MINI-PROJECT

YASHARTH DWIVEDI

Conway's Game of Life: Exploring Cellular Automata

Conway's Game of Life

1. What is Conway's Game of Life?
2. Basic Rules of the Game
3. Visualization and Simulation

4.Advantages

5. Real-World Applications
6. Conclusion

7.Working Commands, CODE & OUTPUT

8.THANK YOU

What is Conway's Game of Life?

- Definition: Conway's Game of Life is a mathematical model and cellular automaton devised by mathematician John Conway in 1970.
- It's not a traditional "game," but rather a simulation of cell evolution based on simple rules.

Basic Rules of the Game

- Birth: A dead cell with exactly three live neighbors becomes a live cell.
- Survival: A live cell with two or three live neighbors remains alive.
- Death: All other live cells die or remain dead.

Visualization and Simulation.

- The game is played on a two-dimensional grid of cells.
- Cells can be in two states: alive (filled) or dead (empty).
- Simulation progresses in discrete time steps, with cell states evolving based on neighboring cells

ADVANTAGES

Importance of Conway's Game of Life

Emergent Behavior: Simple rules lead to complex and unpredictable patterns.

Model for Complex Systems: Serves as a model for various natural systems.

Scientific Applications: Used in biology, physics, and other fields to study pattern formation

Cellular Automata in Science

Real-world significance - Simulating population growth and decline.

Studying pattern formation in biological systems

Patterns and Structures

Blinkers, gliders, spaceships, and oscillators.

Building blocks for creating more intricate structures

Turing Completeness

Concept of Turing completeness: A system that can simulate any computation.

Remarkably, Conway's Game of Life is Turing complete, showcasing its computational power.

Creative and Artistic Exploration

Beyond science: Game of Life's creative applications.

Generating aesthetic patterns and designs

Real-World Applications

Practical uses:

- Modeling predator-prey relationships.
- Image processing for pattern recognition

Conclusion

- Recap: We've explored the significance of Conway's Game of Life.
- Ongoing relevance: Continues to inspire research, creativity, and computational exploration.

WORKING COMMANDS

COMMANDS :-

1. **Spacebar**: While the simulation is running, press the spacebar to pause it. Press it again to resume the simulation.
2. **'c'**: While the simulation is running, press the 'c' key to clear the grid (set all cells to dead).
3. **'r'**: While the simulation is running, press the 'r' key to randomize the grid (set cells to either alive or dead randomly).
4. **Mouse Click**: While the simulation is running, click on a cell in the grid to toggle its state between alive and dead. This allows you to interactively modify the grid.

CODE

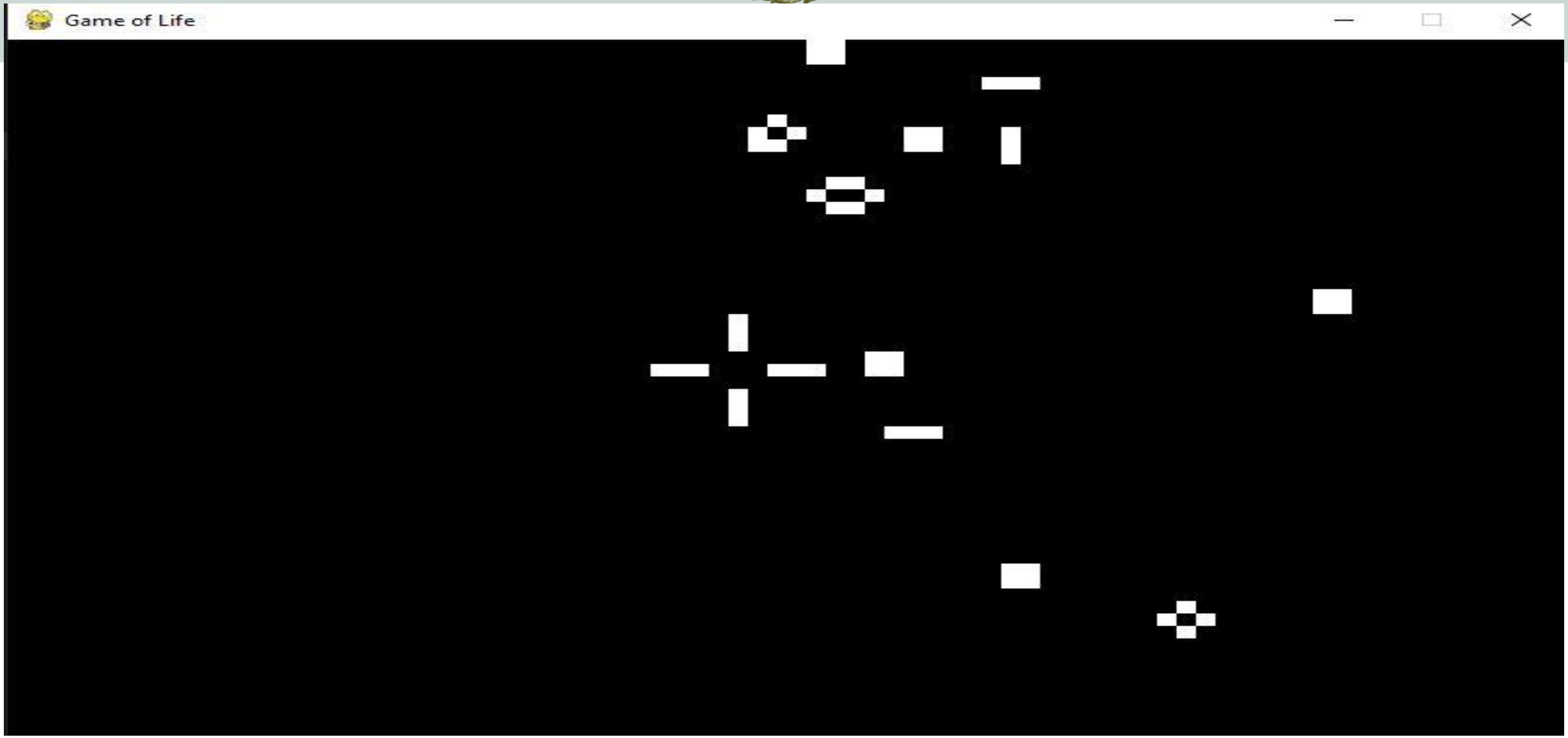
```
import pygame
import numpy as np

# Initialize pygame
pygame.init()

# Set up display
width, height = 800, 600
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Game of Life")

# Set cell size and grid dimensions
cell_size = 10
cols, rows = width // cell_size, height // cell_size

# Create grid and initialize variables
grid = np.zeros((rows, cols), dtype=int)
running = False
clear_grid = False
```

```python
# Define colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

# Main loop
clock = pygame.time.Clock()
while True:
    screen.fill(BLACK)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        elif event.type == pygame.MOUSEBUTTONDOWN and not running:
            if event.button == 1:  # Left mouse button
                row = event.pos[1] // cell_size
                col = event.pos[0] // cell_size
                grid[row, col] = 1 - grid[row, col]
        elif event.type == pygame.MOUSEMOTION and pygame.mouse.get_pressed()[0] and not running:
```

```python
            elif event.type == pygame.MOUSEMOTION and pygame.mouse.get_pressed()[0] and not running:
                row = event.pos[1] // cell_size
                col = event.pos[0] // cell_size
                grid[row, col] = 1

            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    running = not running
                elif event.key == pygame.K_c:
                    grid = np.zeros((rows, cols), dtype=int)
                elif event.key == pygame.K_UP:
                    clock.tick(30)  # Increase the simulation speed
                elif event.key == pygame.K_DOWN:
                    clock.tick(5)   # Decrease the simulation speed

        if running:
            new_grid = np.copy(grid)

            for row in range(rows):
```

```
        for row in range(rows):
            for col in range(cols):
                x = col * cell_size
                y = row * cell_size

                # Draw cells
                if grid[row, col] == 1:
                    pygame.draw.rect(screen, WHITE, (x, y, cell_size, cell_size))

                # Count live neighbors
                neighbors = np.sum(grid[max(0, row - 1):min(rows, row + 2),
                                        max(0, col - 1):min(cols, col + 2)]) - grid[row, col]

                # Apply Game of Life rules
                if grid[row, col] == 1 and (neighbors < 2 or neighbors > 3):
                    new_grid[row, col] = 0
                elif grid[row, col] == 0 and neighbors == 3:
                    new_grid[row, col] = 1

        grid = new_grid

    pygame.display.flip()

    clock.tick(100)  # Adjust the speed of the simulation here
```

OUTPUT



Thank you

YASHARTH DWIVEDI

yadw22csds@cmrit.ac.in

USN :- 1CR22CD061

Sec/Roll.no :- L-58