CS231

Project 2

Yazan Bawaqna

Report: Simulating the Game of Life

Abstract:

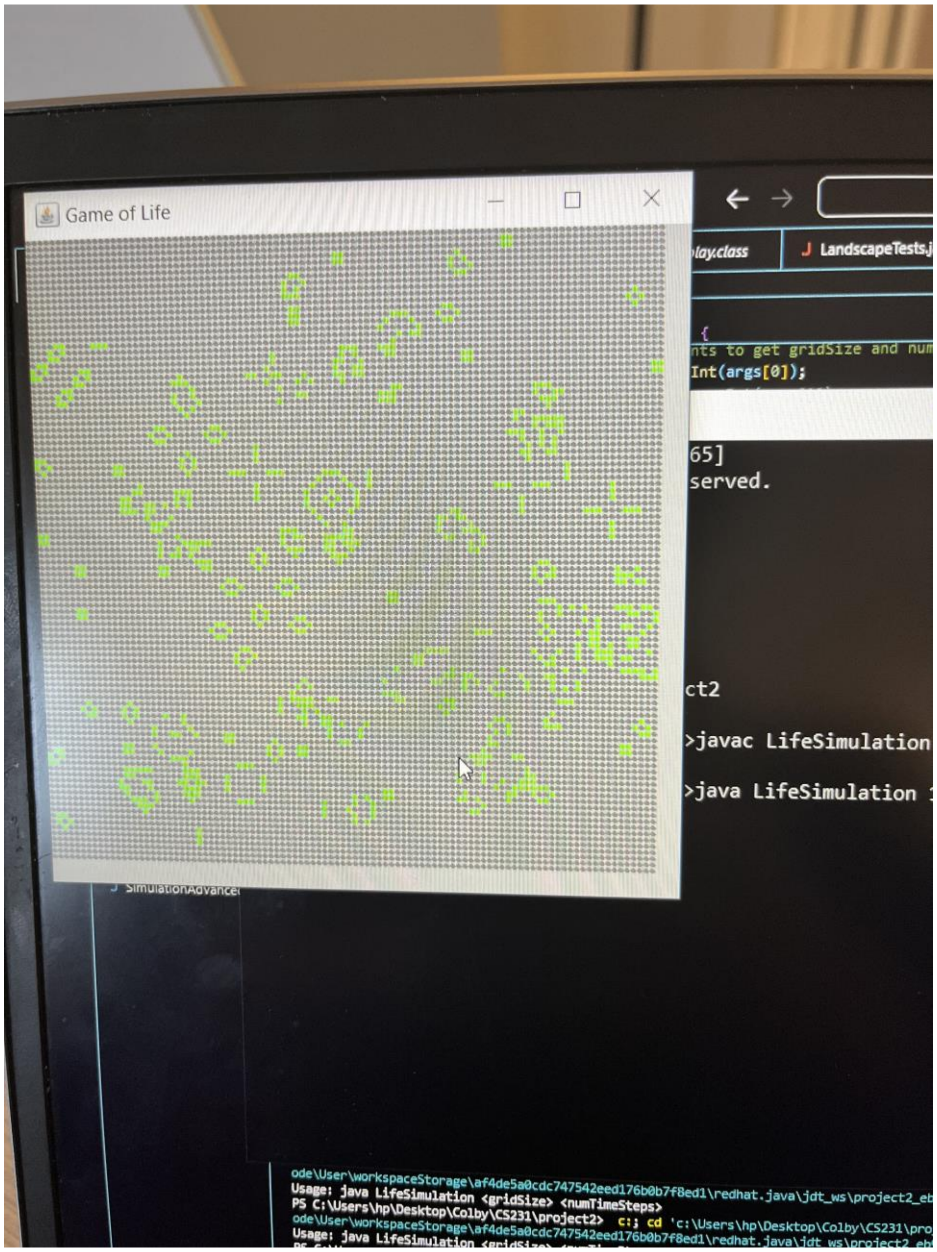
This project delves into the realm of multidimensional ArrayLists within the framework of Conway's Game of Life, a classic cellular automaton simulation. Conway's Game of Life, a renowned example of cellular automata, explores the dynamics of entities on a 2D grid, where each cell's state depends on the states of its neighbors. Our goal was to develop a Java program for this simulation, emphasizing the data structure of multidimensional ArrayLists to manage the grid and its evolving state.

For new students, in computer science, multidimensional ArrayLists represent dynamic, nested data structures capable of storing and manipulating collections of objects in multiple dimensions. The way they work is similar to how 1D Arraylists work, with the difference being the objects inside the Arraylists are Arraylists themselves. This trick allows us to store data in a grid, with each data entity having a row (outer Arraylist) index and a column (inner Arraylist) index.

While arrays and one-dimensional ArrayLists serve as fundamental data structures, multidimensional ArrayLists empower us to navigate the intricate landscape of the 2D grid, providing a dynamic framework for updating cell states and modeling complex cellular automaton behavior. In this project, cells are represented via one data entity in the grid or in other words an index of an inner array.

Results

By running the simulation for a specified number of time steps and adjusting initial conditions, users gain insights into how simple rules can lead to complex, evolving patterns in the grid. Users can experiment with various grid sizes through the command-line interface and observe the simulation's behavior.

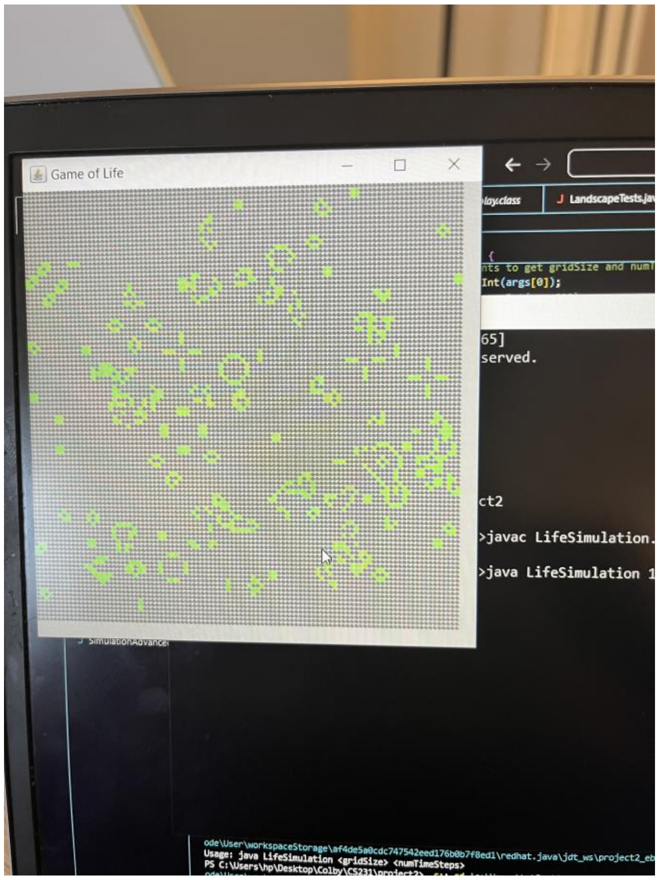


```
nts to get gridSize and num
Int(args[0]);
```

```
65]
served.

ct2

>javac LifeSimulation

>java LifeSimulation
```

```
ode\User\workspaceStorage\af4de5a0cdc747542eed176b0b7f8ed1\redhat.java\jdt_ws\project2_eb
Usage: java LifeSimulation <gridSize> <numTimeSteps>
PS C:\Users\hp\Desktop\Colby\CS231\project2>  c:; cd 'c:\Users\hp\Desktop\Colby\CS231\pro
ode\User\workspaceStorage\af4de5a0cdc747542eed176b0b7f8ed1\redhat.java\jdt_ws\project2_eb
Usage: java LifeSimulation <gridSize>
```

Image-1: The first frame of the simulation after launching LifeSimulation class



```
 {
nts to get gridSize and num
Int(args[0]);
```

```
65]
served.

ct2

>javac LifeSimulation.

>java LifeSimulation 1
```

```
ode\User\workspaceStorage\af4de5a0cdc747542eed176b0b7f8ed1\redhat.java\jdt_ws\project2_eb
Usage: java LifeSimulation <gridSize> <numTimeSteps>
PS C:\Users\hp\Desktop\Colby\CS231\project2>
```

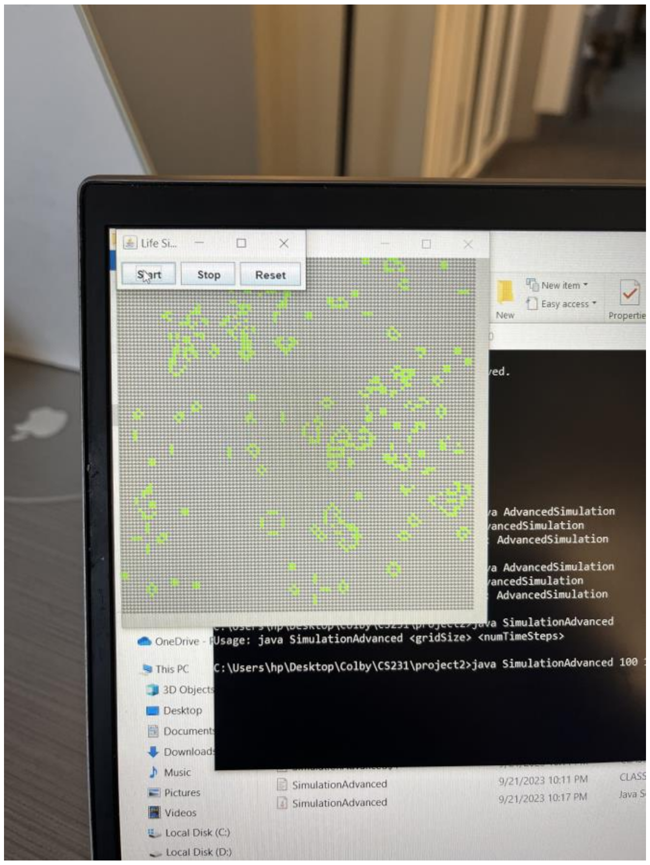
Image-2: The second frame of the simulation after launching LifeSimulation class

In examining consecutive frames of the Conway's Game of Life simulation, one can observe a dynamic interplay of cell births, survivals, and deaths. New live cells appear in some frames, driven by the rule that a dead cell with precisely three live neighbors springs to life in the next generation. Live cells, guided by the principle of having two or three live neighbors, persist from one frame to the next. Conversely, cells facing underpopulation or overcrowding meet their demise. The patterns themselves exhibit fascinating behavior; some remain static ("still lifes"), while others oscillate in place. Additionally, gliders traverse the grid in diagonal motion, injecting a sense of motion and complexity into the evolving patterns. The specific outcomes depend on the initial cell configuration, resulting in visually captivating sequences of life and death within the simulation.

Extension

In our project extension, we introduced a graphical user interface (GUI) to enhance the user experience of our LifeSimulation program. This GUI consists of several key elements: a visual representation of the landscape grid, a control panel with buttons, and a JFrame to house the entire interface. The primary purpose of this GUI is to provide users with a more interactive and intuitive means of controlling and observing Conway's Game of Life simulation.

The control panel offers three essential buttons: "Start," "Stop," and "Reset." The "Start" button triggers the simulation, allowing it to progress through multiple time steps. Conversely, the "Stop" button halts the simulation's progression, providing users with the flexibility to pause and observe the landscape. The "Reset" button serves as a handy tool to return the simulation to its initial state. This extension greatly enhances the usability of our project by allowing users to interact with and control the simulation, enabling them to experiment with different initial conditions and observe the dynamic evolution of the grid. Overall, the addition of this GUI element significantly improves the accessibility and engagement of our LifeSimulation project, making it a more user-friendly and enjoyable experience. Attached to the file submitted, is a video demonstrating the mechanism of how does the extension class work, the video name is “Extension”



```
ved.

va AdvancedSimulation
vancedSimulation
  AdvancedSimulation

va AdvancedSimulation
vancedSimulation
  AdvancedSimulation

va SimulationAdvanced
Usage: java SimulationAdvanced <gridSize> <numTimeSteps>

C:\Users\hp\Desktop\Colby\CS231\project2>java SimulationAdvanced 100
```

Image-3: Running SimulationAdvanced class, which is the extension class, and testing the buttons.

Acknowledgements

- Dinesh Varyani (Youtube channel), gave me the trick of using the getNeighbors method
- ChatGPT, helped me build GUI buttons for my extension, debugged my advance and my reset functions
- Bro Code (youtube channel) gave me the structure of “Catch and try” as well as Jframe imports.
- Michael Tenkorangir, Minor fixation for my testing method.