Saishnu Ramesh Kumar
Harry Atulbhai Patel
Rong Jun Leong
Chester Lee

CSCI 154 - Project 1 Report (Conway's Game of Life)

Introduction:

Conway's Game of Life is a cellular automaton created by a British mathematician, John Horton Conway. The goal of creating this game was to help define unpredictable cellular automatons. The original concept of cellular automaton was introduced by John von Neumann in the 1940s and Conway based his Game of Life on this. A cellular automaton is essentially a collection of cells that can be arranged in a grid-like form where every cell changes its state according to the set of rules given by the neighboring cells. Unfortunately, in 2020, Conway passed away due to symptoms of COVID-19. In this report, we will discuss the following: motivation, problem statement, rules, approach, interesting properties, interesting patterns/objects, and results obtained from the code that we created using Python.

Motivation:

The motivation of the Game of Life is to help in defining the unpredictability of cellular automatons and help create a universal set of them. Using a set of simple rules, we can then produce complex patterns and behaviors that can be obtained from it. With the implementation of the Game of Life, we are able to then design more efficient algorithms for cell evolution within a grid. This also helps us to explore different types of patterns and notice their transitions from static to dynamic.

Problem Statement:

Our problem statement is as follows, we are trying to figure out the rules that would need to be followed in the game. This also includes us trying to find any interesting properties or patterns/objects through the program. As well as observing how each pattern evolves overtimes and how they can affect one another.

Rules:

In the Game of Life, each cell will be interacting alongside its neighbors of cells that are either horizontally, vertically, or diagonally adjacent to each other. The rules of this game are as follows:

1. Underpopulation is when live cells containing less than two live neighbors die.
2. Overpopulation is when live cells containing more than three live neighbors die.
3. Any live cells that contain two or three live neighbors will move on to the next generation of cells.
4. Any dead cell that has exactly three live neighbors will become alive.

Approach:

Our approach to this project is as such, using Python, we programmed the game with the implementations of two libraries called pygame and numpy to simulate the Game of Life. We implemented the following functions: create_grid, update_grid, draw_grid, and main. In the create_grid function, we used the numpy library to generate a random grid, and each cell in the grid is either 0 (dead) or 1 (alive) based on the probability distribution of 0.7 and 0.3 where the probability randomly assigns 0's or 1's to each cell where 0.7 is 0 (dead) and 0.3 is 1 (alive). Moving on, the update_grid function, follows the rules of Game of Life which checks the number of neighbors that are alive and determines the new state of the cells. It will iterate in each cell in a grid and once the function determines the state of the cell, it will update and return the new grid. The draw_grid function always displays the current state of the grid based on the values of the cell. If the cell is dead, the grid displays gray color but if the cell is alive, it is shown as black. The main function essentially calls all functions to get the program to display the Game of Life.

Interesting Properties:

Some interesting properties of the Game of Life are that it is Turing complete meaning that given enough time and memory with the set of instructions, it is able to solve computational problems no matter the complexity. It is also a zero-player game as it evolves as determined by the initial state without the need for further human input. The game also contains self-replication and

shows us the different interactions of various patterns found which can also result in random outcomes throughout the duration of the game.

Interesting Patterns/Objects:

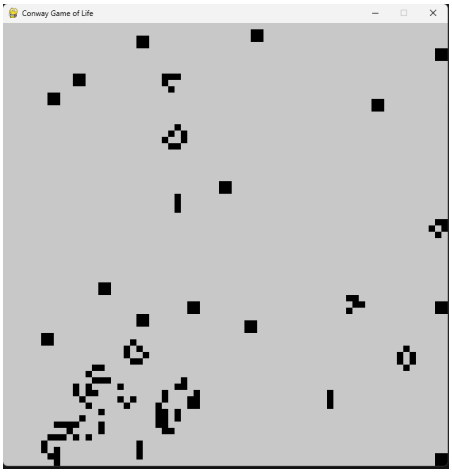
The interesting patterns/objects we discovered about the game from the code we had implemented are listed as such:

1. **Still Life:** No pattern changes, it stays in one spot, and does not change over time.
2. **Oscillator:** A pattern that changes but will repeat itself after a while, it maintains its position and looks like a spinning fan.
3. **R-Pentomino:** A pattern containing around five connected cells along its edges, it is unstable and is the first pattern that is not finite, it will also die out of becoming stable after ten generations.
4. **Glider:** Considered an important pattern to the game, this pattern moves continuously along the grid like it is gliding on the screen. The glider moves in a cyclical motion.
5. **Glider Gun:** Similar to the glider, this pattern would grow and result in a continuous stream of gliders. The glider gun grows independently and consists of a group of cells that will periodically emit gliders.
6. **Lightweight Spaceship (LWSS):** It is the smallest orthogonal spaceship and possesses a tail spark to disrupt other objects while it is moving.
7. **Beehive:** The beehive consists of 6 cells and it is the second most common still life. This pattern is static and it does not evolve over time. It will be stable for the most part unless disturbed by its neighboring cells.
8. **Ship:** The ship pattern moves in a straight line while maintaining its shape throughout the grid. It adds one cell to its corner to create a fleet and if it removes one from the corner it is a boat. If both cells are removed it is a tub.

Results:

From observing our program of the Game of Life, we were able to analyze and see various patterns and movements in the game. The screenshot provided below is the output of a frame

from the running program displaying the movement of the alive cells (shown in black) and dead cells (shown in gray). The grid displays the paths of the various patterns found within the Game of Life and shows the transition from the start to the end of the game. The game starts off with a lot of alive cells moving around the grid and after a while, a number of them start to die out which results in a stop to the game. At the final end of the game, some cells are still moving within their spot but do not replicate themselves as compared to the start of the game.



Conclusion and Contributions:

To conclude this report, Conway's Game of Life is an interesting example of a cellular automaton and it is a well-known invention that has been popular in the programming world for decades. The game is based on simple rules which can produce complex and unpredictable patterns over time. We have implemented the program to support functions that help create a grid, update the cell state, and display the evolution of the cells. In the simulation, we observed various patterns and their cell distribution while still observing the rules set in place. Conway's Game of Life has an important contribution to the development of a complex system that scientists use to simulate real-world phenomena such as the spreading of diseases, traffic patterns, and many more. Overall, Conway's Game of Life has developed as a trendy subject that we can use to explore the properties of cellular automata and their other applications.

Feedback: For this section of the project, we did not get much feedback about changes but overall, the slides were well-presented and organized and did not include too much text. Nonetheless, we do think that we could have improved on it by going through more about the approach and how we tackled creating the program as well as the properties. We believe we could have concluded the project presentation better as it seemed a little too vague.