Mixed Reality & Simulation - Forest Fire

Kranzl, Manuel
ai22m038@technikum-wien.at

November 6, 2023

1 Introduction

The objective of this exercise was to implement a basic forest fire simulation based on cellular automata. The simulation involves three different cell states:

- Tree
- Tree on Fire
- Empty

The following rules had to be implemented in the automata:

- If a cell is empty, a tree may appear in the next step with a probability of p_{tree}.
- If a tree is not on fire, the neighboring trees that are on fire will be counted.
  - If there are no burning trees in the neighborhood, the tree will randomly catch fire with a probability of p_{fire}
  - If there is a burning tree in the neighborhood, the tree itself will catch fire in the next step.
- If a tree is on fire, it will disappear in the next step.

The neighborhood could be chosen from either vanMoore or vanNeumann. Both p_{tree} and p_{fire} are customizable. The user has the capability to click on a tree to initiate a fire. A performance measurement has to be obtained when running the code on a forest with the minimum dimensions of 1024x1024.

2 Solution

2.1 General Structure and flags

The solution was implemented using C++ in conjunction with SDL2 for rendering the forest. A Makefile is provided for convenient compilation. As noted in the README file the following arguments can be set when starting the program (last column shows their default value):

height	height of the simulation forest	1024
width	width of the simulation forest	1024
measure	Disables graphic output for measures computational time	false
generations	number of generations when only measuring	100
threads	number of CPU threads	1

2.2 Calculation

The calculation of the Cellular Automata occurs within its simulate routine. During this process, the previous tree statuses are copied, and a parallelized loop below calculates the new generation.

```
#pragma omp parallel num_threads(nthreads)
    {
    unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
    std::mt19937 gen(seed);
    std::uniform_real_distribution<float> dis(0.0, 1.0);

        #pragma omp for
    for (int i = 0; i < width; i++){
        for (int j = 0; j < height; j++){
            if (!old_status[i][j].get_tree()) {
                float prob = dis(gen);
                if (prob<=probGrowth) {
                    status[i][j].set_tree();
                }
            }
            else {
                if (old_status[i][j].get_fire()) {
                    status[i][j].reset_tree();
                }
                else {
                    int fire_neighbor_count = 0;
                    for (int k = 0; k < neighbors.getNeighborCount(); k++) {
                        int x_neighbor = i + neighbors.getNeighbor(k).first;
                        int y_neighbor = j + neighbors.getNeighbor(k).second;
                        if ((x_neighbor>=0) && (x_neighbor<width) && (y_neighbor>=0) && (y_neighbor<
                            if(old_status[x_neighbor][y_neighbor].get_fire()){
                                fire_neighbor_count++;
                                break;
                            }
                        }
                    }
                    if (fire_neighbor_count > 0) {
                        status[i][j].set_fire();
                    }
                    else {
                        float prob = dis(gen);
                        if (prob<=probCatchFire) {
                            status[i][j].set_fire();
                        }
                    }
                }
            }
        }
    }
```

After this loop the old status gets deleted.

2.3 Features

The available features are as follows:

j	Decrease p_{fire}
k	Increase p_{fire}
n	Decrease p_{tree}
m	Increase p_{tree}
Esc	End the Simulation

3 Results

3.1 Time Measurement

Here are the time measurement results for a resolution of 1024x1024 and 1000 generations:

```
Resolution 1024x1024, 1000 Generations:
0:16.926; (1 threads)
Resolution 1024x1024, 1000 Generations:
0:03.140; (12 threads)
```

Please note that these measurement times were obtained using an AMD Ryzen 7 3800x processor.

3.2 Screenshots

Below are screenshots of the completed simulation with a 300x300 field.

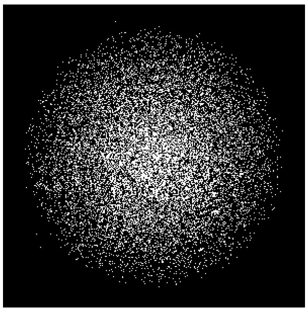

Figure 1: Start

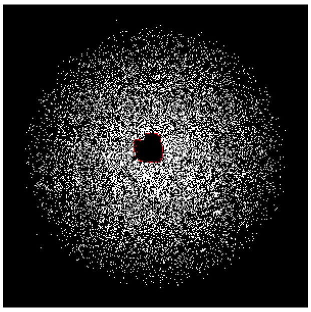

Figure 2: Fire

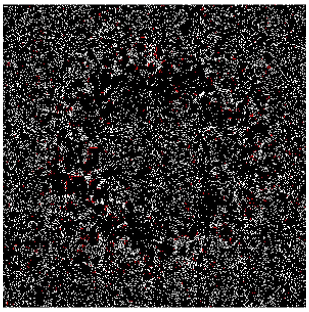

Figure 3: End