Display and Control of Conway’s Game of Life

Abstract

In this project, we explored Conway’s Game of Life, a classic example of a cellular automaton, where entities on a 2D grid evolve based on local interaction rules. We implemented this simulation using Java, creating Cell, Landscape, and LifeSimulation classes, and visualized the evolution with a Swing-based GUI. The project enabled us to apply core computer science concepts, including object-oriented programming, data structures (such as 2D arrays and lists), and event-driven programming through timers and GUI controls. My results show that the initial probability of alive cells strongly affects long-term patterns, with moderate probabilities stabilizing at higher living cell counts and very low or high probabilities often leading to die-out, demonstrating how simple rules create complex emergent behavior.

Results

- **Experiment**

To explore the long-term behavior of Conway’s Game of Life, simulations were run on a 100×100 grid with varying initial alive-cell probabilities, ranging from 0.1 to 0.9 in increments of 0.1. Each simulation was executed for 1000, 1500, and 2000 time steps. The primary metric recorded was the average number of living cells at the end of each simulation. This metric captures how initial population density affects the survival, stabilization, and evolution of the cellular system over time. Each configuration used random initialization to account for stochastic variation in outcomes.

- **Hypothesis**

I hypothesized that moderate initial probabilities (around 0.4–0.6) would produce the most surviving cells in the long term due to a balance between underpopulation and overcrowding. Low probabilities (0.1–0.3) were expected to have fewer surviving cells due to insufficient neighbors for reproduction. High probabilities (0.7–0.9) were expected to experience a rapid die-off due to overcrowding. By examining multiple time steps, the experiment investigates not only the initial impact of density but also the stability and evolution of patterns over time.

- **Table of results**

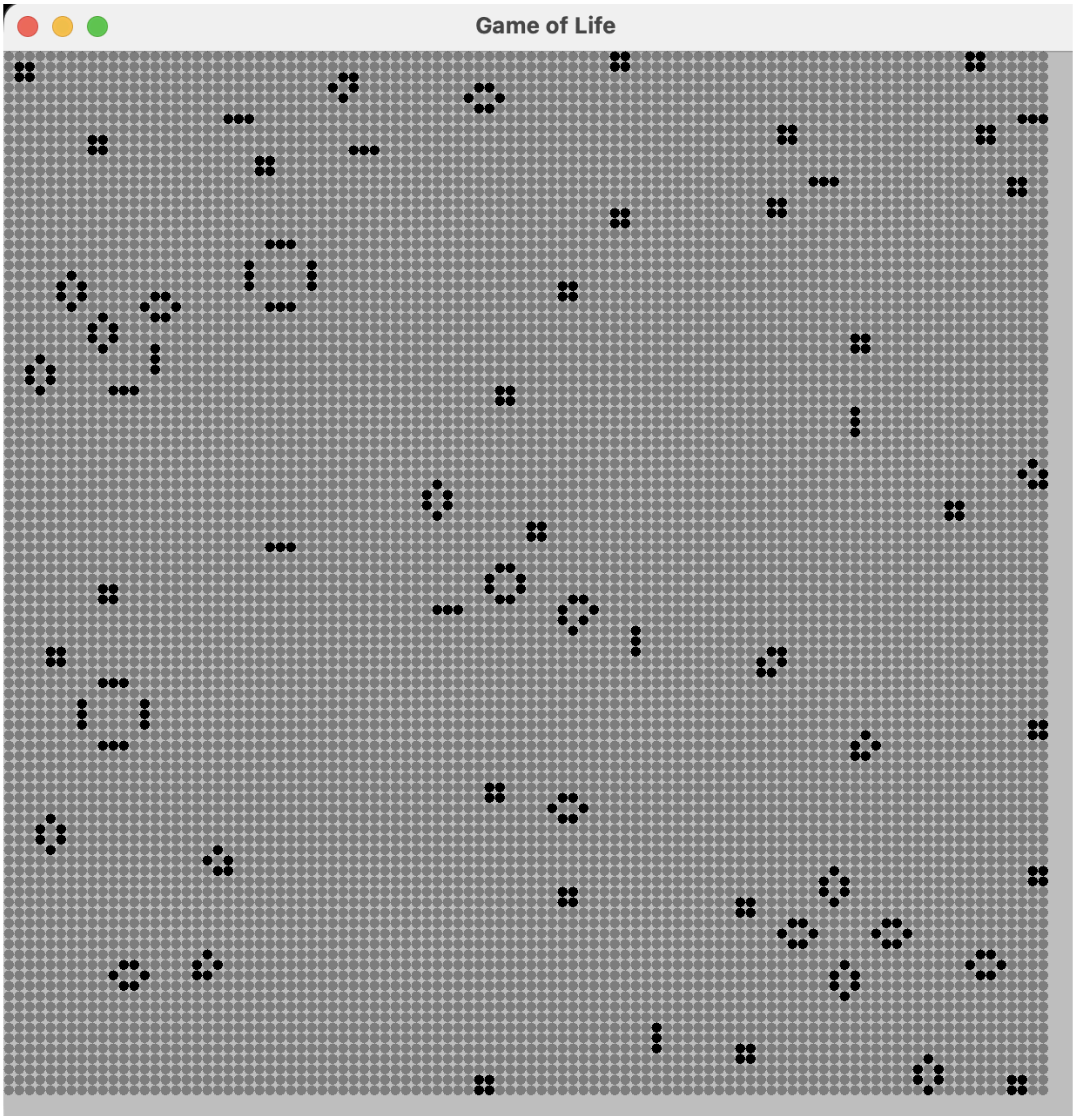
Table 1. Average Living Cells vs. Initial Probability at Different Steps.

Initial alive probability	Average living cells after 1000 steps	Average living cells after 1500 steps	Average living cells after 2000 steps
0.1	212.4	248.8	262.8
0.2	321.4	308.0	322.6
0.3	321.2	333.2	342.0
0.4	348.8	329.4	320.6
0.5	357.4	359.4	320.4

0.6	308.2	328.4	281.8
0.7	312.2	277.0	229.6
0.8	31.2	28.4	64.0
0.9	1.6	1.6	2.4

Average number of living cells on a 100×100 grid for varying initial probabilities at 1000, 1500, and 2000 steps. Moderate initial probabilities (0.4–0.6) generally maintain the highest population, while extremely low and high probabilities result in lower survival. Over time, certain densities show population oscillations, illustrating the dynamic evolution of the cellular system. I created a TestProbability class, and the output data text file is included in the zip folder.

Figure 1: Living Cells at 1000 Steps with Randomized Initial Probability.



The initial alive-cell probability is randomly chosen between 0% and 100% for this run, so the plot represents a single example rather than an average across probabilities.

- **Synthesis**

From Table 1, we found that initial probabilities between 0.3 and 0.6 consistently produce the largest surviving populations across all time steps, confirming the hypothesis. These densities allow the formation of stable structures such as blocks, blinkers, and oscillators, which persist over time. Meanwhile, low probabilities increase slowly. Probabilities of 0.1–0.2 show gradual growth over time (e.g., 0.1 goes from 212.4 → 262.8), reflecting slow colonization due to underpopulation constraints. In contrast, high probabilities cause die-off. Probabilities above 0.7 experience drastic reductions (e.g., 0.8 drops to 28.4 at 1500 steps) due to overcrowding, validating the predicted negative effect of high initial density. We can also notice some densities exhibit non-monotonic behavior over time (e.g., 0.8 goes from 31.2 → 28.4 → 64.0), indicating temporary regrowth after die-offs, consistent with local interactions forming oscillatory structures.

These results highlight how initial population density critically shapes system dynamics in cellular automata, analogous to ecological and population systems in the real world. Underpopulation and overcrowding both lead to reduced survival, while moderate densities favor stability and long-term persistence. This study demonstrates the emergence of complex behavior from simple local rules and illustrates the balance required for stable systems, whether in artificial simulations or natural populations.

Acknowledgements

I would like to thank the course staff for providing the project template and guidance site. Thanks, Oracle, for the detailed documentation at https://docs.oracle.com/javase/tutorial/uiswing/ and https://docs.oracle.com/javase/tutorial/uiswing/painting/, as well as the W3Schools Java Tutorial.

Extensions:

WHAT:

In the original Landscape and LifeSimulation classes, the program displayed the simulation of Conway's Game of Life but lacked interactive controls to manipulate simulation parameters dynamically. Specifically, there was no way to adjust the initial density of alive cells after the program started, and the simulation would automatically run a fixed number of steps upon launch.

For my extension, I updated the LandscapeDisplay class to include:

1. Start, Stop, and Reset buttons for controlling the simulation.
2. A density slider that allows the user to adjust the percentage of alive cells before resetting the simulation.
3. Reset behavior that stops the simulation when clicked, reinitializes the landscape based on the slider's density, and prevents the simulation from continuing automatically after a reset.

I also updated the Landscape class to support initializing the grid with a specified density (initialize(double density)) while storing the chosen density as a reference. This ensures that the density slider can reliably change the grid configuration.

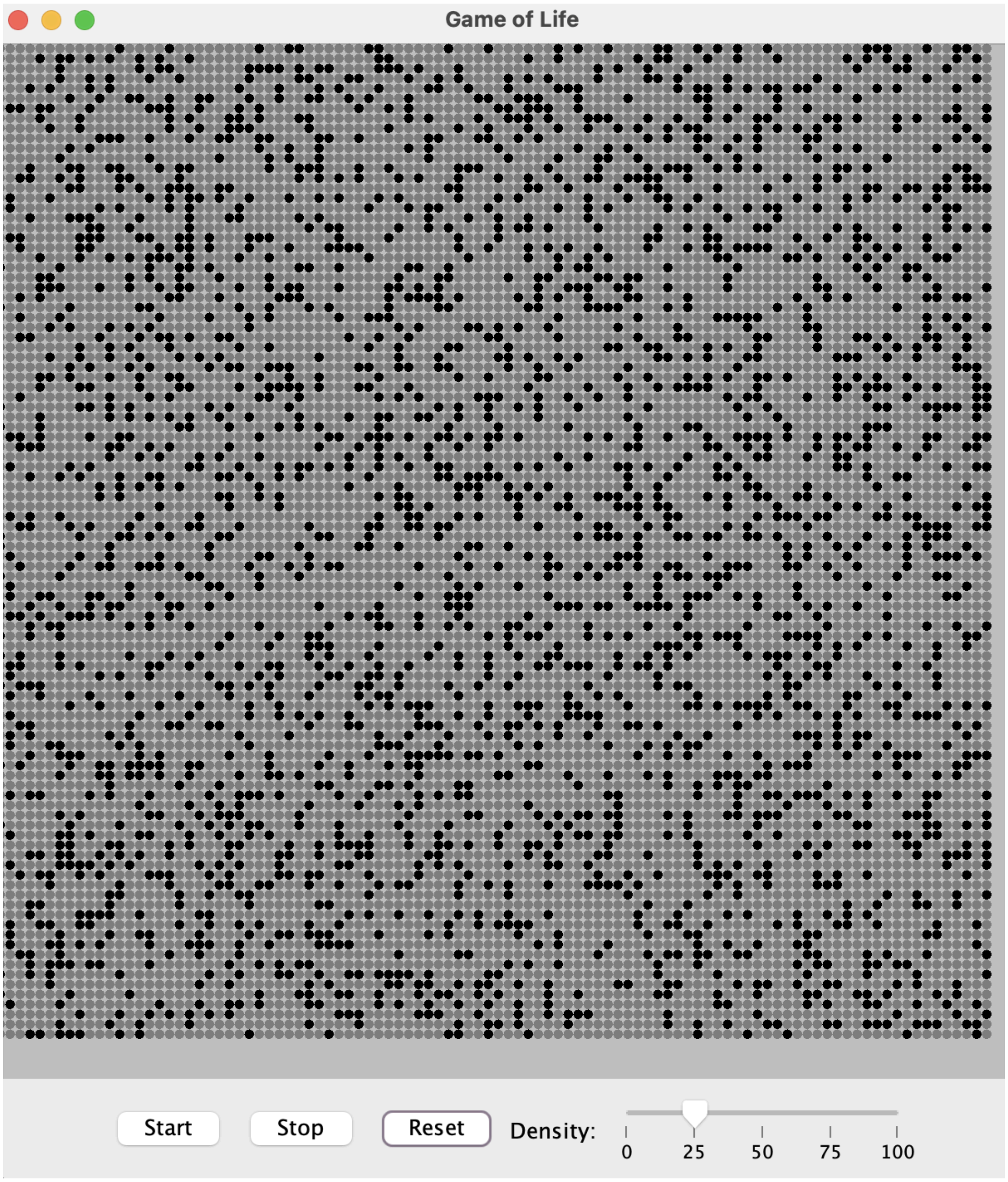
OUTCOME:

The extension resulted in several improvements to the simulation:

1. Users can interactively control the simulation, including starting, stopping, and resetting the grid at any time.
2. The density slider works dynamically. Moving the slider and clicking "Reset" immediately reconfigures the landscape with the chosen density.
3. After clicking Reset, the simulation defaults to a stopped state instead of resuming automatically. This makes it easier for the user to inspect the landscape or adjust parameters before continuing.

4. Using a grid scale of 6, the window layout was updated to ensure that all controls, including the slider, are visible without overlapping or being clipped.

   Figure 2: Demonstration of the Game of Life GUI After Applying the Extension.



The interface includes Start, Stop, and Reset buttons, along with a Density slider to control the initial proportion of alive cells.

Overall, the program became more user-friendly, allowing for exploratory experiments, such as testing how different densities affect the evolution of the Game of Life.

HOW TO RUN:

To replicate this extension, compile and run the LifeSimulation class. The simulation window will appear, displaying the grid at the specified scale of 6 pixels per cell.

- Use the Start button to begin the simulation and the Stop button to pause it.
- Adjust the Density slider to choose the desired initial probability of alive cells.
- Click Reset to reinitialize the landscape using the slider’s density. The simulation will remain stopped until you click Start again.