IONIAN UNIVERSITY, DEPARTMENT OF INFORMATICS

Modeling and Simulating Fire Propagation in Forest Landscapes

Author: Athanasios Tsipis
Scientific Coordinator: Markos Avlonitis

June 10, 2016

Modeling fire propagation in a flammable landscape, such as a forest, under different weather conditions, is an interesting and challenging issue, widely studied in past literature. This is mainly due to the fact that fire is an extremely difficult process to experimentally manipulate and recreate, especially at a landscape level. Many approaches have been proposed over the years to address this problem. One of the most prominent and successful is based on Cellular Automata. Current work revolves around the implementation of a fire spreading model, based on two-dimensional cellular automata and various environmental factors. The developed model integrates geographic information and variables found in previous researches, with the goal of adequately predicting spatial fire behavior, in heterogeneous and non-heterogeneous forest areas. The results are then being presented in a user-friendly fashion using simple symbols and tools of representation.

1. INTRODUCTION

Wildfires have always been a major plague for all natural area ecosystems and a significant ecological threat, and over the course of many centuries have exerted an exceptionally important influence on forest landscapes [1]. In fact, it has been calculated, that each year fires burn between six and fourteen million hectares of forest ground [2]. Therefore, they have the power to greatly transform forest environments and alter their systems. Improved understanding of their inner propagation mechanisms could aid in discovering new ways of predicting, preventing, ensuring fire safety and control, and mitigating the risks [3].

However, fires are a difficult process to experimentally predict or recreate at a landscape level. Computer simulations can reduce the time it takes to evaluate impacts generated by their destruction paths, but it requires a deep understanding of the particular ecosystem dynamics, and enough knowledge about the target environment, to be able to adequately model fire spatial processes, prior to the event [4].

Fires are the result of complex interactions among a plethora of different nature variables. These variables vary depending on their field of affect and include, but are not limited to, weather conditions, such as temperature, humidity or wind velocity and direction, ignition material, vegetation type, fuel moisture, topography, ground elevation and altitude, slope, spotting effect, etc., [5], [6].

Although computational tools that help predict fire behavior are still far from perfect [7], there have been many efforts in modeling fire spread and a number of different approaches and mathematical models have been proposed over the years [8], [9]. Many studies are based on deterministic models and are generally composed of a collection of equations, whose solution gives numerical values for the spatial/temporal evolution of one or more fire variables [10]. Others take advantage of the well known and regarded semi-empirical model of Rothermel [11], which allows to forecast the rate of spread and the reaction knowing certain fire properties and landscape conditions [5]. In any case the mathematical models proposed can be classified according to two approaches [12]:

- *Vector models:* These models assume that the fire front spreads according to a well-defined growth law, which evaluates the spacial evolution in relation to time [2]. If burning conditions are uniform, then the model takes a single, standard, geometrical shape that can be easily examined to determine the fire size, the perimeter over time and the area by using fractals [13]. Some of these mod-

els go as far as incorporating wave propagation techniques, while employing mechanisms to determine the temperature fields in sequence to fire growth by performing turbulent fluid flow calculations [2], [12], [14].

- *Cellular Automata:* The cellular automata models (henceforth referred to as CA), first introduced by Von Neumann [15] and later detailed by Wolfram [16], have been successful in modeling complex physical systems and processes. They have the ability of reproducing complicated phenomenon and representing complex systems, by applying simple rules in a lattice space or grid, where each cell is in a particular state at a specific moment in time according to the state of its neighborhood [17], [18]. In other words, CA are mathematical models in which space and time are discontinuous, and the state factors can take on values from a finite set, depending on local transitions rules and a small number of neighbors. The evolution rules can be connected to deterministic or probabilistic functions of the neighborhood and define the state transitions [5].

Research using the latter technology has been advancing in various fields, including transportation, economics, chemistry, and engineering [19], [18]. Furthermore, CA are used extensively in recreating several physical problems, where local interactions are involved [1], or to better understand dynamic and complicated urban and physical events, such as plant competition, epidemic propagation and vaccination, habitat fragmentation, cell reproduction, insect evolution, population growth, plant invasion and dispersal, disease spread, etc. [17].

The reason behind the popularity of CA can be traced back to their simplicity [12] and their ability to integrate spatial information and geographic data sets from various digital sources [17], [20]. Consequently, CA, and in particular 2-dimensional array (or grid) CA, are considered a suitable tool to model and simulate the behavior of fire fronts and how they spread in sequence to time.

Karafyllidis and Thnailakis in their work, in 1997, developed a CA model for predicting fire propagation, which was applied to hypothetical landscapes under various scenarios of weather and topography [1]. Later on, many more researchers adopted their proposed model in order to further improve on its parameters, customize its equations to their needs, and come up with alternative solutions to the everlasting fire problem. One such work refers to the simulation experiment, conducted by Alexandridis, et al., in 2008, regarding the wildfire case that swept through Spetses Island, in 1990 [21]. An extension to this model was proposed in the work of [22], where additional parameters were inserted to the fire propagation equation with the aim

of simulating more accurately the phenomenon. In the work of Quartieri et al., one more parameter was introduced, fire ignition, which was directly related to the burning neighbors, fuel topology, wind direction and land slope [12].

Current work revolves around the recreation of the experiments undertaken and described in the aforementioned articles. The main goal is to investigate and discover potential uses of CA in modeling and simulating wildfires in dynamic ecosystems, like forests. Hence, the rest of the study is organized as follows: In Section 2 we present the necessary theoretical background involving the CA framework. Section 3 introduces and describes in detail the proposed model and its mathematical concepts, equations, and variables, based mainly on the work of [21]. Section 4 continues with the presentation of the simulation process and its procured results. It is worth mentioning, that in order to validate the whole procedure, geographical data found in the paper of [21] were integrated to the fire model, which was simulated by the use of the programming language C. Section 5 concludes the research and provides directions for future thoughts and improvements on the problem and its solution, respectively.

2. Theoretical Background on Cellular Automata

Any physical system satisfying differential equations may be approximated as a CA by introducing finite differences and discrete variables [23]. CA, despite their structural simplicity, can exhibit complex dynamical behavior and are capable of successfully representing many processes. From a theoretical point of view, CA are comprised of four main components, which are analyzed more thoroughly below [12].

1. **The physical environment:** This element defines the immediate structure of "universe" on which the CA is computed. This structure is usually characterized by a discrete and uniform n-dimensional lattice (or a grid, or an array) of cells, which could adopt several different topologies, the most popular being the rectangular or hexagonal ones. Typically, these cells have equal dimensions and a finite size, whereas the lattice itself can be finite or infinite. The lattice can be further categorized according to its dimensionality. So in the case of CA with 1 dimension, their lattice is named *elementary cellular automata (ECA)* and their

cells are placed in a linear string. On the other hand if they are comprised of 2 dimensions the lattice becomes a grid [12].

2. **The cells' states:** At each site of the lattice, called cell, a physical quantity takes specific values. This value refers to the local state of the cell at any given time, whilst the global state of the CA's cells is characterized as its *global configuration*. [1]. Distinct states are usually represented by an integer. However, there are many cases in which cells inherit a continuous range of values and, therefore, the lattice is called a *coupled map lattice (CML)* respectively. Depending on the shape of the lattice, each cell has a fix size, which in the case of rectangular grids it is typically a l *X* l square.

3. **The cells' neighborhoods:** For every cell a neighborhood is defined that locally determines the evolution of the cell. The neighborhood consists of the target cell itself and some or all of the immediately adjacent cells. Each cell is restricted to local neighborhood interaction and as a result it is incapable of immediate global communication. Nevertheless, the state of each cell is altered simultaneously at any given time step based on conditions occurring at its neighborhood at the preceding time step. Generally, there are two types of neighborhood used in CA. The first one, called *von Neumann neighborhood*, includes along with the target cell its four adjacent cells to the north, east, south and west. The second one, named *Moore neighborhood*, contains the previous cells plus the ones on the northeast, southeast, southwest and northwest of current cell. The two types are illustrated in Figure 2.1 [12].

4. **The local transition rules:** The algorithm used to compute and update the cells' states is known as the CA's local transition rule. The rule acts upon a neighborhood in such a way that changes the cell's state from one time step to the next. Consequently, the CA involves in time as the transition rule takes effect and is applied to all the cells in parallel. The nature of the rule determines also the type of the CA. In the unusual case of more than one rules existing for the transition process the term *hybrid* CA is used. On the other hand, if the rule contains no stochastic components the CA is referred to as a *deterministic* CA, whereas if the converse is true, then the CA becomes a *stochastic (or probabilistic)* CA. It is worth mentioning that the transition rule is directly connected to a certain computational function, capable of transforming the CA's global configuration in time, hence it also known as the *configuration map*.

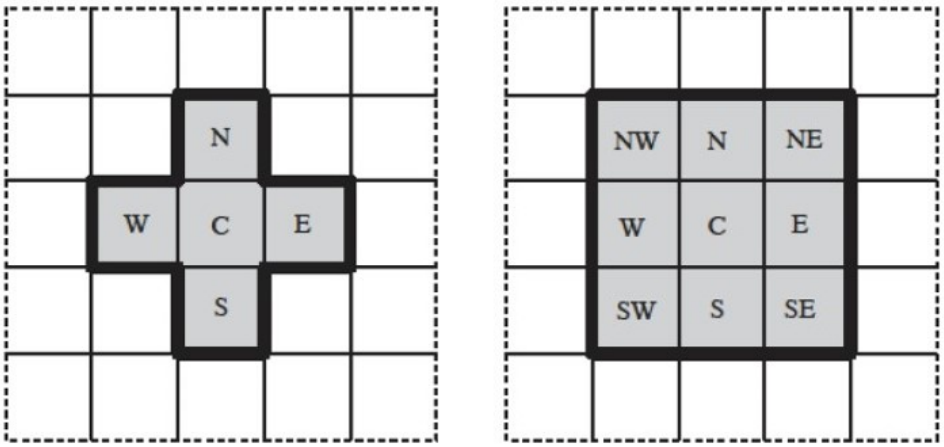

Figure 2.1: CA Neighborhoods: von Neumann (left), Moore (right) [12]

3. Fire Propagation Model

Based on the concepts described in Section 2 it is believed that CA are the best models to represent fire propagation in a homogeneous environment, such as a forest. Their ability to embed simple rules of evolution, that correspond to complex mathematical differential equations, make them the perfect candidate for modeling the phenomenon.

In present section the CA model for fire spreading in a forest field is presented. The proposed methodology incorporates a 2-dimensional grid, which corresponds to the forest area and maps different landscape units or patches of ground to the grid's cells. Though in many past cases, forest fires are simulated through the use of hexagonal cells, e.g. the research of [24], in current work the rectangular cell grid is preferred and utilized, with the aim of simplifying the necessary calculations and providing a more easily adaptable solution. Consequently, the shape of each cell is set to be a square with dimensions $l \; x \; l$, where l represents ground units in meters, and is defined based on the forest in question. The target forest can be simulated by acquiring the given area from a satellite image and converting it in a 2D array. However, for the purposes of current work a randomly generated forest area will be generated in order to test the model and its qualities, which adopts the framework proposed in the work of [21].

3.1. Implemented Methodology

The proposed methodology implements fire fronts by including fire evolution rules, which update the states of particular cells and their neighborhoods, and consequently CA's global configuration, over discrete time and space in a stochastic manner. Specifically we assume the following:

1. A random forest is generated and mapped to the 2-dimensional CA using a probability function.
2. The are four possible states where each cell can be found in at a specific instance in time, that correspond to forest fuel currently burning, forest fuel not yet burned, forest fuel completely burned down, and fuel-free forest ground.
3. A forest cell cannot be re-burned.
4. If a forest cell catches fire, then it is completely burned in one discrete time step.
5. The fire spreads from one burning cell to its neighbor cells probabilistically.
6. The probability of fire propagation depends on the variables: type of vegetation, density of vegetation, speed and direction of wind, and ground elevation.
7. Each of the aforementioned variables affects the fire growth independently.
8. The values for these variables are specified a priory to the simulation, based on geographical data received from the papers of [21][22].

3.2. Definition of Cell States

Every cell in the forest grid is characterized by a specific state, which evolves in discrete time steps. There are four possible states for each cell. These are:

- *State 0:* The cell contains no forest fuel. Thereafter, it is not flammable and cannot be burned.
- *State 1:* The cell contains forest fuel, which has not yet been ignited and remain unburned.
- *State 2:* The cell contains forest fuel that has been ignited and is currently burning.
- *State 3:* The cell contains forest fuel that has been already completely burned.

As already stated the states of all cell at a discrete time step define the global configuration of the CA. These states can be coded to a 2D matrix accordingly. This matrix is called the *State Matrix*, and an example of it can be seen in Figure 3.1 [25].

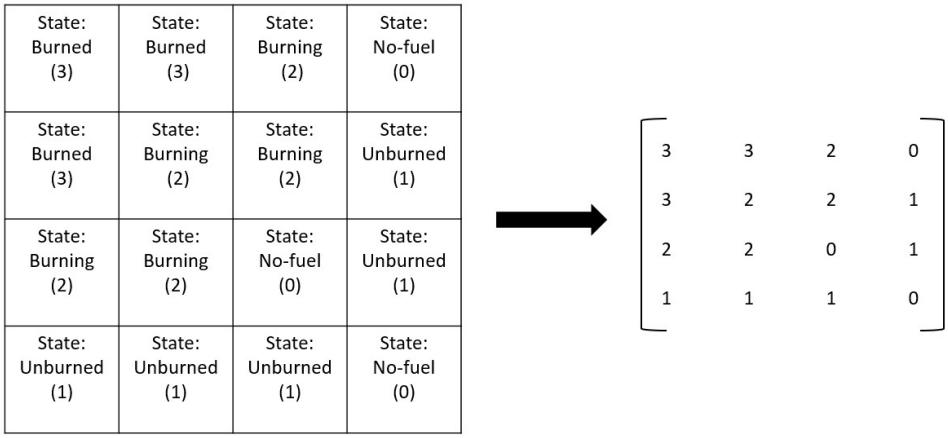

Figure 3.1: Random State Matrix

3.3. Rules of Transition

The evolution rules define the way in which the CA evolves in time. At each discrete time step *t* the rules take effect in all cells simultaneously. In order to model this effect the proposed transition rules are applied to all elements *(i,j)* of the state matrix. These rules are as follows:

- ***Rule 1:*** *IF state(i,j,t) = 0 THEN state(i,j,t+1) = 0.* This rule denotes that a cell with no forest fuel remains the same after a time step and cannot catch fire.
- ***Rule 2:*** *IF state(i,j,t) = 2 THEN state(i,j,t+1) = 3.* This rule implies that a cell that has caught fire in time step t will be completely burned in time step t+1.
- ***Rule 3:*** *IF state(i,j,t) = 3 THEN state(i,j,t+1) = 3.* This rule defines that a forest cell that has been burned will remain in the same state in the next time steps and cannot catch fire again.
- ***Rule 4:*** *IF state(i,j,t) = 2 AND state(i$\pm$1,j$\pm$1,t) = 1 THEN state(i$\pm$1,j$\pm$1,t+1) = 2 with p.* This rule inclines that a burning cell can ignite neighboring cells and spread the fire with a probability p, depending on the forest environmental factors occurring in the neighborhood Figure 3.2 [5].

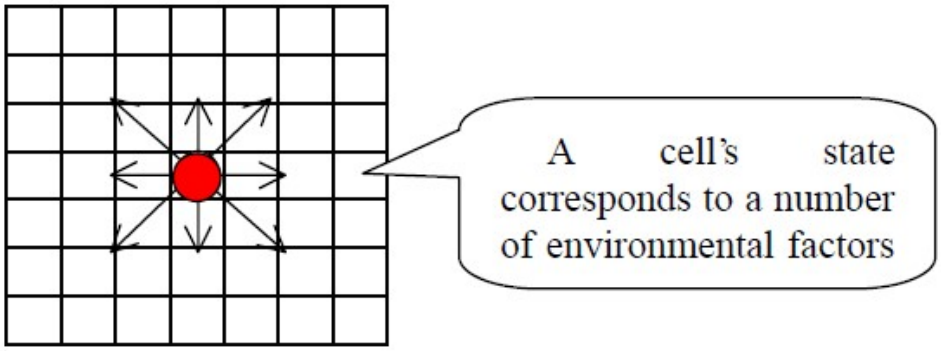

Figure 3.2: Possible fire growth positions based on probability p [5].

3.4. Fire Propagation Environmental Factors

In current subsection the transition differential equation for the fire propagation probability is provided, along with the definition of the environmental factors that influence its outcome. In this context, the concepts described below are validated.

The probability of a forest cell catching fire, symbolized as p_{burn} is calculated by the equation:

$$p_{burn} = p_h(1 + p_{veg})(1 + p_{den})p_w p_s \tag{3.1}$$

where p_h is considered to be a constant probability that a cell, witch is adjacent to a burning cell, under no other factoring conditions will catch fire at the next time step. On the other hand, the variables p_{veg}, p_{den}, p_w, and p_s denote the probabilities of fire propagation according to the type of vegetation, the density of vegetation, the wind speed and direction, and the ground elevation (slope) occurring in the cell, respectively. All these environmental factors are multiplied by the p_h constant, in order to finally acquire the corrected possibility of fire spreading.

The probabilities of vegetation type (p_{veg}) and density p_{den} are split into a number of specific categories, where each category adopts a discrete value. These categories are shown in Tables 3.1 and 3.2. Their combination represents the number of possible cases that can be found in the forest in relation to its vegetation. It is worth stating that in order to illustrate the different combinations, in regards to the State Matrix, the two factors are further organized into similar matrices, the *Vegetation Matrix* and *Density Matrix*, where each element is assigned one of the classification values [21].

Table 3.1: Vegetation Type Classification

Categories		
Classes	**Type**	**Value**
1	Plants	-0.3
2	Bushes	0
3	Trees	0.4

Table 3.2: Vegetation Density Classification

Categories		
Classes	**Density**	**Value**
1	Sparse	-0.4
2	Normal	0
3	Dense	0.3

The above values are the ones used in current experiment. However, they can vary depending on the target forest and can be obtained from geographical data sets. The same is valid for the case of modeling the probability of the wind effect, which is declared by the following equation:

$$p_w = exp(c_1 V) f_t \tag{3.2}$$

where f_t is calculated as follows:

$$f_t = exp(c_2 V(cos(\theta_w) - 1)) \tag{3.3}$$

Replacing f_t in equation 3.2 with the one in 3.3 we get:

$$p_w = exp(c_1 V) exp(c_2 V(cos(\theta_w) - 1)) \tag{3.4}$$

where c_1, c_2 are constants based on gathered environmental data, V is the wind velocity and θ_w represents the angle between the fire propagation and the wind direction, which can take any continuous values between 0^o and 360^o.

The last probability factor relates to the slope angle between the burning cell and the neighboring cell and depends on the ground elevation difference. The equation that models this effect is presented below:

$$p_s = exp(\alpha\theta_s) \tag{3.5}$$

where α is a constant, that can be adjusted from experimental data, and α_s defines the slope angle. Due to the fact that the implemented CA is being modeled as a rectangular grid, depending on whether the two neighboring cells are adjacent or diagonal to the burning cell the value of θ_s is calculated differently. So in the case of adjacent cells we have the following equation for θ:

$$\theta_s = tan^{-1}((E_1 - E_2)/l) \tag{3.6}$$

whereas in the case of diagonal cells the above equation takes the form of:

$$\theta_s = tan^{-1}((E_1 - E_2)/l\sqrt{2}) \tag{3.7}$$

In both cases the *l* refers to the specified length of the cells' square side, whilst E_1 and E_2 denote the altitudes (ground (E)levations) of the two cells, respectively. In order to model the different altitudes, a similar approach to the vegetation type and density was used and so the *Slope Matrix* was created, which receives values for its elements based on Table 3.3. The values in this table are selected randomly. However, it is possible to gain exact information regarding certain forest ground elevations from corresponding geological data sets.

The rest of the used variables, found in equations 3.1, 3.4, 3.5, 3.6, and 3.7, were optimized for the CA's algorithm and are listed in Table 3.4. These values are based on the work of [21]. Nevertheless, they can be adjusted to correspond to all forest fire situations or changed according the chosen CA implementation.

Table 3.3: Ground Elevation Classification

Categories		
Classes	**Altitude**	**Value**
1	Flat	1
2	Extremely Low	2
3	Very Low	3
4	Low	4
5	Quite Low	5
6	Average	6
7	Quite High	7
8	High	8
9	Very High	9
10	Extremely High	10

Table 3.4: Environmental Factor Analysis

Factors	
Variable	**Value**
p_h	0.58
α	0.078
c_1	0.045
c_2	0.131
V	9
l	2

4. Fire Spreading Simulation

The model described in Section 3 provides the guidelines for the development of a fire spread testing program. In this way it is possible to recreate the fire conditions and apply them to a certain forest, with the goal of generating a simulation of the fire propagation in relation to time and the specific landscape. For the purposes of this study a program was written in the *C* programming language using the platform of *Dev*C++, that uses the generated data to illustrate the fire growth in a text file using simple text symbols. For a screen-shot regarding an example of the simulation process, in intervals of 5 time steps, please see Figure 4.1.

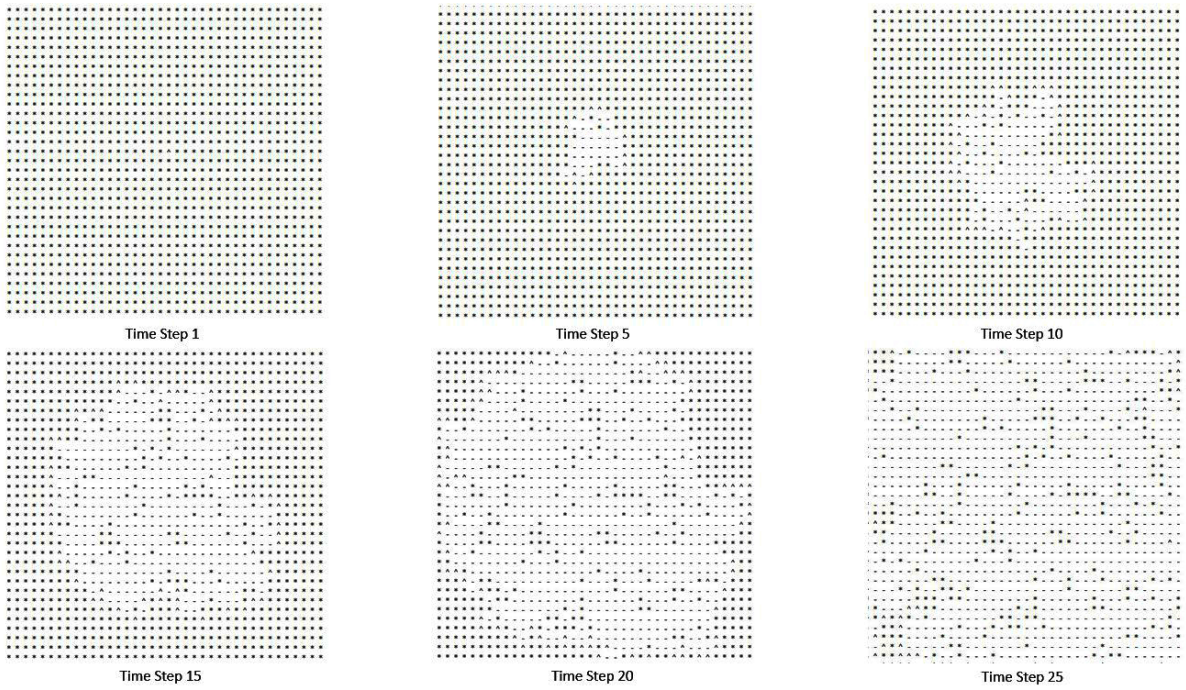

Figure 4.1: Fire Propagation Simulation Text Output

For the simulation 4 symbols were used (Table 4.1), that represent the 4 CA states:

Table 4.1: Definition of Simulation Symbols

Symbol	**Definition**
blank (space)	Area with no forest fuel
* (asterisk)	Area with forest fuel that has not been burned
$\wedge$ (caret)	Area with forest fuel that is currently burning
- (hyphen)	Area with forest fuel that has been burned and cannot be ignited

The first part of the program, Listing 1, lists the necessary libraries, and defines and initializes the variables used, which are based on the elements described in previous sections. In addition, it opens the appropriate text file for writing, in order to save the data and illustrate the simulation. The CA grid that is created has dimensions *100 x 100* and the simulation time steps that will take place are set to reach 70 in number before terminating the simulation program.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define LENGTH 100  // area length
#define WIDTH 100  // area width
#define TIME 70 // total time of simulation in timesteps

int main(int argc, char *argv[]) {

FILE *fire = fopen("fire_simulation.txt", "w+") ;
if (fire == NULL){
printf( "\nError opening file! \n") ;
}

// variables' declaration and initialization
int i = 0, j = 0, r = 0, c = 0, l = 2, angle_w = 180, con1 = 0.045, con2 =
    0.131, V= 9, timestep = 0, stateMatrix [LENGTH][WIDTH] = {0},
   nextStateMatrix [LENGTH][WIDTH] = {0};
int vegMatrix [LENGTH][WIDTH] = {0}, denMatrix [LENGTH][WIDTH] = {0},
   slopeMatrix [LENGTH][WIDTH] = {0};
char forestMatrix [LENGTH][WIDTH] = {0};
float p = 1, p_burn = 1, p_h = 0.58, p_veg = 1, p_den = 1, p_w = 1, p_s =
   1, a = 0.078;
```

Listing 1: Libraries and Variables

For the next part of the simulation, Listing 2, a randomly generated forest is created, using the *rand()* function, and mapped to the 2-dimensional array that represents the CA's grid. Each cell of the grid corresponds to an element of the array. Furthermore, according to the structure of the generated forest CA, the *State Matrix*, the *Vegetation Matrix*, the *Density Matrix*, and the *Slope Matrix* are also created, accordingly.

```
srand((int)time(NULL));

// rendomly generating trees in an area grid with size LENGTH x WIDTH and
    initializing slope and vegetation type and density
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    slopeMatrix[i][j] = (rand () % 10) + 1;
    i = rand() % LENGTH;
    j = rand() % WIDTH;

    if (forestMatrix[i][j] == 0){
      forestMatrix[i][j] = '*';
      stateMatrix[i][j] = 1;
      nextStateMatrix[i][j] = 1;
      vegMatrix[i][j] = (rand () % 3) + 1;
      denMatrix[i][j] = (rand () % 3) + 1;
    }
  }
}
```

Listing 2: Generation of CA forest and state metrices

In order to commence the fire simulation a flame must ignite a cell in the CA. That cell must contain flammable substances, which means it must be an element of the array that holds some kind of forest vegetation, symbolized with the number 1 in the *State Matrix*. This cell is selected randomly at time step 1. At that point the state of that cell changes to 2, which indicates that the forest material is now burning and can in turn spread the fire in its neighborhood according to a probability p_{burn}, Listing 3. If a cell with no vegetation is selected, then the program iterates until it chooses a flammable grid cell. When it does find such a cell, then the corresponding *forestMatrix* cell also changes from an * to a $\wedge$, indicating the fire ignition.

```
//Somewhere in the forest a fire is ignited

i = rand() % LENGTH;
j = rand() % WIDTH;

if(stateMatrix[i][j] == 1){
  stateMatrix[i][j] = 2;
  forestMatrix[i][j] = '^';
  }
else{
  while (stateMatrix[i][j]!=1){
  i = rand() % LENGTH;
  j = rand() % WIDTH;
  if(stateMatrix[i][j] == 1){
    stateMatrix[i][j] = 2;
    forestMatrix[i][j] = '^';
    break;
    }
  }
}
```

Listing 3: Fire iginition somewhere in the forest

The calculation of the CA's *global configuration* takes place in a *while{}* loop, which for every passing time step, it calculates for each element of the CA's array its state in the next discrete time step according to the transition rules, described in Section 2 and defined in the loop, with the use of nested *if/else* statements. In order to compute the probability p_{burn} for a burning cell to transport the fire to a neighboring cell, the program draws values from the vegetation, density, and slope matrices and calculates the probabilities p_{veg}, p_{den}, and p_s, respectively. Moreover, it estimates the p_w probability based on the declared variables. The transition rules are applied simultaneously on all array elements. To be able to implement this attribute another temporary matrix was created, called the *Next State Matrix*, which hosts all the necessary information for updating the cells' states during the next time step. For more information on the implementation process please refer to

Appendix A.

For the last part of the simulation (Listing 4), the program prints the results of each passing time step, by comparing the *State Matrix* with the *Next State Matrix* and updating the necessary data. This continues until the number of completed time steps

reaches the declared limit of 70.

```
// printing forest grid after each fire spreading timestep
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    if(stateMatrix[i][j]==3){
      forestMatrix[i][j]='-';
    }
    else if(stateMatrix[i][j]==2){
      forestMatrix[i][j]='^';
    }
    else if(stateMatrix[i][j]==0 || stateMatrix[i][j]==1){
      // forestMatrix cell state remains the same
    }
    printf("%c ", forestMatrix[i][j]);
    fprintf(fire, "%c ", forestMatrix[i][j]);
  }
  printf("\n");
  fprintf(fire,"\n");
}

// updating forest grid to next state
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    stateMatrix[i][j] = nextStateMatrix[i][j];
  }
  printf("\n");
}
```

Listing 4: Global Configuration update after each time step

To better understand the results of the experiment the simulation was repeated for a full forest grid, where all 10000 array elements contained flammable material (vegetation). The aim was to generate a graph, which explains the behavior of fire in a forest landscape with the aforementioned parameters. The result of this action is illustrated in Figure .2, which successfully depicts fire behavior in forest environments by showing the progression of total burned cells in time. The allocation of the graph can be explained if we consider that at time step 1 a vegetation cell catches fire, but does not burn until the next time step. From that point on, for each passing time step, cells start to be ignited in almost exponential rate due to the fact that more and more cells are burning and therefore can spread the fire. So we can see a dramatic increase in burned cells overtime. However, after a considerable amount of time has passed,

the fire begins to decline and eventually stabilizes at a certain number of burned cells, which in this example case scenario was somewhere just above 8000 cells. This phenomenon occurs because, by that time step, most cells in the CA are already burned, and consequently the fire cannot spread any further and eventually dies out. The simulation continuous until all involved cells are burned and the available time steps expire.

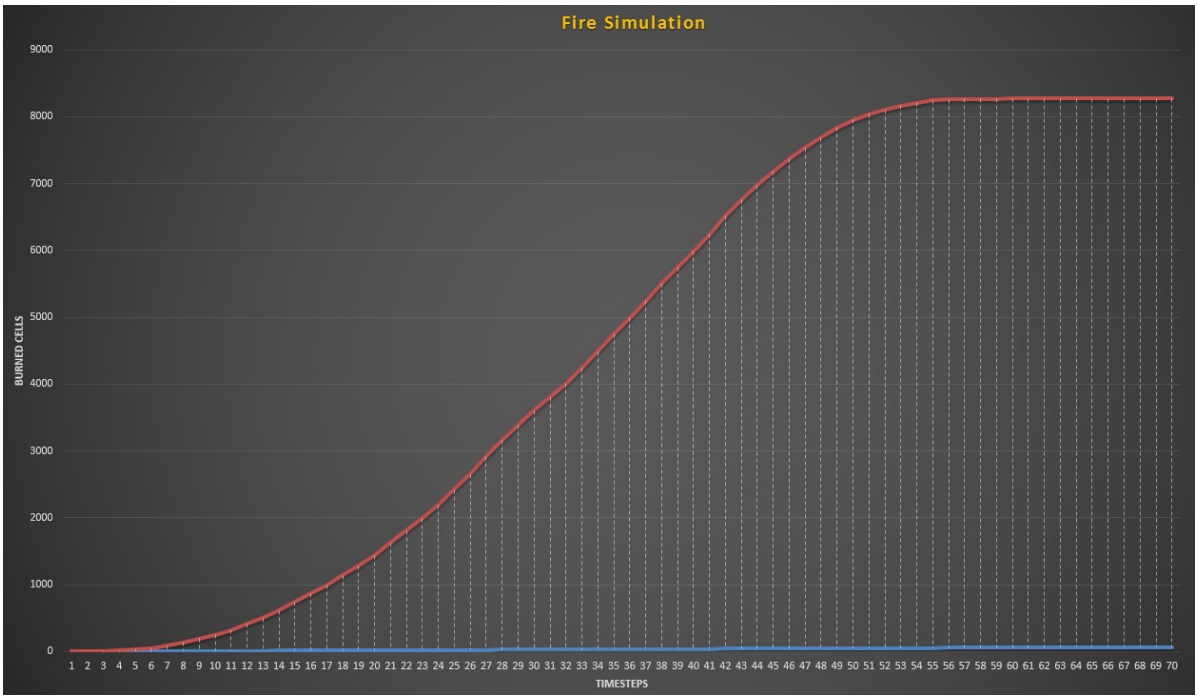

Figure .2: Fire Propagation Simulation Graph

A. Conclusions and Future Work

Present work attempted to provide a fire propagation framework that corresponds to real life fire circumstances in forest landscapes. The main goal was to formulate fire propagation parameters and create a model capable of explaining how fire progresses in time, on a certain area, under specific environmental conditions that relate to weather and geological data acquired from past literature.

To test the proposed theory, a *C* program was written that adopts these concepts and simulates the phenomenon, in a graphical manner, using simple text symbols. The model was then validated and considered successful due to its graphical representa-

tion and accuracy in explaining how fire progresses.

Future work may turn around the adoption of extra geographical and ecological factors that affect fire growth. However, it will be extremely important not to limit the scope only to ecosystem variables. On the contrary, it is considered especially meaningful to search for ways to insert into the model additional variables that relate to human reaction, such as fire suppression techniques by firefighters, or man-made constructions, such as cities or villages. This could lead to models that not only successfully predict the outcome of fires in forest or urban systems, but also could help in discovering new potential methods of fire repression, whilst avoiding negative consequences that emanate from uncontrolled wildfires.

REFERENCES

[1] I. Karafyllidis and A. Thanailakis, "A model for predicting forest fire spreading using cellular automata," *Ecological Modelling*, vol. 99, no. 1, pp. 87–97, 1997.

[2] A. H. Encinas, L. H. Encinas, S. H. White, A. M. del Rey, and G. R. Sánchez, "Simulation of forest fire fronts using cellular automata," *Advances in Engineering Software*, vol. 38, no. 6, pp. 372–378, 2007.

[3] F. Williams, "Mechanisms of fire spread," in *Symposium (international) on Combustion*, vol. 16, pp. 1281–1294, Elsevier, 1977.

[4] Y. Wu, F. H. Sklar, K. Gopu, and K. Rutchey, "Fire simulations in the everglades landscape using parallel programming," *Ecological Modelling*, vol. 93, no. 1, pp. 113–124, 1996.

[5] X. Li and W. Magill, "Modeling fire spread under environmental influence using a cellular automaton approach," *Complexity International*, vol. 8, pp. 1–14, 2001.

[6] W. W. Hargrove, R. Gardner, M. Turner, W. Romme, and D. Despain, "Simulating fire patterns in heterogeneous landscapes," *Ecological modelling*, vol. 135, no. 2, pp. 243–263, 2000.

[7] F. A. Sousa, R. J. dos Reis, and J. C. Pereira, "Simulation of surface fire fronts using firelib and gpus," *Environmental Modelling & Software*, vol. 38, pp. 167–177, 2012.

[8] A. KULESHOV and E. MYSHETSKAYA, "Numerical simulation of forest fires based on 2d model,"

[9] D. R. Weise and G. S. Biging, "A qualitative comparison of fire spread models incorporating wind and slope effects," *Forest Science*, vol. 43, no. 2, pp. 170–180, 1997.

[10] E. Pastor, L. Zarate, E. Planas, and J. Arnaldos, "Mathematical models and calculation systems for the study of wildland fire behaviour," *Progress in Energy and Combustion Science*, vol. 29, no. 2, pp. 139–153, 2003.

[11] R. C. Rothermel, "How to predict the spread and intensity of forest and range fires," *The Bark Beetles, Fuels, and Fire Bibliography*, p. 70, 1983.

[12] J. Quartieri, N. E. Mastorakis, G. Iannone, and C. Guarnaccia, "A cellular automata model for fire spreading prediction," *Latest Trends on Urban Planning and Transportation*, pp. 173–178, 2010.

[13] M. A. Finney, "Mechanistic modeling of landscape fire patterns," *Spatial Modeling of Forest Landscapes: Approaches and Applications. Cambridge University Press, Cambridge*, pp. 186–209, 1999.

[14] A. Lopes, A. Sousa, and D. Viegas, "Numerical simulation of turbulent flow and fire propagation in complex topography," *Numerical Heat Transfer, Part A: Applications*, vol. 27, no. 2, pp. 229–253, 1995.

[15] J. v. Neumann and A. W. Burks, "Theory of self-reproducing automata," 1966.

[16] S. Wolfram, "Universality and complexity in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1, pp. 1–35, 1984.

[17] S. Yassemi, S. Dragićević, and M. Schmidt, "Design and implementation of an integrated gis-based cellular automata model to characterize forest fire behaviour," *Ecological Modelling*, vol. 210, no. 1, pp. 71–84, 2008.

[18] A. Ohgai, Y. Gohnai, and K. Watanabe, "Cellular automata modeling of fire spread in built-up areasâĂŤa tool to aid community-based planning for disaster mitigation," *Computers, environment and urban systems*, vol. 31, no. 4, pp. 441–460, 2007.

[19] A. Ohgai, Y. Gohnai, S. Ikaruga, M. Murakami, and K. Watanabe, "Cellular automata modeling for fire spreading as a tool to aid community-based planning

for disaster mitigation," in *Recent Advances in Design and Decision Support Systems in Architecture and Urban Planning*, pp. 193–209, Springer, 2004.

[20] S. G. Berjak and J. W. Hearne, "An improved cellular automaton model for simulating fire in a spatially heterogeneous savanna system," *Ecological Modelling*, vol. 148, no. 2, pp. 133–151, 2002.

[21] A. Alexandridis, D. Vakalis, C. I. Siettos, and G. V. Bafas, "A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through spetses island in 1990," *Applied Mathematics and Computation*, vol. 204, no. 1, pp. 191–201, 2008.

[22] A. Alexandridis, L. Russo, D. Vakalis, G. Bafas, and C. Siettos, "Wildland fire spread modelling using cellular automata: evolution in large-scale spatially heterogeneous environments under fire suppression tactics," *International Journal of Wildland Fire*, vol. 20, no. 5, pp. 633–647, 2011.

[23] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of modern physics*, vol. 55, no. 3, p. 601, 1983.

[24] G. A. Trunfio, "Predicting wildfire spreading through a hexagonal cellular automata model," in *Cellular Automata*, pp. 385–394, Springer, 2004.

[25] X. Liu, J. Zhang, Z. Tong, and Y. Bao, "Grassland fire disaster spreading simulation based on cellular automata," in *Recent Advances in Computer Science and Information Engineering*, pp. 623–628, Springer, 2012.

APPENDIX A

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define LENGTH 100  // area length
#define WIDTH 100  // area width
#define TIME 15 // total time of simulation in timesteps

int main(int argc, char *argv[]) {

FILE *fire = fopen("fire_simulation.txt", "w+") ;
if (fire == NULL){
printf( "\nError opening file! \n") ;
}

// variables' declaration and initialization
int i = 0, j = 0, r = 0, c = 0, l = 2, angle_w = 180, con1 = 0.045, con2 =
    0.131, V= 9, timestep = 0, stateMatrix [LENGTH][WIDTH] = {0},
   nextStateMatrix [LENGTH][WIDTH] = {0};
int vegMatrix [LENGTH][WIDTH] = {0}, denMatrix [LENGTH][WIDTH] = {0},
   slopeMatrix [LENGTH][WIDTH] = {0};
char forestMatrix [LENGTH][WIDTH] = {0};
float p = 1, p_burn = 1, p_h = 0.58, p_veg = 1, p_den = 1, p_w = 1, p_s =
   1, a = 0.078;

// calculation of wind probability
p_w = exp(con1 * V) * exp(V * con2 * (cos(angle_w) - 1));

srand((int)time(NULL));

// rendomly generating trees in an area grid with size LENGTH x WIDTH and
   initializing vegetation type and density
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    slopeMatrix[i][j] = (rand () % 10) + 1;
    i = rand() % LENGTH;
    j = rand() % WIDTH;

```

```
    if (forestMatrix[i][j] == 0){
      forestMatrix[i][j] = '*';
      stateMatrix[i][j] = 1;
      nextStateMatrix[i][j] = 1;
      vegMatrix[i][j] = (rand () % 3) + 1;
      denMatrix[i][j] = (rand () % 3) + 1;
    }
  }
}
fprintf(fire, "A Randomly Generated Forest Environment...\n\n");

// printing forest grid
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    printf("%c ", forestMatrix[i][j]);
    fprintf(fire, "%c ", forestMatrix[i][j]);
  }
  printf("\n");
  fprintf(fire, "\n");
}
printf("\n\n\n");
fprintf(fire, "\n\n\n");

// printing the vegetation type matrix for each cell
fprintf(fire,"Vegetation Matrix (1 = Plants, 2 = Bushes, 3 = Trees) \n");
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    fprintf(fire,"%d ", vegMatrix[i][j]);
    printf("%d ", vegMatrix[i][j]);
  }
  fprintf(fire,"\n");
  printf("\n");
}
fprintf(fire,"\n\n\n");
printf("\n\n\n");

// printint the vegetation density matrix for each cell
fprintf(fire,"Density Matrix (1 = Sparce, 2 = Normal, 3 = Dense) \n");
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    fprintf(fire,"%d ", denMatrix[i][j]);
    printf("%d ", denMatrix[i][j]);
  }
  fprintf(fire,"\n");
```

```
    printf("\n");
}

fprintf(fire,"\n\n\n");
printf("\n\n\n");

// printint the slope matrix for each cell
fprintf(fire,"Slope Matrix (1 - 10 possible ground elevations) \n");
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    fprintf(fire,"%d ", slopeMatrix[i][j]);
    printf("%d ", slopeMatrix[i][j]);
  }
  fprintf(fire,"\n");
  printf("\n");
}

fprintf(fire,"\n\n\n");
printf("\n\n\n");

printf("Fire is ignited somewhere in the forest.\n\n");

//Somewhere in the forest a fire is ignited

i = rand() % LENGTH;
j = rand() % WIDTH;

if(stateMatrix[i][j] == 1){
  stateMatrix[i][j] = 2;
  forestMatrix[i][j] = '^';
}
else{
  while (stateMatrix[i][j]!=1){
  i = rand() % LENGTH;
  j = rand() % WIDTH;
  if(stateMatrix[i][j] == 1){
    stateMatrix[i][j] = 2;
    forestMatrix[i][j] = '^';
    break;
    }
  }
}

fprintf(fire,"----------------------------------------\n\n");
```

```
fprintf(fire,"Fire Spread Simulation in the specified Forest!!!\n\n\n");
fprintf(fire,"-----------------------------------------\n\n");

fprintf(fire,"Fire Ignition somewhere randomly in the Forest...\n\n
    ------------------------------(Timestep = %d)\n", timestep);

// printing state forest grid
for (i=0; i<LENGTH; i++){
  for(j=0; j<WIDTH; j++){
    printf("%d ", stateMatrix[i][j]);
    fprintf(fire, "%c ", forestMatrix[i][j]);
  }
  printf("\n");
  fprintf(fire,"\n");
}
fprintf(fire,"\n");
printf("\n\n\n");

printf("Fire starts speading in the Forest...\n\n");
fprintf(fire, "Fire spread for each passing timestep:\n\n");

while(timestep != TIME){
  fprintf(fire, "\n------------------------------(Timestep = %d)\n", timestep
   +1);

    // Rules of Transition
    for (i=0; i<LENGTH; i++){
      for(j=0; j<WIDTH; j++){
      if(stateMatrix[i][j]==0 || stateMatrix[i][j]==3){
      // state remains the same
      nextStateMatrix[i][j] = stateMatrix[i][j];
      continue;
      }
      else if(stateMatrix[i][j]==1){
      if(stateMatrix[i-1][j-1]==2 || stateMatrix[i-1][j]==2 || stateMatrix
   [i-1][j+1]==2 || stateMatrix[i][j-1]==2 ||
      stateMatrix[i][j+1]==2 || stateMatrix[i+1][j-1]==2 || stateMatrix[i
   +1][j]==2 || stateMatrix[i+1][j+1]==2){
      for (r=i-1; r<=i+1; r++){
      for(c=j-1; c<=j+1; c++){
      if(stateMatrix[r][c]==2){
      p=1;
      p_burn=1;
      p_s = 1;
```

```
164       // Calculation of p_veg probability
          if(vegMatrix[i][j] == 1){
166         p_veg = -0.3;
          }
168       else if(vegMatrix[i][j] == 2){
            p_veg = 0;
170       }
          else if(vegMatrix[i][j] == 3){
172         p_veg = 0.4;
          }
174
          // Calculation of p_den probability
176       if(denMatrix[i][j] == 1){
            p_den = -0.4;
178       }
          else if(denMatrix[i][j] == 2){
180         p_den = 0;
          }
182       else if(denMatrix[i][j] == 3){
            p_den = 0.3;
184       }

186       // Calculation of p_s probability
          if(stateMatrix[i-1][j]==2){
188         p_s = exp(a * atan((slopeMatrix[i-1][j] - slopeMatrix[i][j]) / l))
    ;
          }
190       else if(stateMatrix[i][j+1]==2){
            p_s = exp(a * atan((slopeMatrix[i][j+1] - slopeMatrix[i][j]) / l))
    ;
192       }
          else if(stateMatrix[i][j-1]==2){
194         p_s = exp(a * atan((slopeMatrix[i][j-1] - slopeMatrix[i][j]) / l))
    ;
          }
196       else if(stateMatrix[i+1][j]==2){
            p_s = exp(a * atan((slopeMatrix[i+1][j] - slopeMatrix[i][j]) / l))
    ;
198       }
          else if(stateMatrix[i-1][j-1]==2){
200         p_s = exp(a * atan( (slopeMatrix[i-1][j-1] - slopeMatrix[i][j]) /
    (l * sqrt(2)) ) );
          }
```

```
202         else if(stateMatrix[i-1][j+1]==2){
              p_s = exp(a * atan( (slopeMatrix[i-1][j+1] - slopeMatrix[i][j]) /
    (1 * sqrt(2)) ) );
204         }
            else if(stateMatrix[i+1][j-1]==2){
206           p_s = exp(a * atan( (slopeMatrix[i+1][j-1] - slopeMatrix[i][j]) /
    (1 * sqrt(2)) ) );
            }
208         else if(stateMatrix[i-1][j-1]==2){
              p_s = exp(a * atan( (slopeMatrix[i+1][j+1] - slopeMatrix[i][j]) /
    (1 * sqrt(2)) ) );
210         }

212         // Calculation of total p_burn probability
            p_burn = p_h * (1 + p_veg) * (1 + p_den) * p_w * p_s;
214
            p = ((float)rand()/(float)RAND_MAX);
216
            if (p >= p_burn){
218           //burnMatrix[i][j] = '>'; // updating burnMartix with the symbol >
              nextStateMatrix[i][j]=2;
220           break;
            }
222         }
            else{
224           // state remains the same
              continue;
            }
      }
228   }
}
230
    else{
232   // state remains the same
      continue;
    }

236 }
    else if(stateMatrix[i][j]==2){
238   nextStateMatrix[i][j] = 3;
      continue;
240   }
    }
242 }
```

```
244 fprintf(fire,"\n\n");

246 // printing forest grid after each fire spreading timestep
for (i=0; i<LENGTH; i++){
248   for(j=0; j<WIDTH; j++){
      if(stateMatrix[i][j]==3){
250     forestMatrix[i][j]='-';
      }
252   else if(stateMatrix[i][j]==2){
        forestMatrix[i][j]='^';
254   }
      else if(stateMatrix[i][j]==0 || stateMatrix[i][j]==1){
256     // forestMatrix cell state remains the same
      }
258   printf("%c ", forestMatrix[i][j]);
      fprintf(fire, "%c ", forestMatrix[i][j]);
260 }
    printf("\n");
262 fprintf(fire,"\n");
}
264
// updating forest grid to next state
266 for (i=0; i<LENGTH; i++){
    for(j=0; j<WIDTH; j++){
268   stateMatrix[i][j] = nextStateMatrix[i][j];
    }
270 printf("\n");
}
272
timestep = timestep + 1;
274 printf("\n\n");
}
276
printf("That's all folks!");
278
fclose(fire);
280
return 0;
282 }
```

Listing 5: Total Simulation program code