Forest Fire Simulation Report

CONTENTS

INTRODUCTION

Simulations run through python can, depending on the parameters chosen, accurately simulate what a real-world scenario may look like. Therefore, it becomes possible to use simulations to mimic how natural disasters such as forest fires may spread. The simulations run within this project therefore aim to mimic how a fire may spread through a forest, with random effects being considered and accounted for.

BASE MODEL DESIGN

ARRAY AND TIME-STEP DESIGN

This simulation will design the array in one of two ways, either by the default values or the user-given values. The default values of N=50, and 1000 time steps will lead to the creation of a 50 by 50 array, with 1000 steps then being run per simulation. N being 50, or the user inputted value, will create the initial array with 50 columns and 50 rows; this array is then filled with 0s and 1s randomly. Random generation of the initial array was chosen to remove bias, and make the simulations run more accurate of how the real-world scenario would be.

RANDOM CHANCES

For the base simulation, two random parameters were used. In the second simulation, a further random effect was added; this was the chance of rain, which will be explained further in the report. Of the two random parameters used within the base simulation, the first used was the chance of lightening randomly occurring. This will turn a 'tree' cell to a 'fire' cell, thus creating a mechanism through which the simulations are unable to become filled with only 'tree' cells. The next was the random chance of a tree growing in an empty cell, thus turning an 'empty' cell to a tree cell. Through this random effect, the simulations are therefore unable to become filled with only 'empty' cells. These random chances are how the initial time-step array becomes distinctly different at each step iteration.

CELL INTERACTION

For the simulations to run, logic must be created behind how cells are able to 'see' each other, and therefore how they can interpret one another. This process however needs to be able to compensate for when the cell is located within the top or bottom row, or when the cell is at the very right- or left-hand side.

These simulations run based on the following cell interaction logic:

- If a cell is within the top row, then the cell above it is considered equal to the state of the cell itself. For example, if the cell being analysed was a 0 (empty), then the cell above would be considered a 0 as well, thus leading to no direct effect on the original cell.
- If a cell is within the bottom row, then the cell below it is considered equal to the state of the cell itself. For example, if the cell being analysed was a 0 (empty), then the cell below would be considered a 0 as well, thus leading to no direct effect on the original cell.
- If the cell is located at the right most-hand side, then the cell to the right is considered to be in the same state as the original cell itself. For example, if the cell being analysed was a 0 (empty), then the cell to the right would be considered a 0 as well, thus leading to no direct effect on the original cell.

- ➢ If the cell is located at the left most-hand side, then the cell to the left is considered to be in the same state as the original cell itself. For example, if the cell being analysed was a 0 (empty), then the cell to the left would be considered a 0 as well, thus leading to no direct effect on the original cell.
- ➢ Corner cells are treated as the previous logic explains.

STEADY STATE INVESTIGATION

A simulation can achieve what is called a 'Steady-State' after N number of steps, and then can be identified as the time step at which the overall amounts of each cell type remain around the same value. The conditions used to define the logic underpinning how a simulation runs are what dictate what the Steady-State conditions are, or if this is possible at all.

In order to investigate this, a number of simulations were run with different conditions, and the outputted graphs investigated to see what the percentages of each cell type were. Due to the number of simulations run, it is impossible to show all the graph in this report; all the results are available within the Data_Tested directory of this project. However, figure one shows one set of results presented in a graphical format. On average for this simulation, it was found that a Steady-State was achieved after around 10-time-steps. This is shown by the three lines becoming mostly horizontal, thus holding at nearly constant values.

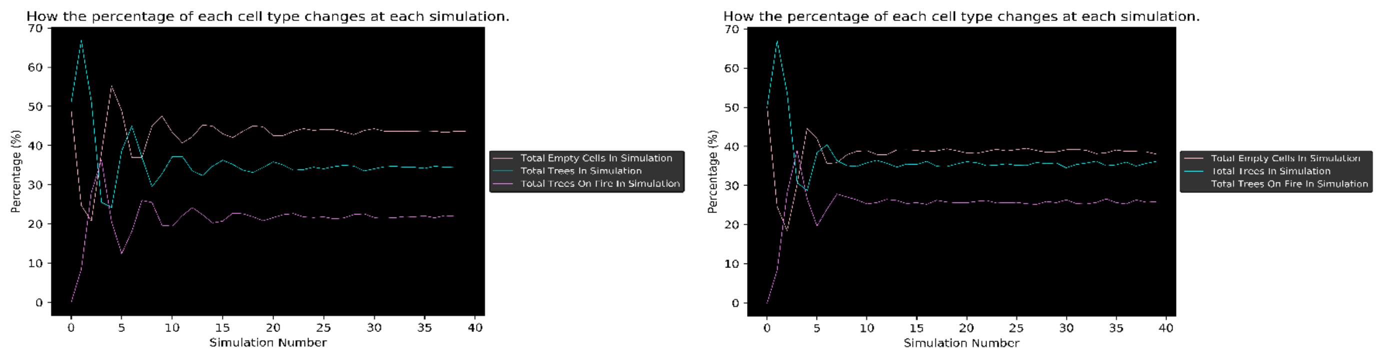

Figure 1 – *Graphical representation of the results from running both simulations, left is the base model, right contains the added effect of rain. After around simulation number (time step) 10, the three lines can be seen to level off, as this is when the simulations achieve a 'Steady-State'. This simulation used the following parameters: array sizes of 100, 40 time steps, chance of rain is 1 in 4, chance of lightening is 1 in 6, and the chance of a tree randomly appearing is 1 in 2.*

In total, nine sets of simulations were run to investigate how changing different parameters affected the seen steady-states. From this, conclusions could be drawn as to what effects the different parameters were having. Firstly, the effects of changing the chance of lightening were investigated

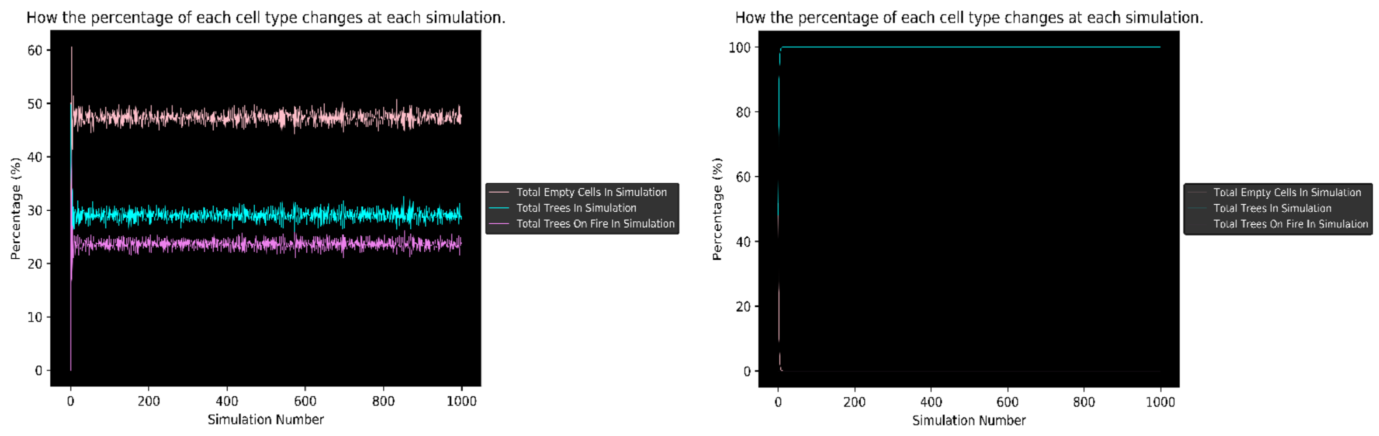

Figure 2 – *Graphical representation of the results from running both simulations investigating the effect of lightening, left is the base model, right contains the added effect of rain.*