Tala Nemeh

101170694

Department of Systems and Computer Engineering

Course: SYSC 4906: Methodologies for Discrete Event Modelling and Simulation

Assignment 2

Fireflies

Introduction:

The objective of this assignment was to simulate the phenomenon of fireflies flashing in unison. In this assignment, a cellular model was created using the Cell-DEVS methodology to simulate this behavior.

Conceptual Model:

Each cell in the grid represents a single firefly. The cells follow a cycle of states between 0-9, and synchronization occurs from local interactions.

Cell states:

- Each firefly is modeled as a cell with a cycle of 10 discrete states (0 through 9).
- State 0: Represents the flash (on) state.
- State 1: The pre-flash state that always transitions into a flash (state 0).
- States 2–9: Represent the dark or off phases, during which the cell naturally decrements its state by 1 in each cycle.

Behaviour: If any adjacent neighbor is flashing (state 0), a cell in states 2–9 resets its countdown to state 9 rather than continuing its natural countdown. This serves as a nudge that eventually synchronizes all cells. Once a few cells begin flashing in unison, a cascading effect occurs through the entire grid, leading to global synchronization.

Neighbourhood: Each cell can perceive its immediate neighbour only in a Moore type neighbourhood. Furthermore, this implementation of the fireflies model is represented in a 2D 20x20 grid that is wrapped around the edges.

Assumptions:

- Fireflies interact only with cells defined in their neighborhood.
- Fireflies are stationary.
- All cells follow the same rules.
- The system is wrapped.

Cell-Devs Fromal Specification:

- Atomic Cell Model for an individual firefly cell:
  - CD = <X, Y, I, S, θ, N, d, δint, δext, τ, λ, D>
    Where:
    X = Y = ∅
    S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
    0: ON state (flashing)
    1: Pre-flashing state
    2-9: Post flash state (dark)
    N =
    [-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]

    delay = transport delay
    d = 1 s
    δ_int:
    if s = 0 -> δ_int (s) = 9
    if s = 1 -> δ_int(s) = 0
    if s = {2, …,9} -> δ_int(s) = 0 if at least one neighbouring cell is in state 0. Otherwise δ_int(s) = s -1
    δ_ext = none. No external inputs in this simulation.
    τ = N -> S, as defined in δ_int
    λ = s
    D = 1 s

- Coupled Cell model (The Habitat):
  - CCA = < S, n, C, η, N, T, τ >

    Where:

    S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
    n = 3
    C={C (i,j,k) | i,j,k∈{1,…,10}}
    η = 1
    N = { (dx, dy, dz) | dx, dy, dz ∈ {–1, 0, 1} }
    Border = wrapped

Implementation:

The C++ implementation models each firefly as a cell with the following rules:

```
if (current_state == 0) then
    next_state = 9
else if (current_state == 9) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 8
else if (current_state == 8) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 7
else if (current_state == 7) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 6
else if (current_state == 6) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 5
else if (current_state == 5) then
    if (statecount(0) > 0) then
```

```
        next_state = 9
    else
        next_state = 4
else if (current_state == 4) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 3
else if (current_state == 3) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 2
else if (current_state == 2) then
    if (statecount(0) > 0) then
        next_state = 9
    else
        next_state = 1
else if (current_state == 1) then
    next_state = 0
```

This pseudocode implements the core synchronization mechanism where the natural countdown cycle is interrupted by a neighbor flashing.

JSON Configuration:

The simulation is configured on a 2D wrapped grid. Initially, the immediate Moore neighborhood (range 1) was used:

```
"neighborhood": [
```

{ "type": "relative", "neighbors": [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]] }

]

When this configuration sulted in a patterned behavior and failed to achieve full synchronization, the neighborhood was extended. The configuration became:

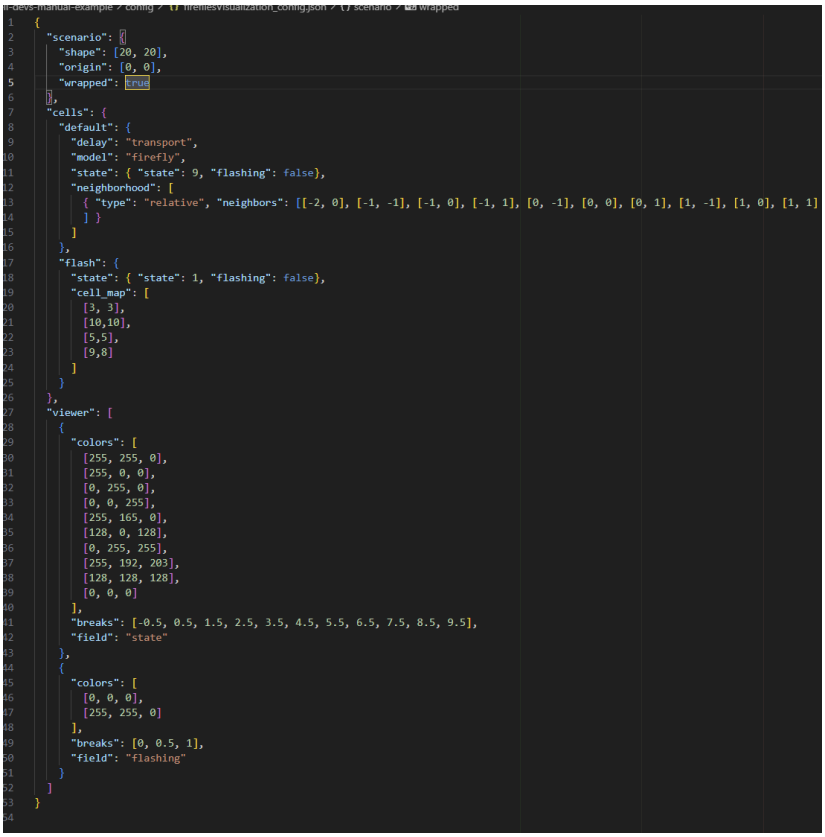
"neighborhood": [

{ "type": "relative", "neighbors": [[-2,0], [-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]] }

]

This extension allows cells to detect flashes from farther away (at offset (-2,0)), which helps synchronize the entire grid.

Sample configuration:



Experiments and Results:

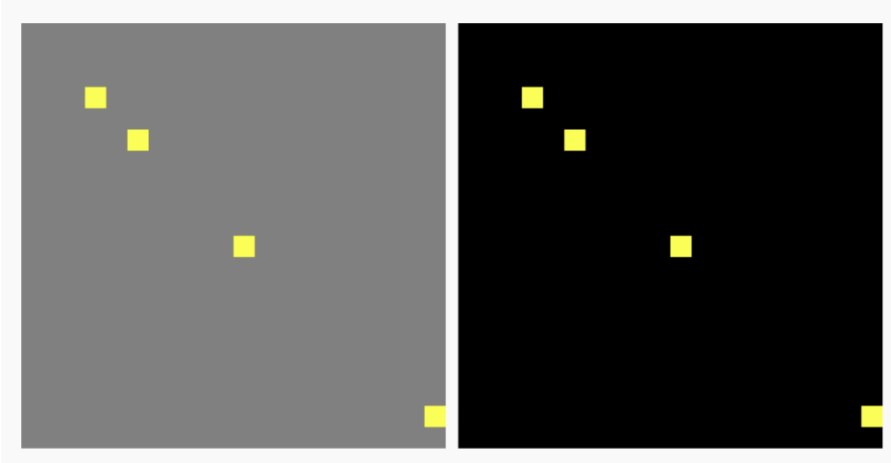
In these experiments, the grid (20×20) was initialized mostly with cells in state 8 (off) and a few cells with a flashing state (state 0) as a trigger.

For the screenshot presented below, the left side represents each cell color coded according to its currents state (0-9). For example, a cell in the ON state (state 0) is represented by yellow. The right side simply indicates whether the cell is flashing or not (yellow for flashing, black for not).

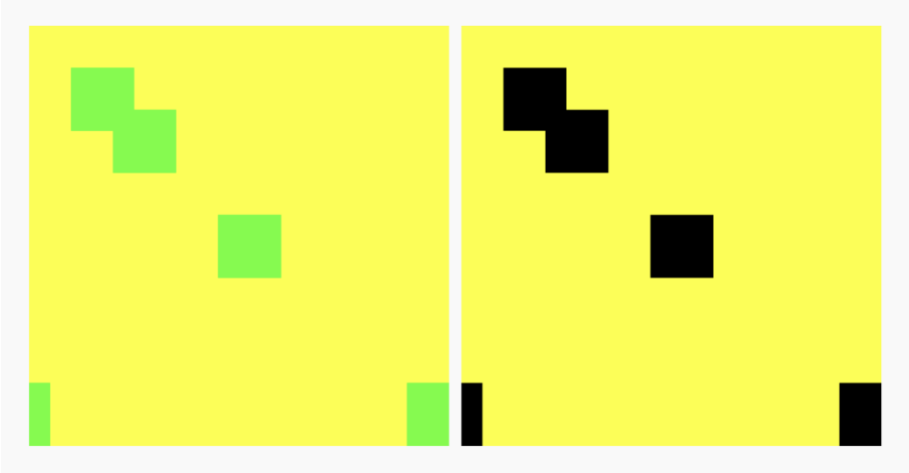
- **Initial experiments:** Initially, when only the immediate neighbors (Moore range 1) were used, the simulation generated a patterned behavior rather than a full synchronization between the cells.
  The following neighborhood is used for this experiment:
  "neighborhood": [
  { "type": "relative",
  "neighbors": [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]] }]

  Although some cells would synchronize, a few cells stayed out of phase. The output of this experiment can be seen below:

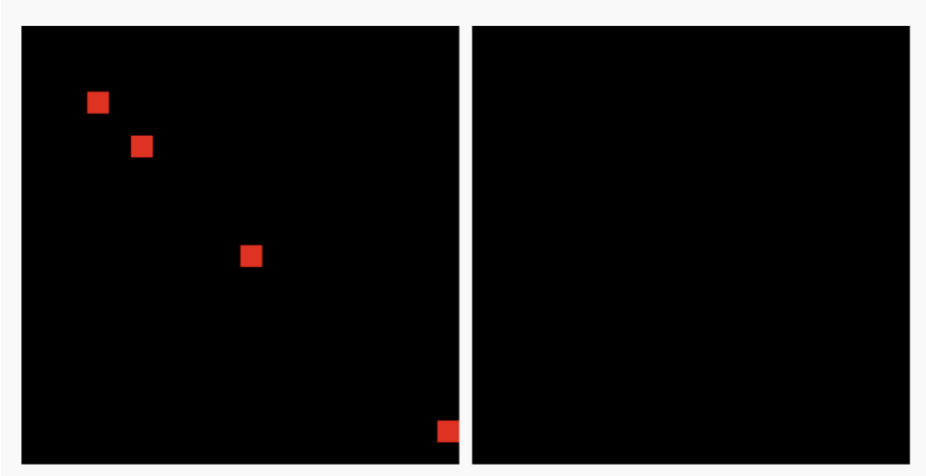
  Output at step 0:



  Output at step 8:



Output at step 9:

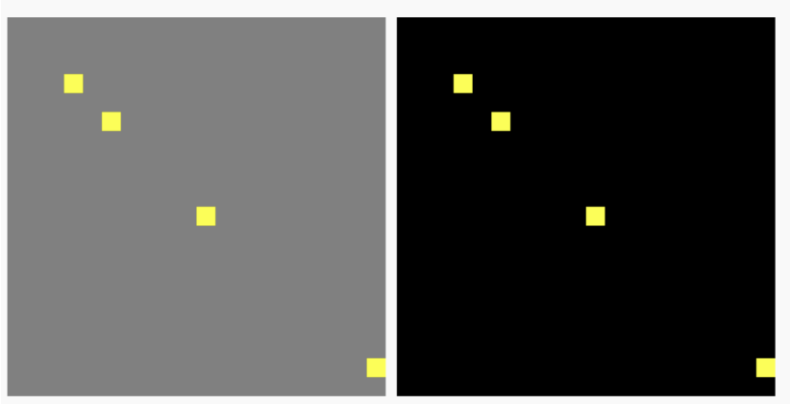


While some cells are able to detect and sync with their flashing neighbours, other cells are left out of sync. This pattern repeats forever in this scenario.

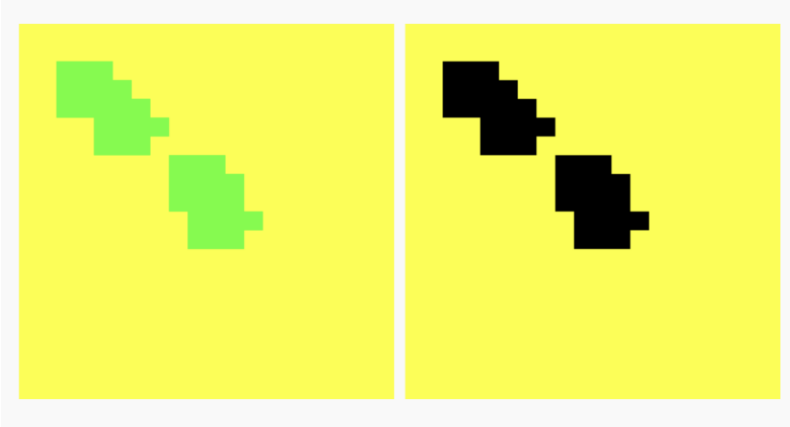
- **Extended neighborhood experiment:** To address the issue encountered in the previous scenario, the neighborhood was extended to include one extra neighbor at offset (-2,0). In this experiment, the simulation successfully shows synchronization of all cells eventually over several cycles. By the modification made to the neighborhood, we get results that meet the assignment's goal.

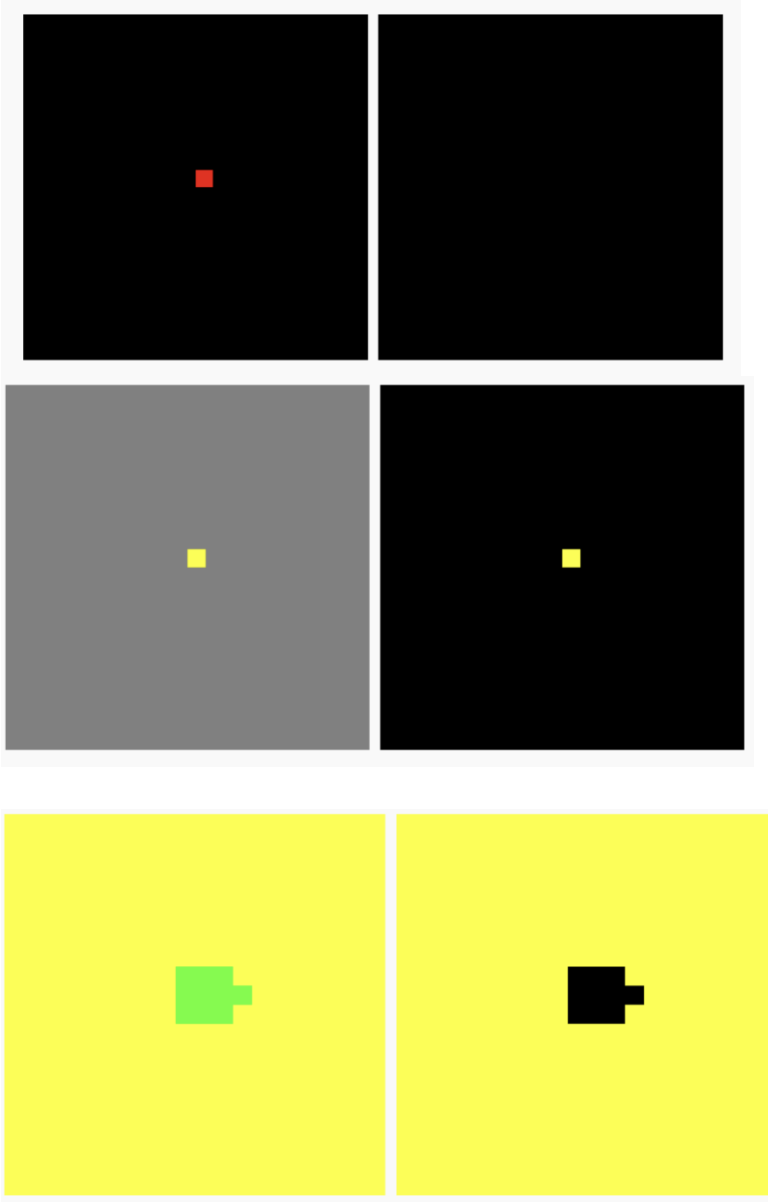
  The results of this experiment is shown below:

  Output at step 0: Initial configuration

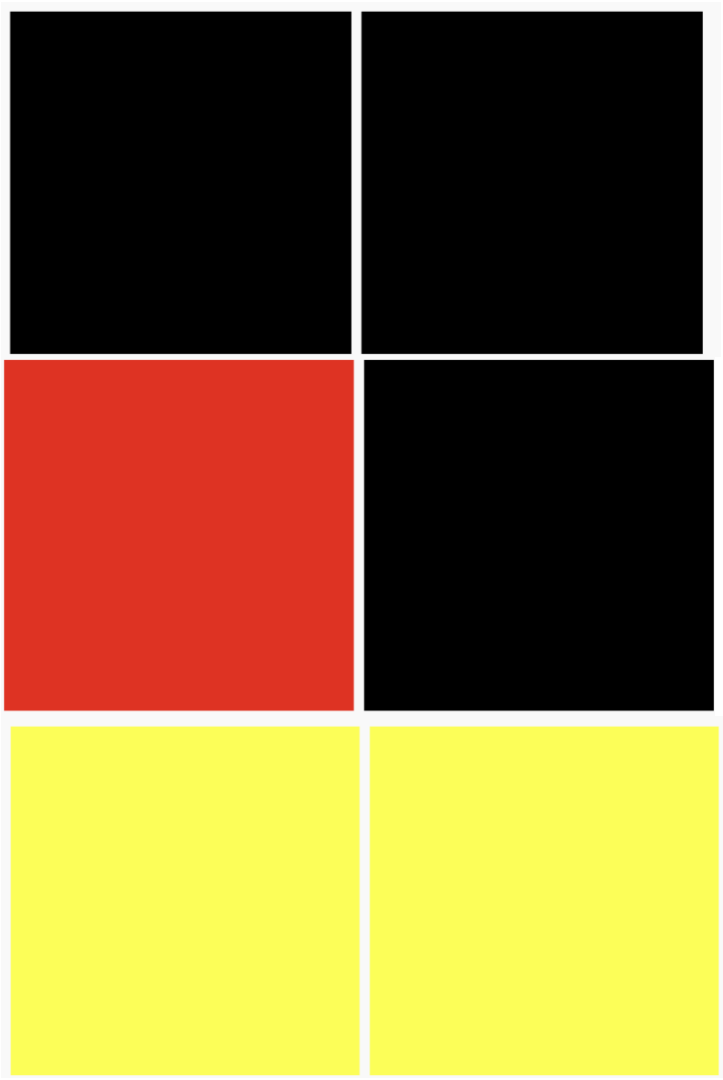


Outputs at step 8,9, 10 and 18: Transition phase showing increased synchronization





Outputs at steps 19, 27 and 28: The grid reaches and maintains full synchrony with all cells flashing on and off together



As we can see, the cells are now able to detect flashing in the neighboring cell at coordinates (-2,0). Eventually, the cells synchronize states and the behaviour of flashing fireflies is successfully simulated.

These experiments scenarios provide a clear comparison between two different neighborhood setups and explain how each is expected to influence the synchronization process.

Conclusion:

The assignment demonstrated that firefly synchronization can be modeled effectively using a cell-based simulation. Full synchrony was achieved by making two adjustments:

- Implementing a local rule that resets a cell's countdown to 9 if any neighbor is flashing.
- Extending the neighborhood so that the influence of a flash reaches farther than just immediate neighbors.

Through systematic experiments, it was shown that these adjustments ultimately result in global synchronization.