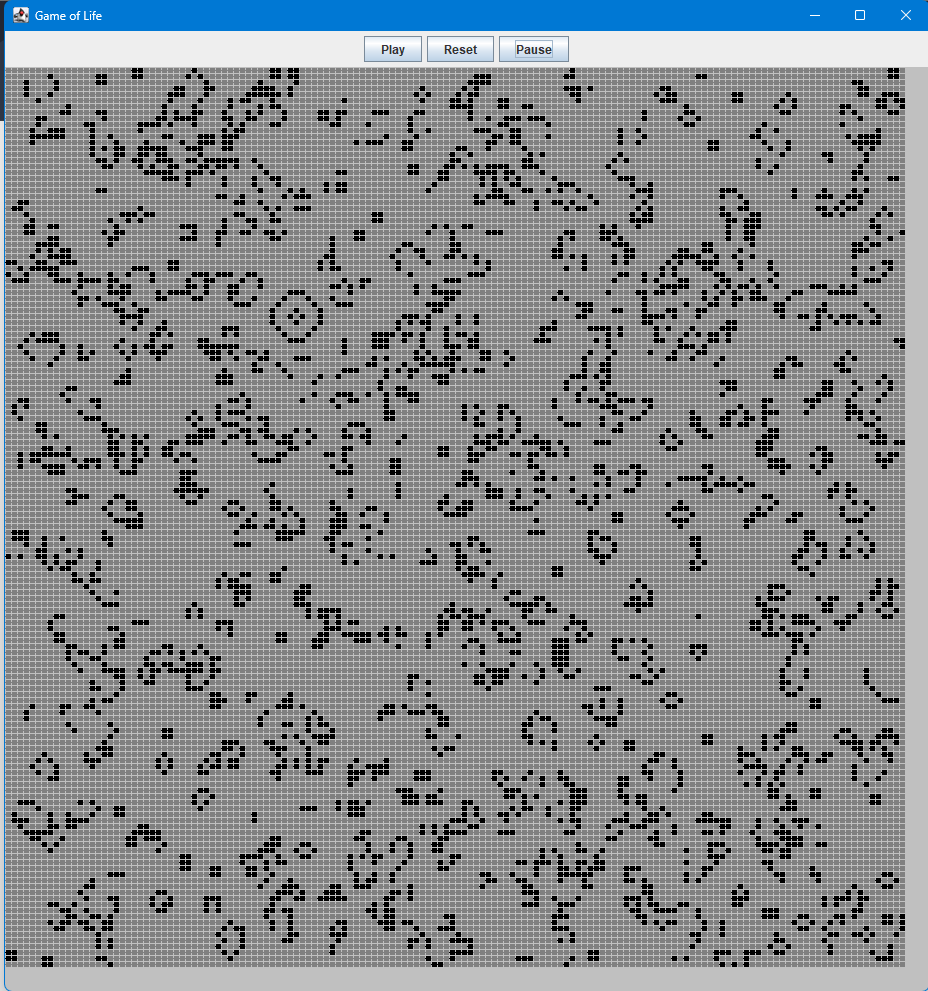
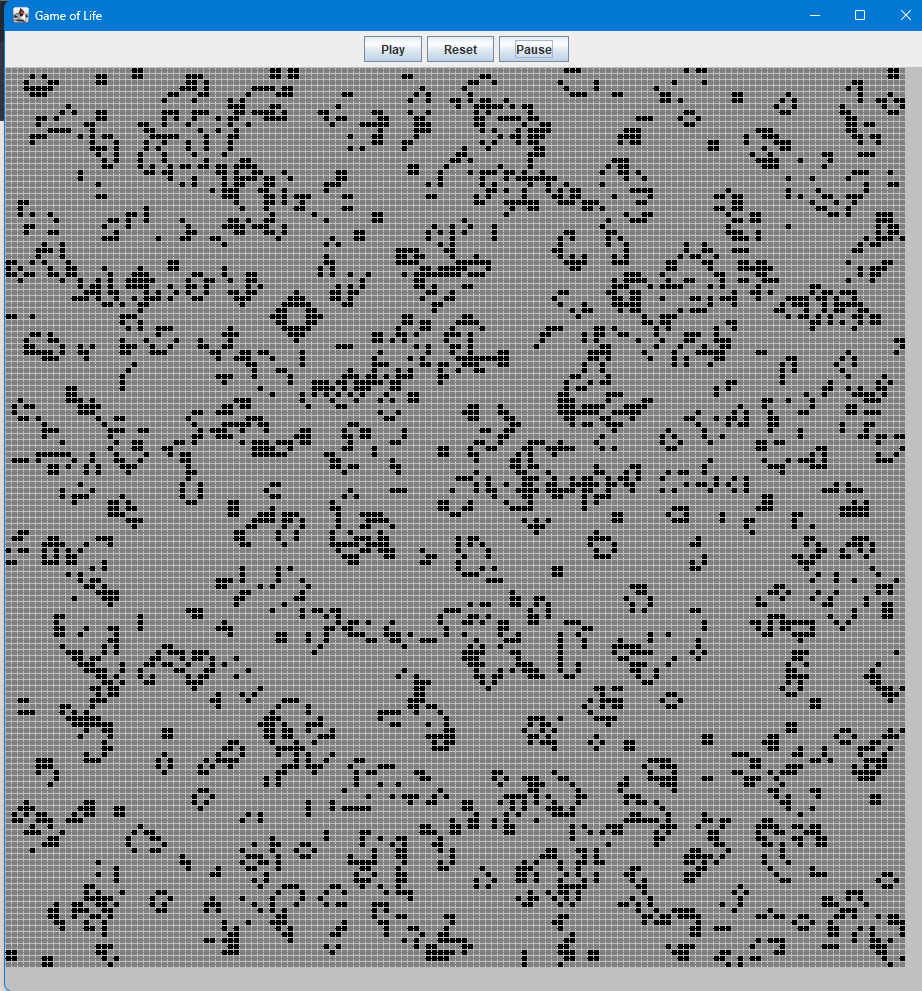
### Simulation of Conway’s Game of Life on a 2D grid

**Abstract:**

This project is about Conway’s game of Life, which is a very popular example of Cellular Automata because of its minimal rules but very complex behavior. We simulated elements of the game on a 2D grid to observe emergent behavior given a particular finite number of iterations. The core computer science concepts used were object-oriented programming for organizing cells and grids and using graphical interface using Java’s swing library. The simulation revealed how initial conditions, such as grid size and cell density, significantly influence the long-term behavior of the system. Some configurations die out quickly, while some including less than 10 cells last for hundreds of generations.

**Results:**

First, we make sure that our program runs as intended. Here, I present two sample images as a before vs after still-image to prove the authenticity.



Looking at the rules of the game, it might be correct to assume that as the size of the grid increases, so does the number of alive cells after a certain number of iterations with same probability.  
To test this, I ran 10 simulations of 1000 iterations each for 100\*100, 150\*150 and 250\*250 grid sizes.

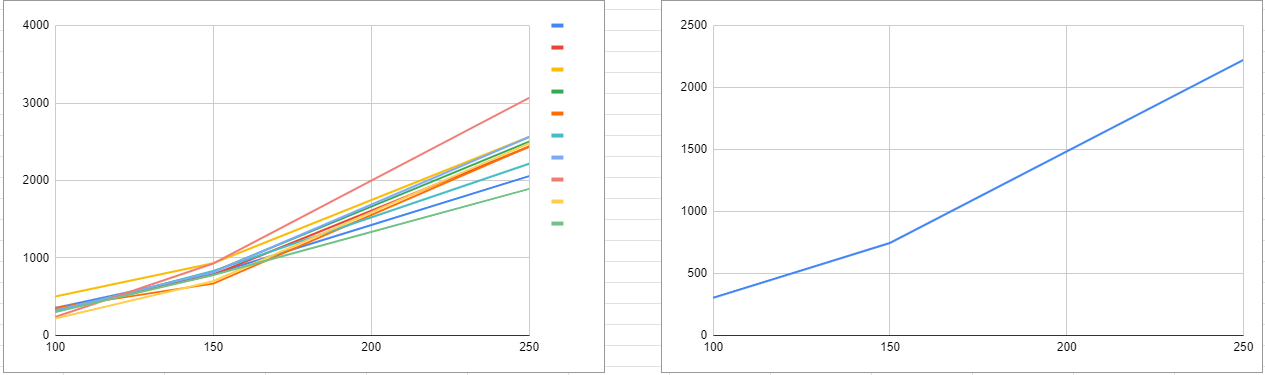
****

Figure 1: Representation of alive cells for each iteration, Figure 2: Representation of avg alive cells

We can clearly see how increasing grid size leads to an increase in number of alive cells after a certain number of generations and same initial density. Specifically, it would grow as a quadratic factor due to the computation of total grid area and same distribution across. We couldn’t see it in out graph due to the small scale of data, which is however enough to prove our hypothesis.

Similarly, for the density, we can hypothesize that since having too many neighbors and too less neighbors are unhealthy, the best grid should have an initial configuration somewhere in the middle.

To test this, I ran the same test as last experiment, except the grid size was constant and the density was varied from 10% to 90%.

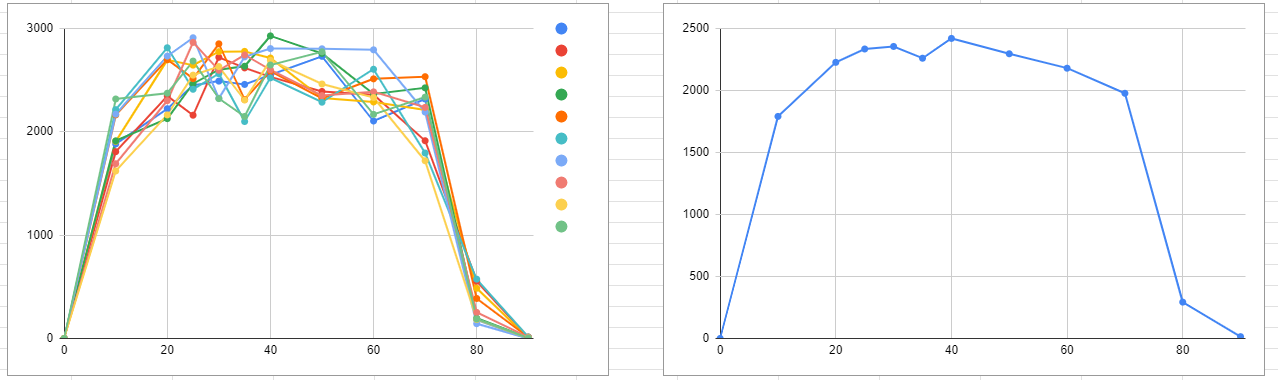


Figure 3: Total living cells after iterations vs density, Figure 4: Average living cell per density vs density

Here, we can observe again how it matches our hypothesis, which makes sense. It starts off low until 10%, goes to a plateau, and drops again near 80%. This matches the standard tests done for the original Conway’s Game of Life.

**Extensions:**

The extensions I chose for this project were adding command line interactions and buttons to the GUI.

For the command line interactions, I simply parsed the values from the string format to required variables. These included number of rows, columns, initial alive probability and number of generations to run the game for. I added usage message and a readme file respectively.

For adding buttons, I used Jpanel and Jbutton in the **LandscapeDisplay** class. I also implemented ActiveListening to my buttons for dynamic usage. Specifically, I used a button panel, added all of my buttons and then added that panel to my original Simulation panel. I implemented ActiveListener method on my buttons and set them to assign an onstate true or false. If the onstate was true, the program would run as usual, but if it was false, it would halt the program until it was true again. This was implemented in the **LifeSimulation** class, in the section to display the landscape panel.

**Acknowledgements:**

This project was made possible due to help from various online articles, youtube videos and inspiration from my classmate Rish.