Final Project
Extending a Wildfire Cellular Automata Model

Finn Joel Bjervig[*], August Forsman[†], and Erik Turesson[‡]

1MA256 - Modeling Complex Systems,
Department of Mathematics,
Uppsala university, Sweden

August 31, 2022

[*]Electronic address: `jobj8920@student.uu.se`
[†]Electronic address: `aufo8456@student.uu.se`
[‡]Electronic address: `ertu2293@student.uu.se`

1 Model Description

Our model is mainly based on one created by Alexandridis et. al. in their paper *A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through Spetses Island in 1990.* [1] Here, cellular automata are connected in a 2-dimensional square grid. Each square is representative of an area of land, containing a state describing its condition. In their model, the 4 possible states represent; cells with no burnable material, cells with burnable material, cells currently on fire, and cells that have already burnt. In this report, we introduce some additional states to differentiate between varying terrains. Our states are as follows:

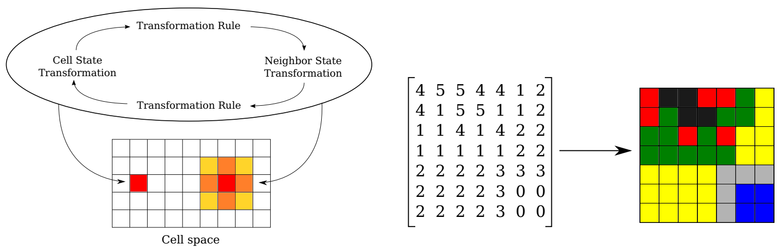
- State 0: The cell contains no burnable material, making fire unable to spread there. Thus, any cells in this state will never change states. Examples of this could be water or rocky terrain.
- State 1: The cell contains burnable forest that has not yet been ignited.
- State 2: The cell contains burnable farmland that has not yet been ignited.
- State 3: The cell contains urban buildings that have not yet been ignited.
- State 4: The cell contains burnable material and is currently on fire. Will go to State 6 by the next time step.
- State 5: The cell once contained burnable material that is now burned into char, thus it cannot catch fire again.

The grid of states is stored as a matrix $\mathbf{S}$, with coordinates corresponding to the grid displayed in Figure 1. It should be noted that some features of the original model have been excluded from this project. Mainly, the original model included varying vegetation densities and types affecting the probabilities. However, since this would require detailed data about the terrain, we opted to instead introduce the additional states seen above to capture a similar phenomenon. The probability of each burnable cell catching fire is denoted as $p_{terrain}$.

1.1 Rules of Evolution

In each time-step t, the state of the cell at i, j can evolve into another state in the next time-step $t+1$ with the following rules:

- Rule 1: If $\mathbf{S}(i,j,t) = 0$, then $\mathbf{S}(i,j,t+1) = 0$: A cell containing no burnable material will never burn.
- Rule 2: If $\mathbf{S}(i,j,t) = 1, 2, \text{or } 3$, then for each $\mathbf{S}(m,n,t) = 4$, $m, n \in [-1, 1]$, $i, j \neq m, n$: $\mathbf{S}(i,j,t+1) = 4$ with probability p_{burn}. Otherwise remains in its current state.
- Rule 3: If $\mathbf{S}(i,j,t) = 4$, then $\mathbf{S}(i,j,t+1) = 5$: A cell currently on fire will be burnt out in the next time-step.
- Rule 4: If $\mathbf{S}(i,j,t) = 5$ then $\mathbf{S}(i,j,t+1) = 5$: A cell previously burnt cannot catch fire once again.



(a) Transformation rules for cells and neighbors in CA [2]

(b) Matrix states with corresponding pixel value in image space

Figure 1

1.2 Impact of the Speed and Direction of Wind

To simulate the weather conditions during the Spetses wildfire, the wind was set to a static 10 m/s. A probability of changes in both speed and direction is implemented to analyze how the model responds to dynamic changes. To model a varying wind direction, it can be chosen to occur at random on an interval on the unit disk, for example, $[0, \frac{\pi}{2}]$ which corresponds to directions between east and north. During an iteration, the wind speed and its direction can be changed if a randomly generated value does not exceed the previous value by a certain tolerance. Abiding by the rules of the original model, a probability p_{wind} is calculated to determine how the wind will impact the spread of the fire. This probability is calculated as

$$p_{wind} = e^{c_1 V + c_2 V(\cos\theta - 1)}, \tag{1}$$

where c_1, c_2 are constants and θ is the relative angle between the direction of the wind and the fire propagation. Thus, if fire is trying to spread in the same direction as the wind, $\theta = 0$ and $p_{wind} > 1$.

1.3 Spotting

Flying through the wind, burning debris from fires can have a significant impact on the spread since the propagation is no longer limited by barriers that do not catch fire, or have a very low probability of catching fire. Pines, for example, will have their pine cones ignited like popcorn .And due to the weight and shape of the cone, it will not be affected by the wind to the extent that light debris like leaves and embers will, so pine cones will not change their trajectory too much during flight, which would be computationally cumbersome to take into account. Therefore light burning debris which could travel very long distances, is not included in the model. So we focus on out pine cones. The pine cones blasting from a burning forest cell is modelled as follows: A cell being burned (state 4) will blast away a number of cones drawn from a Poisson probability density function with expectation value λ

$$N_p \sim \text{Pois}(\lambda)$$

For a reasonable spread of the fire, the expectation value may lie around $\lambda = 2 - 4$. In the simulations, it's 2.

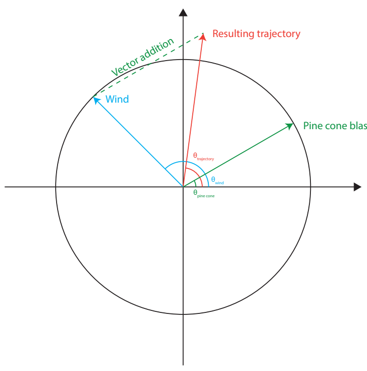


Figure 2: By adding the x and y components of respective direction, one yield a resulting direction that may be favourable in modeling the winds effect on the trajectory of the "popped" pine cone

Now when the number of pine cones that will "pop" has been defined, the cell it will travel to must be established. Due to the implementation of wind, this aspect must be taken into account, because the popped cones are flying through the air and are affected by the wind. However, it is beyond the scope of this model to simulate any sort of trajectory of greater detail. In the original paper, [1], the direction of the blast is simply drawn from a random distribution, and the distance is dependent on the relative angle of the wind and the blast angle of the pine cone. A minor extension made in this model is that the trajectory angle is instead the unit vector addition of the pine cone blast angle and the wind direction as depicted in figure 2.

$$\hat{ds} = \begin{bmatrix} \cos(\theta_{pinecone}) \\ \sin(\theta_{pinecone}) \end{bmatrix} + \begin{bmatrix} \cos(\theta_{wind}) \\ \sin(\theta_{wind}) \end{bmatrix} \tag{2}$$

The function for the spotting, as it's called, will return a vector of the number of cells in the x and y direction for which the pine cone will travel

$$\vec{ds} = \begin{bmatrix} dx \\ dy \end{bmatrix} = r \left(\begin{bmatrix} \cos(\theta_{pinecone}) \\ \sin(\theta_{pinecone}) \end{bmatrix} + \begin{bmatrix} \cos(\theta_{wind}) \\ \sin(\theta_{wind}) \end{bmatrix} \right) \tag{3}$$

$$r = r_n e^{v_{wind} c_2 \cos(\theta_{wind} + \theta_{pinecone}) - 1} \tag{4}$$

$$r_n \sim \mathcal{N}(\mu, \sigma) \tag{5}$$

$\mu = 0$ and $\sigma^2 = 10$ is the expectation value and variance for the normal distribution. The result of this implementation is that now the fire can spread beyond the rivers on the map, as seen in figure 3.

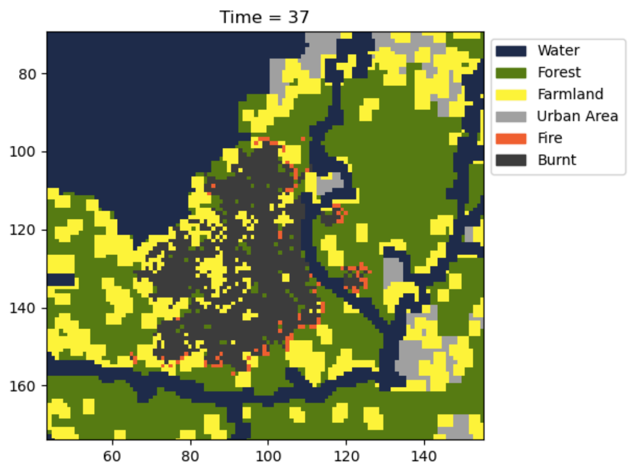


Figure 3: With spotting enabling the fire to spread beyond rivers on the map, the total number of burnt cells is increased. The wind direction is west-southwest and the wind speed 3.2 m/s

A drawback of this implementation is that it does not take into account what state the fire cell was before ignition, as it could have been a farmland or urban cell, where pine cones are scares or even non-existent. To mend this pitfall, one would have to check the previous state before the cell became ignited in that case

1.4 Final Model

With these features, the resulting probability of a cell spreading fire to a neighbour, p_{burn}, will be:

$$p_{burn} = p_{terrain} \cdot p_{wind} \tag{6}$$

It should be noted that these approaches allow for probabilities of igniting above 100%, since both the multipliers stemming from wind and humidity can be above one. It is possible that accuracy could be improved by applying a function similar to a softmax or a sigmoid function to the probabilities that would be more realistic in the extremes. This would compress all probabilities to the interval $[0, 1]$ while not impacting intermediate probabilities as much.

In <figgy 63>, the arrow displays wind magnitude by length and its direction.

2 Extensions of the Model

2.1 Humidity and Precipitation

A major impact on the spread of a wildfire is the soil water content.[3] A cell with large amounts of water should be less likely to be set on fire. Humidity can stem from multiple different sources, such as proximity to a river or a lake. We have implemented a simple method, mainly aimed at including recent precipitation, but that could be used to simulate geographical features or droughts. This is done by introducing another matrix where each element is a factor, $p_{humidity}(i, j)$, corresponding to the relative probability of that cell catching fire. A factor of 1 would mean normal conditions, whereas a factor of 0.5 would mean that the cell is half as likely to catch fire as if it was dry. This also allows for values greater than 1, which could be used to simulate extremely dry areas more likely to burn. In our implementation, we assume these values to be constant and not change with the evolution of the cellular automata, but introducing an update should be relatively easy. As wildfires can spread over longer periods of time it would of course be more realistic if something like this was implemented, but it was deemed as outside the scope for this project. With this extension, the probability of catching fire, p_{burn}, is now:

$$p_{burn} = p_{terrain} \cdot p_{humidity} \cdot p_{wind} \tag{7}$$

3 Simulation Results

To understand the behaviour of the system, we recorded and visualised simulations of the wildfire. Constants used are recorded in Table 1. An initial wind

Table 1: Parameter values used in simulations in this section

p_{forest}	$p_{farmland}$	p_{urban}	c_1	c_2
0.4	0.2	0.1	0.08	0.13

speed of 4 m/s in an eastern direction was used. In addition, we added two regions of deviating humidity. In Figure 4, the red rectangle marks an area of drought, with $p_{humidity} = 1.7$, and the blue rectangle marks an area where of higher humidity with $p_{humidity} = 0.6$. By looking at the simulation, it can be seen that the fire is unable to spread far into the rectangle of recent precipitation before dying out. Simultaneously, the wildfire seems to spread at a more rapid pace inside of the dry region.

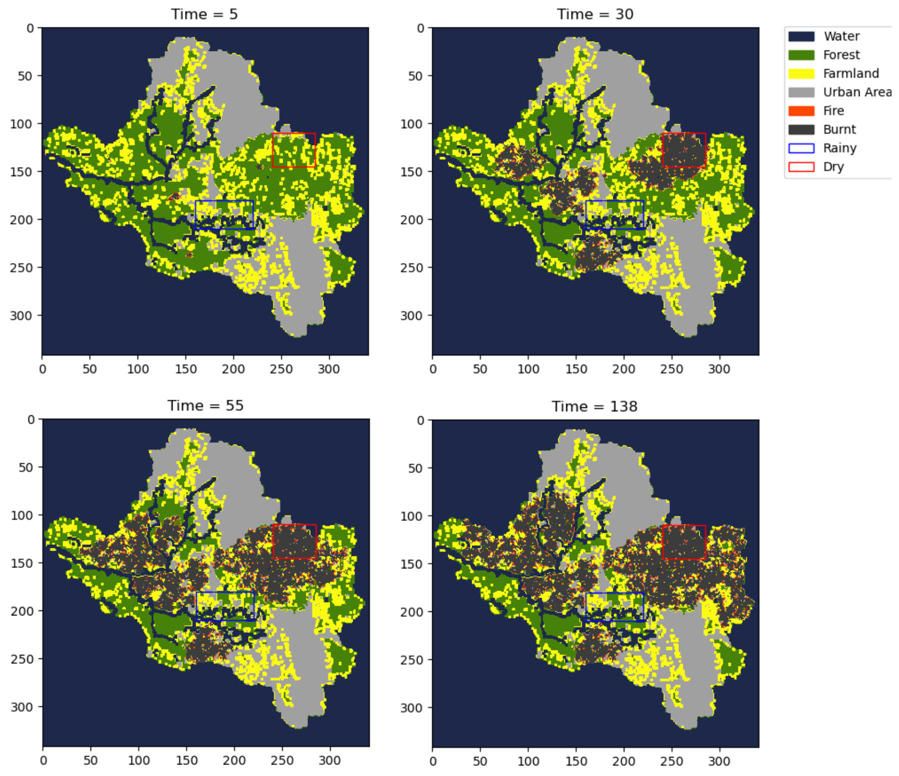


Figure 4: The spread of the fire over time. Depending on the material on the map, there will be a different probability of the fire spreading to that part. More over there are defined regions of drought and rain, which modifies the probabilities for that local area. The wind speed and direction changes a little bit over time, so the arrow in the plot is indicating the general direction (west) and the wind speed lies around 3 m/s throughout the simulation

To measure the evolution of the wildfire, we count the number of cells in each state for each time-step. This allows us to ease the investigation of how parameter changes impact the system, and the severity of the spread. See Figure 5 for the counts using the parameters in Table 1. Looking at the figure 5, an initial rapid spread that then slows down can be seen. Further, notice that while some farmland has been burnt, mainly forest tiles seem to have burnt down, while the urban cells are mostly constant. This is easily explained by the different values of $p_{terrain}$ for these states. The majority of farmland remaining ”unburnt” can also be seen in Figure 4.

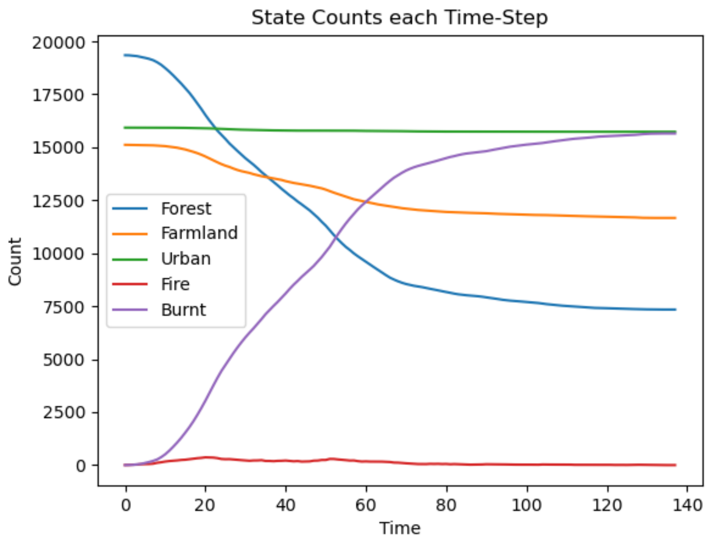


Figure 5: The number of each state over time as the fire spreads thought the map. The curves seen are from the simulation seen in figure 4. There is a striking resemblance between the cell count curves and the population curves of a SIR model. Imagine the fuel (forest and farmland) analogous to the susceptible population, fire cells are infected and burnt cells are recovered.

4 Variation of Parameters

We decided to mainly vary parameters for the wind and the spotting effects of the simulation. The parameters for the terrains were deemed unnecessary to vary, since the only differentiation between the terrains are variations in $p_{terrain}$, thus we already see the impact of these variations during other simulations.

4.1 Direction and Strength of Wind

In Figure 6, the number of cells on fire in each time step is shown for some selected values of different wind speeds. One can see quite clearly that the spread seems to slow down for higher winds. In the previous section, it was shown that under normal conditions, the fire spreads to most forest tiles in each direction. However, when heavier wind is introduced, it makes it harder for the fire to propagate against the wind, while making the propagation along the wind direction easier. This effectively limits the fire from spreading against the wind, the number of burning cells in total will decrease, and also vanish faster once all cells in that direction have been burnt. Other interesting results from these simulations that are not shown in the figures, is that the difference between burning rates for farmland and forests decreases for stronger winds, as the wind may allow for farmland that didn't previously burn to now burn.

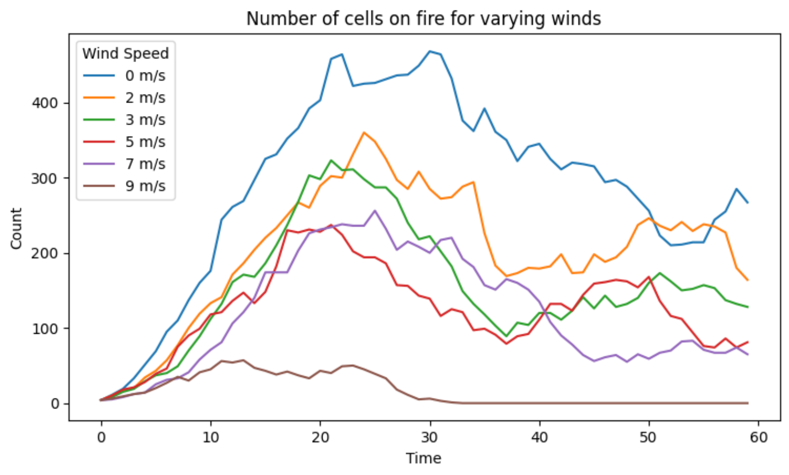
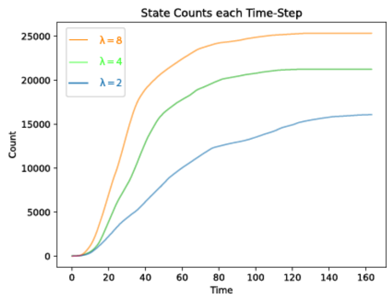


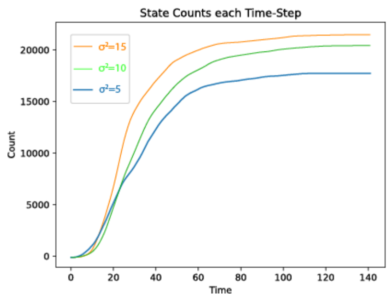
Figure 6: The current number of cells on fire in each time-step for varying wind speeds

4.2 Spotting parameters

There is quite a lot to vary in the spotting rule that affects the fire propagation. First of all, changing the expectation value of the Poisson probability density function from which the number of ignited pine cones are drawn from, will increase the number of burnt cells and the rate of burning, see figure 7. The distance the cones will travel is also scaled with an "amplitude" r that is multiplied to the exponential function. this number is drawn from a normal distribution with a certain expectation value and variance. Here one may change the variance so that the distance traveled by a cone on average increases. This allows longer jumps into areas that may not have been reached otherwise,



(a) λ is the expectation value (of a Poisson distribution) for how many pine cones will be ignited and blaster through the air for a given cell that has caught fire.



(b) σ^2 is the variance of the normal distribution for how far the the pine cone will travel.

Figure 7: Increasing wither of the parameters both result in a higher rate of burning and a higher total number of burnt cells.

meaning the number of burned cells will also increase with increased variance of the normal distribution function. See figure 7. Further, the winds speed and direction affects the trajectory of the pine cone blast.

5 Conclusions

The results from the simulations suggests that environmental factors are of great impact on how wildfires spread. Complex flows of wind due to the heat during the fire and recent precipitation can be detrimental to the modeled outcome. A study by Russo, et al shows that the approach of not modifying the optimized Spetses parameters c_1 and c_2 in Table 1 produces similar results when simulating other wildfires that has occurred on Greek islands with a heterogeneous environment. [4]. In comparison to CFD models that take these complex winds into account, there's an obvious trade-off between expensive hardware and this hypothetically not so precise CA that easily runs on a laptop. burning debris traveling by wind also has a profound effect on the spread of the fire which can now go beyond rivers and reach places that might not have been affected otherwise.

6 Future Work

The extensions done in this report showcased the easy implementation of new features in the cellular automata, and the potential of other extensions that could improve the model. First off, using hexagonal grids instead of a square grid may have more realistic spatial interactions and lead to more accurate results. [5] But this would likely give rise to larger computational inefficiencies as well.

Other extensions could be implemented to improve on the model itself, such as including firefighters or water bombers to research how the stop the spread of fire as efficiently as possible. More geographical and topological information could be utilized to understand how elevations impact the results or how it could lead to accumulations of rain water in certain low spots. As previously mentioned, another refinement of the model would be to apply some smoothing function to handle the probabilities of the extreme ends, to make sure that probabilities are confined within $0 - 100\%$.

References

[1] 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*, 204(1):191–201, 2008.

[2] John von Neumann, Arthur W Burks, et al. Theory of self-reproducing automata, 1966.

[3] Zachary A. Holden, Alan Swanson, Charles H. Luce, W. Matt Jolly, Marco Maneta, Jared W. Oyler, Dyer A. Warren, Russell Parsons, and David Affleck. Decreasing fire season precipitation increased recent western us forest wildfire activity. *Proceedings of the National Academy of Sciences*, 115(36):E8349–E8357, 2018.

[4] L Russo, D Vakalis, and Constantinos Siettos. Simulating the wildfire in rhodes in 2008 with a cellular automata model. *Chemical Engineering Transactions*, 35:1399–1404, 2013.

[5] L. Hernández Encinas, S. Hoya White, A. Martín del Rey, and G. Rodríguez Sánchez. Modelling forest fire spread using hexagonal cellular automata. *Applied Mathematical Modelling*, 31(6):1213–1227, 2007.

Appendix

Links to animations

1. Inital wind speed 4 m/s
2. Inital wind speed 24 m/s

Simulation Code

```python
from tkinter import Label
import matplotlib
matplotlib.use('TkAgg')
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from PIL import Image
import random as rd
import math

pici = 0
rd.seed(10)

pr_fr = 0.4     # probability of fire for forest
pr_fl = 0.2     # probability of fire for farmland
pr_ur = 0.1     # probability of fire for urban env
                # if we introduce wind, they need to travel in wind
    direction
pr_rain = 0.6   # fraction of likelyhood that a cell ignites after rain
pr_draught = 1.7
L = 2 #expectation of value of poisson distribution for number of
    pinecones igniting

colors = [np.divide((30, 40, 75),255), np.divide((70, 130, 10),255), np
    .divide((250, 250, 20),255), np.divide((160,160,160),255), np.
    divide((255,70,0),255) ,np.divide((60,60,60),255)]
n_bin = len(colors)
cmap = LinearSegmentedColormap.from_list('landscape', colors, N=n_bin)

c_1 = 0.045
c_2 = 0.131
def wind_spread_prod(wind_angle, wind_speed):
    wind_probs = np.zeros((3,3))
    for x in [-1, 0, 1]:
        for y in [-1, 0, 1]:
            relative_angle = wind_angle+math.atan2(y,x)
            wind_probs[x+1,y+1] = math.exp(c_1*wind_speed) * math.exp(
    wind_speed*c_2*(math.cos(relative_angle)-1))
    return wind_probs

def spotting(wind_angle, wind_speed):
    rn = np.random.normal(0,5) # max distance traveled is 7

    pinecone_blast_angle = np.random.uniform(0,2*math.pi)
    relative_angle = pinecone_blast_angle - wind_angle
    r = rn*math.exp(wind_speed*c_2*math.cos(relative_angle)-1)
    #return np.round([r*math.cos(pinecone_blast_angle), r*math.sin(pinecone_blast_angle)])
    return np.round(np.multiply(r,[math.cos(pinecone_blast_angle) + math.cos(wind_angle) , math.sin(pinecone_blast_angle) + math.sin(wind_angle)]))

wind_dir = rd.random()*math.pi #0: Wind towards North, pi/2: east, pi: south, 3pi/2: west
speed = 4.0 #m/s

def initialize():
    global time, config, nextConfig, dim, wind_probs, wind_dir, rain
    time = 0

    # img = Image.open('finalproj/topo1.png').convert('RGB')
    img = Image.open('topo1.png').convert('RGB')
    config = np.array(img, dtype=np.int8)[:,:,0]
    dim = config.shape[1]

    config[146,225] = 4
    config[137,70] = 4
    config[236,152] = 4
    nextConfig = np.zeros([dim,dim], dtype=np.int8)
    config[177,140] = 4 # added

    wind_probs = wind_spread_prod(wind_dir, speed)

    rain = np.ones(shape=(dim,dim))
    rain[180:210, 160:220] = pr_rain    #recent rain in this area
    rain[110:145, 240:285] = pr_draught #draught in this area


def observe():
    global time, config, nextConfig, pici, wind_dir
    plt.imshow(config, cmap = cmap)
    #plt.arrow(275, 60, 40*math.cos(wind_dir-math.pi/2), 40*math.sin(wind_dir-math.pi/2), length_includes_head = True,head_width = 6, color = 'red')
    plt.title(f'Time = {time}')



    patches = [matplotlib.patches.Patch(color=cmap(0.2*i)) for i in range(6)]
    patches.append(matplotlib.patches.Rectangle((1,1),1,1, edgecolor='b', fill=False))
    patches.append(matplotlib.patches.Rectangle((1,1),1,1, edgecolor='r', fill=False))
    labs = ["Water", "Forest", "Farmland", "Urban Area", "Fire", "Burnt", "Rainy", "Dry"]
    plt.legend(handles=patches, labels=labs, bbox_to_anchor=(1.0,1), loc=2)
    plt.annotate(f"Wind Speed: {speed:.2f} m/s", xy=((350, 200)), annotation_clip=False)
    plt.arrow(410, 270, 10*math.cos(wind_dir-0.5*math.pi), 10*math.sin(wind_dir-0.5*math.pi), clip_on=False, width=3)
    plt.xlim(0,340)
```

```python

    #rectangle for rain
    ax = plt.gca()
    ax.add_patch(matplotlib.patches.Rectangle((160,180), 60, 30, linewidth=1, edgecolor='b', fill=False))
    ax.add_patch(matplotlib.patches.Rectangle((240,110), 45, 35, linewidth=1, edgecolor='r', fill=False))
    plt.savefig(f'plots/plot{pici}.png')
    pici += 1


total_counts = []

def update():
    global time, config, nextConfig,dim, wind_probs, speed, \
    speed_temp, wind_dir, wind_dir_temp, wind_tol, speed_tol, dim
    wind_tol = 0.2
    speed_tol = 0.25
    wind_dir_temp = rd.random()*2*math.pi
    speed_temp = 10 * rd.random() #generate random [0,10]
    if math.fabs(wind_dir_temp-wind_dir) < wind_tol:
        wind_dir = wind_dir_temp
    if math.fabs(speed_temp-speed) < speed_tol:
        speed = speed_temp
    plt.cla()
    time = time + 1
    print(time)
    # print(wind_probs)
    # 0: Water
    # 1: Forest
    # 2: Farmland
    # 3: houses/city/urban env
    # 4: Fire
    # 5: Burnt

    for x in range(dim):
        for y in range(dim):
            state = int(config[x, y])
            #print(state)
            if state == 1:      # if any neighboring cell is on fire prob pr_fr*pr_w this cell catch on fire
                for dx in range(-1,2):
                    for dy in range(-1,2):
                        if((config[x+dx,y+dy] == 4) and (rd.random()<= pr_fr*wind_probs[dx+1,dy+1]*rain[x,y])):
                            state = 4
            elif state == 2:    # if any neighboring cell is on fire prob pr_fl*pr_w this cell catch on fire
                for dx in range(-1,2):
                    for dy in range(-1,2):
                        if((config[x+dx,y+dy] == 4) and (rd.random()<= pr_fl*wind_probs[dx+1,dy+1]*rain[x,y])):
                            state = 4
            elif state == 3:
                for dx in range(-1,2):
                    for dy in range(-1,2):
                        if((config[x+dx,y+dy] == 4) and (rd.random()<=
    pr_ur*wind_probs[dx+1,dy+1]*rain[x,y])):
                            state = 4

            elif state == 4:
                # how many pinecones N ignited
                N = np.random.poisson(L)
                for i in range(N):
                    ds = spotting(wind_dir,speed)
                    xhop = int(x+ds[0])
                    yhop = int(y+ds[1])
                    if((xhop<dim and xhop >= 0) and (yhop<dim and yhop
    >= 0)):
                        if( ((config[xhop,yhop] == 1) and (rd.random()<
    pr_fr*rain[xhop,yhop])) or
                            ((config[xhop,yhop] == 2) and (rd.random()<
    pr_fl*rain[xhop,yhop])) or
                            ((config[xhop,yhop] == 3) and (rd.random()<
    pr_ur*rain[xhop,yhop])) ):
                            nextConfig[xhop,yhop] = 4
                state = 5       # become burnt
            nextConfig[x, y] = state
    config, nextConfig = nextConfig, config
    counts = [0,0,0,0,0,0] #6 states
    for x in range(dim):
        for y in range(dim):
            counts[nextConfig[x,y]]+=1
    total_counts.append(counts)

import pycxsimulator
pycxsimulator.GUI().start(func=[initialize, observe, update])


plt.figure()
# np.arange(0,len(total_counts),1)
plt.plot(np.arange(0,len(total_counts),1), [total_counts[i][1] for i in
     range(len(total_counts))], label="Forest")
plt.plot(np.arange(0,len(total_counts),1), [total_counts[i][2] for i in
     range(len(total_counts))], label="Farmland")
plt.plot(np.arange(0,len(total_counts),1), [total_counts[i][3] for i in
     range(len(total_counts))], label="Urban")
plt.plot(np.arange(0,len(total_counts),1), [total_counts[i][4] for i in
     range(len(total_counts))], label="Fire")
plt.plot(np.arange(0,len(total_counts),1), [total_counts[i][5] for i in
     range(len(total_counts))], label="Burnt")
plt.legend()
plt.title("State Counts each Time-Step")
plt.xlabel("Time")
plt.ylabel("Count")
plt.savefig("statecount.png")
```