Description:

This simulation will display a swarm of drones fighting a wildfire on a 2-D grid. The program will include 3 function files, all working together to accurately simulate extinguishing fires. One will show the intensity, duration, and size of the fires present in the environment. The second file will encase a struct array of drone agents that will locate and extinguish the fires, using the data provided by the fire function. The last file will be a driver script that runs the entire simulation loop, updates, and manages display or output.

The following data structures will be implemented:

params: overall simulation settings, time step, decay rate, grid size, etc.

fire: fire intensity, size, and duration grid

drone(i): represents an individual drone with position, target, and logs

The following functions will be implemented:

Fire = FireIntensity (fire, params)
Inputs:

- fire: fire intensity, size, and duration grid
- params: time step, decay rate, grid size, etc.

Outputs:

- Fire: an updated fire grid

Drone = UpdateDrone (drone, fire, params)
Inputs:

- drone: structure encasing each drone's position, target, water, speed to target
- fire: fire intensity, size, and duration grid
- params: time step, decay rate, grid size, etc.

Output:

- UpdateDrone: an updated status of each drone, a description of nth drone's new position based on previous target (fire)

RunSimulation:

Inputs:

- drones: structure encasing every drone's position, target, water, speed to target
- fire: fire intensity, size, and duration grid

- params: time step, decay rate, grid size, etc.

Outputs:
- RunSimulation: Runs the time-stepping loop

Fire Dynamics Model:

This simulation will use a 2D cellular automaton to efficiently simulate how fire spreads across a landscape.

Grid:
- The environment is represented by a 2D grid with cells representing a portion of the land

Cells:
- Each cell has a fire intensity value of 0 or 1
- 0 represents no fire (unaffected or extinguished)
- 1 represents burning

Update Rules:
- Each cell can affect neighboring cells (N, S, E, W) through increasing their intensity slightly
- Each cell's fire decreases by a small constant over time
- When water is dropped onto cell, the cell's intensity resets to 0
- Intensities are kept within [0,1] to ensure stability

Testing Plans:

- A single burning cell spreading symmetrically
- One drone extinguishing a 1x1 fire in different spots of the grid
- Making sure neighboring intensities are being affected by a cell, making sure the threshold value comparison is held true for all cells with fire

Fire grid and drone position
- Set grid size to (x,y)
- Start fire at (x,y)
- Place a drone (x,y)
- Run Simulation(drone, fire, parms.

Should show a fire hotspot at (x,y) and a drone moving towards position.

Fire spread and decay
- Fire should decrease/spread depending on our parms

Drone reaches fire.
- Drone(i) at (x,y)

- Fire at (x,y)
- Small grid(x,y)

Drones reaches target, intensity should decrease at grid position, drone assigns new state.

Multiple drones
- Randi num drones with different positions
- Fire states at two spost
- Each drone moves to fire
- Fire decreases faster
- Simulation end before max steps

Fire Extinguishes
- Small fire
- N drones
- Run n steps
- Simulation ends quickly
- “Clear zone”
- Fire grid = zero

Pseudocode:

Function Fire = FireIntensity (fire, params)

Inputs:

fire : 2D grid of fire intensity values, including size and duration

Params: structure that contains the following:
TimeStep: 0.1
MaxSteps: 1
DecayRate: 0.05 (rate at which fire intensity decays)
SpreadRate: 0.01 (rate at which fire spreads to neighboring cells)
GridSize: [30, 30]
NumDrones: 5

Outputs:
Fire: updated fire grid

Fire = copy(fire)
- Making sure not to overwrite original data when updating fire grid
- This new grid will store update fire data without changing the original data

For every cell located at row i and column j in the fire grid:
- Use old grid to get the present intensity

- Reduce the intensity based on decay rate and time step: NewIntensity = OldIntensity * exp(-DecayRate* TimeStep)
- (if loop) If new intensity is smaller than the threshold value, the intensity of this cell is now set to zero; it is considered extinguished
- If larger than threshold (else) the new intensity value will be in the updated grid (not extinguished)

For remaining cells that are still on fire:

- Neighboring cells—N, S, E, W—will be checked and some fire intensity will be spread based on spread rate and time step
- Skip cells outside grid

Make sure fire intensities are kept between [0,1] to ensure stability

Return updated fire grid

End

Drone Data Structure Antonio will do
params struct: {grid_size, time_step, spread_rate, decay_rate, num_drones, max_steps}
GridSIze = [30,30]
TimeStep= .1
MaxSteps = 1
SpreadRate=.01
DecayRate=0.05
NumDrones= 5

DroneStructure = {DroneID, DronePosition, DroneTarget,DroneState,DroneWater}
DroneID = [1:5]
DronePosition = [10,20]
DrobeTarget = [30,30]
DroneState = [searching, moving, extinguishing]
DroneWater = 30

Driver Script

Function RunSimulation (drones, fire, params)

Inputs:

fire : 2D grid of fire intensity values, including size and duration

Params:
Structure that contains the following:

- TimeStep: 0.1
- MaxSteps: 1
- DecayRate: 0.05 (rate at which fire intensity decays)
- SpreadRate: 0.01 (rate at which fire spreads to neighboring cells)
- GridSize: [20, 30]
- NumDrones: 5

drones:
A collection of drones containing:

- DroneID = [1:5]
- DronePosition = [10,20]
- DroneTarget = [30,30]
- DroneState = [searching, moving, extinguishing]
- DroneWater = 30

Output:
RunSimulation: runs the time-stepping loop to update

Initialize simulation state

- Set time to zero
- Prepare data structures (fire, params, drones) for output
- Display all initial data states, including fire grid and drone location

Begin loop:

- While time is less than the total simulation
- Update fire environment by calling the FireIntensity function
- Fire grid is now updated

- Use updated grid for each drone in drones
- Select target position based on proximity
- Calculate step towards target with max velocity of 25mi/hr
- When drone reaches target, use water supply to reduce fire intensity
- Update fire grid again to show new statuses of fire
- Assign new targets or mark as idle

- Update global state and time by replacing old fire grid with an updated one after the drone actions
- Simulate by time step (0.1)

- End the simulation

- Stop time loop when all fires are extinguished
- Log final statues of grid and drones

Division of Labor:

The project will be divided among two team members:

- Irene will handle fire and environment updates
- Antonio will handle drone logic, ie drone position, navigation, and updates.
- Visualization and integration into the driver script will be a collaborative effort.