dies (overpopulation)
* Each pixel with two or three live neighbors survives (stability)

For ‘dead’ pixels:

* Each pixel with exactly three live neighbors survives.

Our output files were created following a standard “Iteration\*.ppm” pattern. These files were then stitched together into a .gif using the ImageMagick command-line tool. The output .gif files represents 1000 iterations.

The source files were executed on a laptop computer with 4 cores, allowing for a maximum of eight threads to be used for parallel operations. For an example run of 100 iterations, the serialized version ran in about 12 seconds, while the parallel version ran in 3 seconds, achieving a speedup of 4.0. The C++ program was parallelized using OpenMP.

int main(){

…

#pragma omp parallel for

for (int i = 0; i < iter; i++) {

inputImage = outputImage;

runGame(inputImage);

temp = "Iteration" + to\_string(i + 1) + ".ppm";

//Convert temp string to char\* and write to file

writePPM(temp.c\_str(), outputImage);

}

}

By default, the omp parallel for pragma will spawn (cores \* 2) threads. This program is able to scale the number of threads spawned dynamically based on the number of CPU cores.

**Functions:   
static PPMImage \*readPPM(const char \*filename)** – reads the input .ppm file. This method was provided in a previous assignment.

**void writePPM(const char \*filename, PPMImage \*img) –** writes the input file to a .ppm. This method was provided in a previous assignment. Output files are written in binary format.

**int getNumLiveNeighbors(int cellMatrix[][3]) –**Called during execution of runGame(). This method returns the number of ‘live’ neighbors for a given input pixel. The cell matrix contains the values of the input pixel and its cardinal/diagonal neighbors.

**int getNumDeadNeighbors(int cellMatrix[][3])--** Called during execution of runGame(). This method returns the number of ‘dead neighbors for a given input pixel. The cell matrix contains the values of the input pixel and its cardinal/diagonal neighbors.

**void runGame(PPMImage \*inputImage) –** The driver function for the Game of Life Ruleset.

* Iterates through each pixel in the input image
* Determines the indices and values of its neighbors
* Gets the number of live and dead pixels using the getter functions
* Updates the input pixel based on the game conditions and the number of live/dead cells