

---

## CS231: Project 2

### Conway's Game of Life

---

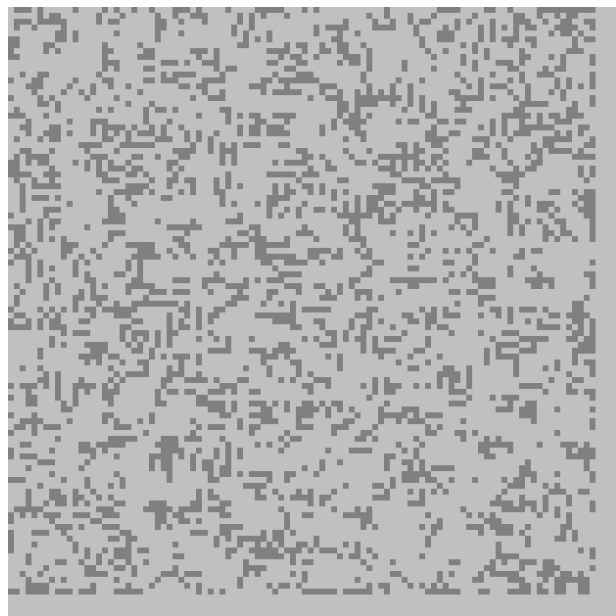
#### **Abstract**

In the project, a Conway's Game of Life has been coded and simulated. The death and life of cells has been modeled in the game according to three simple rules<sup>1</sup>:

1. Any live cell with two or three live neighbours survives.
2. Any dead cell with three live neighbours becomes a live cell.
3. All other live cells die in the next generation. Similarly, all other dead cells stay dead.

A 2D array has been used to implement the canvas that the game draws on. A 2D array can be imagined to be a grid with rows and columns, like an Excel spreadsheet. The animation on the right is composed of the grid and each dark grey square/pixel is a cell. A cell is the smallest part/unit of a grid.

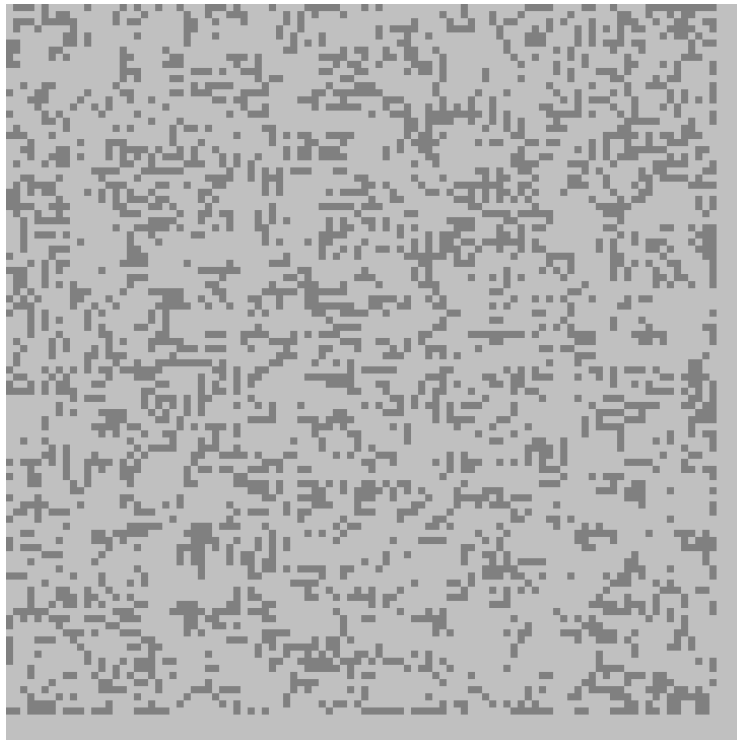
Each slide change on the animation on the right is the state after one more generation i.e., after the rules of the game are applied to every cell once.



A GIF depicting 500 stages of a simulation of the Game of Life (in folder because PDF doesn't support GIF)

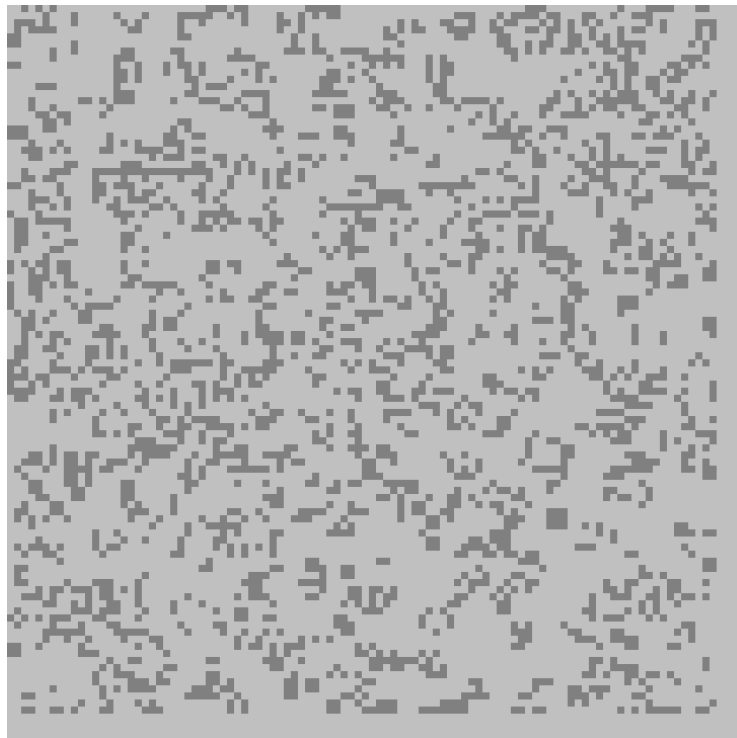
---

<sup>1</sup> "Conway's Game of Life," in *Wikipedia*, September 22, 2021, [https://en.wikipedia.org/w/index.php?title=Conway%27s\\_Game\\_of\\_Life&oldid=1045768148](https://en.wikipedia.org/w/index.php?title=Conway%27s_Game_of_Life&oldid=1045768148).



**(above) Initial state of the board**

**(below) State of the board after one update**



Note: It was asked on the project page that why can't we move each cell though to the next generation one-by-one. The answer is that the change in the state of a cell will lead to its neighbours getting a different value for the number of neighbours they have, causing miscounting of alive cells.

## **Extension 1**

For this extension, I've gone through the *Cell* and *Landscape* classes' main and other methods and added comments stating what is the last line where an object is used or when it is cleaned out of memory.

In general:

1. Objects that are created in constructors are usually removed from memory when the method in which the instance of their class was created.
  - a. The exception to this is if the object reference is assigned to the fields or local variables of any other method. In that case, they are removed when the object of the other class(es) is/are removed. In case of a local variable, they are removed from memory when that method ends.
2. Objects referenced using local reference variables are cleaned out of the memory when the method in which the variables exist ends.

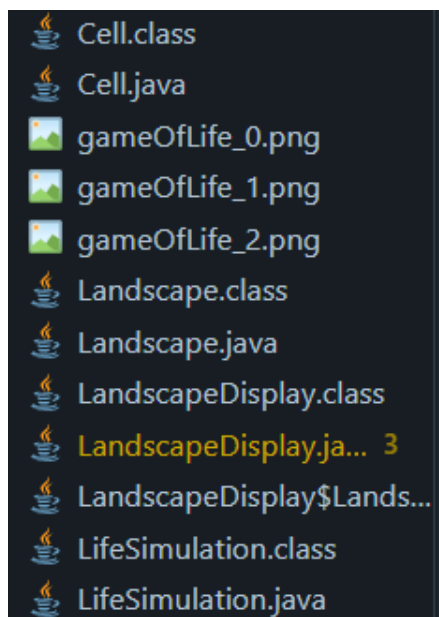
## Extension 2

For this extension, I've gone through the *LifeSimulation* class and added functionality for the user to be in full control of the simulation. The following functionality has been added:

1. The user can enter command line arguments for number of rows and columns. If the user does not enter anything, it defaults to 100 for both values.
2. The user is now asked every time if they want to continue the simulation further or not. They can enter true to continue or false to stop.
3. Lastly, the user can take a screenshot at every step. These will be sequential and have names that are of the form *gameOfLife\_0.png*, *gameOfLife\_1.png* and so on.

```
PS C:\Users\Parth\OneDrive\College Classes\CS231\Projects\Project 2 Extensions\Extension02> javac LifeSimulation.java
PS C:\Users\Parth\OneDrive\College Classes\CS231\Projects\Project 2 Extensions\Extension02> java LifeSimulation 50 50
Do you want to take a screenshot? Enter true or false: true
Screenshot saved as: gameOfLife_0.png
Do you want to continue? Enter true to continue or false to stop: true
Do you want to take a screenshot? Enter true or false: false
Do you want to continue? Enter true to continue or false to stop: true
Do you want to take a screenshot? Enter true or false: false
Do you want to continue? Enter true to continue or false to stop: true
Do you want to take a screenshot? Enter true or false: true
Screenshot saved as: gameOfLife_1.png
Do you want to continue? Enter true to continue or false to stop: true
Do you want to take a screenshot? Enter true or false: true
Screenshot saved as: gameOfLife_2.png
Do you want to continue? Enter true to continue or false to stop: false
PS C:\Users\Parth\OneDrive\College Classes\CS231\Projects\Project 2 Extensions\Extension02> █
```

A screenshot of the terminal (above). The resulting file structure (below)



A screenshot of a file explorer window showing the project files and folders. The files are listed in a dark-themed interface. The files are: Cell.class, Cell.java, gameOfLife\_0.png, gameOfLife\_1.png, gameOfLife\_2.png, Landscape.class, Landscape.java, LandscapeDisplay.class, LandscapeDisplay.java... 3, LandscapeDisplay\$Lands..., LifeSimulation.class, and LifeSimulation.java. Each file has a small icon to its left, representing its type (e.g., a coffee cup for Java files, a landscape for PNG files).

This has been implemented by editing the *LandscapeDisplay* class to add a private int variable called *screenshotIndex* which keeps track of index numbers for the screenshots, a getter-setter pair for *screenshotIndex*, and a while loop in the *LifeSimulation* class to keep asking the user till when they want to continue and if they want to take a screenshot.

## References/Acknowledgements

The *LandscapeDisplay* class was retrieved from <https://cs.colby.edu/aharper/courses/cs231/f21/labs/lab02/LandscapeDisplay.java>. I looked at the data members of the *Graphics* class at <https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html> to get an idea of the functions that I had to use, as recommended. I looked at the [Wikipedia page for Conway's Game of Life](#) to use test scenarios for my simulation. Prof. Harper helped me figure out where the *draw()* methods would go in the classes. I have imported classes like *Scanner* and *ArrayList* from the *java.util* package.