Gefen Efrati

CMPSC-132 Project 2 Report

Description & Significance of the Project

This project is designed to simulate the spread of a wildfire in a forest environment. The simulation attempts to model fire propagation through a grid of pixels that can represent trees at varying altitudes, as well as incombustible objects modeling things like rocks or firebreaks. The simulation is also designed to respond to wind direction variability (indicated by the yellow arrow in the top right of the screen). The project is intended to be easily accessible and used for educational purposes; this accessibility is enabled by an intuitive graphical representation of the fire spread as well as a concluding statistical breakdown of the fire at the end of the simulation. The parameters are also easily found and modifiable in the code to better represent more realistic or specific initial conditions. The significance of this project lies in the fact that wildfires behave highly chaotically and are very complex to effectively analyze. Fire will spread faster in the direction wind blows, and faster uphill, but slower in the converse cases. Perlin noise was also implemented to better model the natural terrain variability of hills and valleys we see in the real world.

Instructions

Installation Steps:

1. Clone or download the repository containing the code.
2. Install python 3.13, if you do not already have it installed.
3. Install the required dependencies using `pip`:
   ```
   pip install pygame numpy perlin-noise
   ```

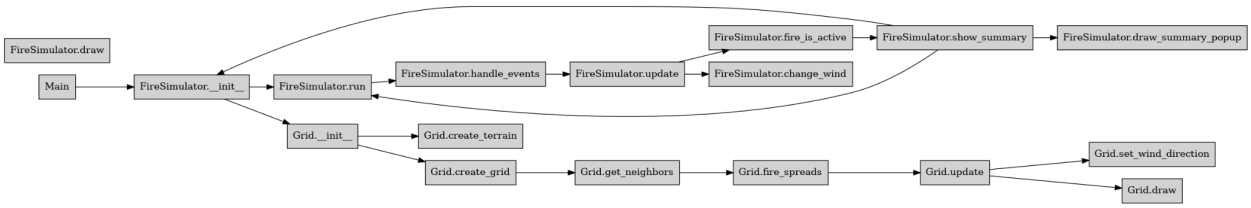
Running the Simulation:

1. Navigate to the directory containing the script.
2. Run the Python script:
   ```
   python wildfire_simulation.py
   ```

Structure

The FireSimulator class manages the simulation by initialization of the grid and event handling. It coordinates the interactions between the background processes and the visual output, as well as handles wind direction changes, and simulation statistics tracking.

The Grid class is responsible for representing the trees, fire, and terrain; it determines the behavior of the fire spreading based on terrain variability and updates the states of cells accordingly.



Class Summary

Grid Class

The Grid class is responsible for representing and managing the grid of trees, fire, and terrain in the wildfire simulation.

Attributes:

size (int): The size of the grid (e.g., 50 for a 50x50 grid).
spread_chance (float): The chance for fire to spread to neighboring trees.
wind_direction (str): The direction of the wind affecting fire spread (e.g., 'N', 'E', 'S', 'W').
grid (np.ndarray): A 2D array representing the grid where each cell contains the state of the cell (e.g., empty, tree, fire).
terrain (np.ndarray): A 2D array representing the terrain, generated using Perlin noise, which affects fire spread based on elevation.

Methods:
__init__(self, size, spread_chance, wind_direction): Initializes the grid, terrain, and attributes for fire spread, wind direction, and grid size.
create_terrain(self): Creates terrain using Perlin noise to simulate variations in elevation that affect fire spread.
create_grid(self): Initializes the grid with trees and places fire in the center of the grid.
get_neighbors(self, y, x): Returns the coordinates of neighboring cells (up, down, left, right).
fire_spreads(self, y, x, ny, nx): Determines whether fire will spread from one cell to its neighbor based on factors like wind, terrain, and spread chance.
update(self): Updates the grid by advancing the fire state (spreading, aging, or burning out cells).
draw(self, screen): Draws the grid on the screen using Pygame, coloring each cell based on its state (empty, tree, fire, burned).
set_wind_direction(self, new_direction): Changes the direction of the wind, influencing fire spread.

FireSimulator Class

The FireSimulator class is the main controller of the simulation, managing the interaction between the Grid and the Pygame screen.

Attributes:

grid_size (int): The size of the grid (e.g., 50 for a 50x50 grid).
cell_size (int): The size of each cell in the grid (e.g., 14).
screen (pygame.Surface): The Pygame screen where the simulation is rendered.
clock (pygame.time.Clock): The Pygame clock object used to control the frame rate.
fps (int): The frames per second for the simulation.
wind_direction (str): The current wind direction, which influences fire spread.
grid (Grid): The instance of the Grid class representing the grid of trees, fire, and terrain.
running (bool): A flag indicating whether the simulation is running.
last_wind_change (float): The timestamp of the last wind change event.
wind_change_interval (int): The interval (in seconds) between wind direction changes.
simulation_start_time (float): The timestamp of when the simulation started.
wind_change_count (int): The number of times the wind direction has changed during the simulation.

Methods:

__init__(self, grid_size, cell_size, fps): Initializes the simulation with the given grid size, cell size, and frames per second.
draw_legend(self): Draws a legend on the screen explaining the meaning of different colors in the grid (e.g., tree, fire, burned).
draw_wind_arrow(self): Draws an arrow on the screen to represent the current wind direction.
rotate_points(self, points, center, angle): Rotates a set of points around a center by a given angle (used to rotate the wind direction arrow).
run(self): Runs the simulation loop, which includes handling events, updating the grid, and drawing the simulation.
handle_events(self): Handles Pygame events like quitting the simulation or user input.
update(self): Updates the grid and fire state, including wind changes and checking if the fire is still active.
draw(self): Draws the grid, legend, and wind arrow on the screen.
change_wind(self): Changes the wind direction at regular intervals.
fire_is_active(self): Checks if there are any active fire cells in the grid.
show_summary(self): Displays a summary of the simulation, including information like the number of trees burned, the wind changes, and elapsed time.
draw_summary_popup(self, lines): Draws a popup window displaying the simulation

summary with options to restart or exit the simulation.
restart_simulation(self): Restarts the simulation by resetting the grid and other parameters.

Example execution results

Fig 1. An unterminated instance of the simulation.

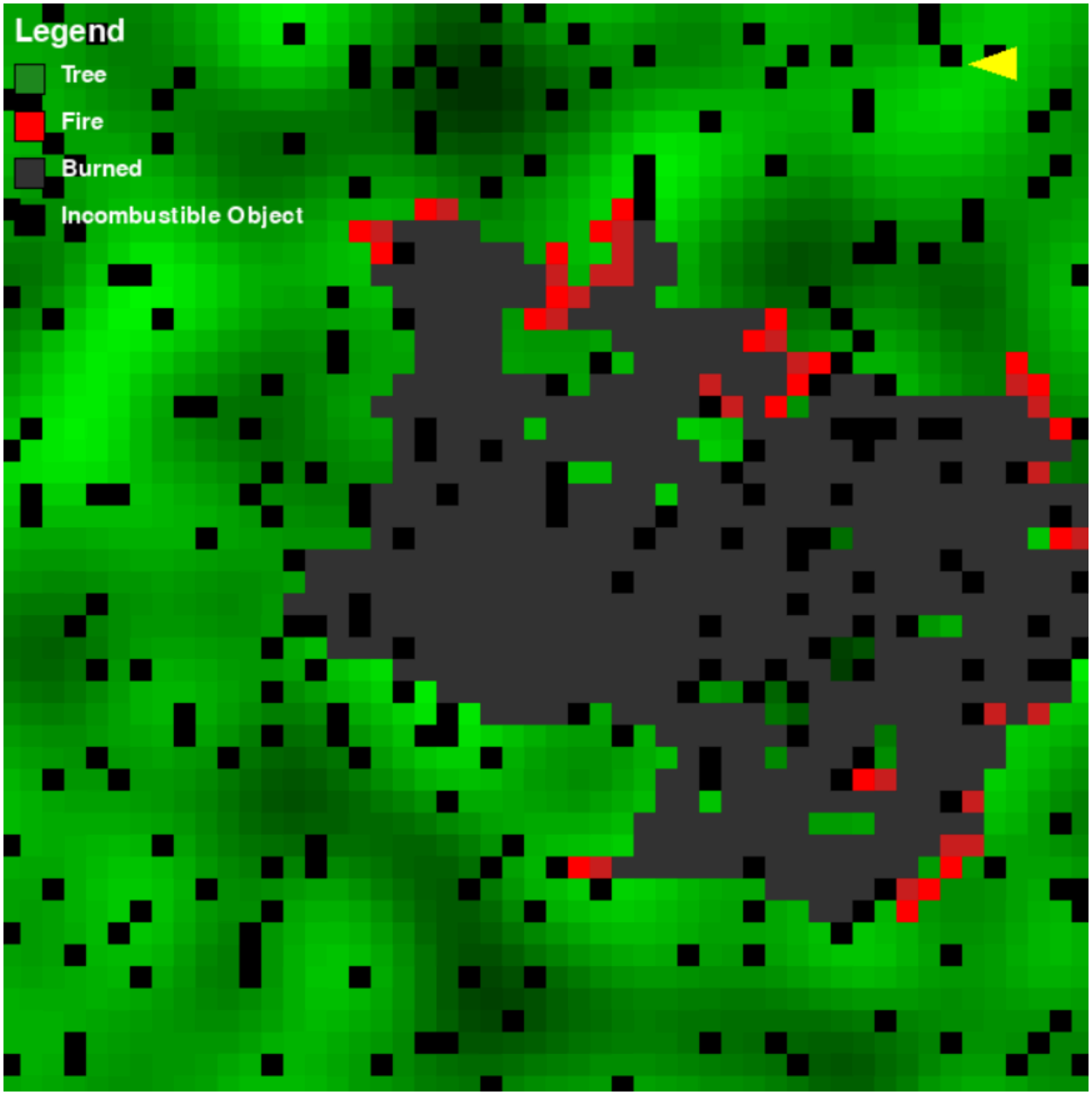
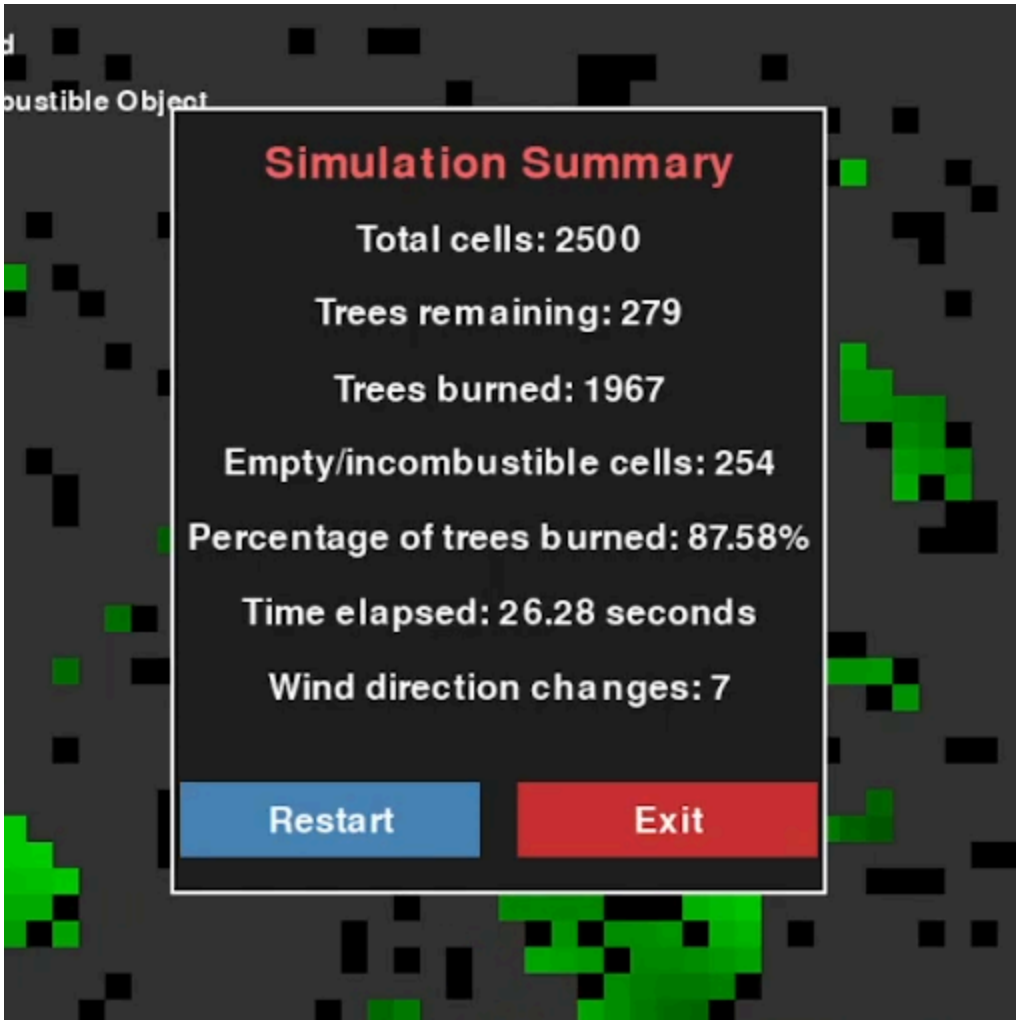


Fig 2. A statistical summary of the simulation results.



Conclusion

Obviously, this simulation tool cannot be used to effectively model a real fire; this is due to the vast number of necessary parameters (regional humidity variability, wind speed, precipitation, vegetation types, etc) that are difficult to implement, as well as the computational complexity of running such a simulation, and that's while ignoring the problem of data quality and the butterfly effect rendering any simulation result worthless beyond a certain depth of analysis. Additionally, scaling the grid to more realistic sizes would be somewhat limited due to the computational cost of things like terrain generation scaling quadratically (O(n^2)). Lastly, the project consistently applies the principles of Object Oriented Programming to improve code clarity and modularity, as was highlighted throughout the course lectures.