# Conway's Game of Life and its Extended Implementations

Conway's Game of Life is a cellular automaton devised by the British mathematician John Horton Conway in 1970. It is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves.

## Aim

The aim of this project is to analyze the emergent behavior of cells when the initial rules of Conway's Game of Life are modified. By changing the rules to non-deterministic ones and introducing new behaviors such as selfishness, sacrifice, and stochastic growth, we aim to deepen our understanding of cellular automata.

## File Structure

| File | Purpose |

|----------------------------|-----------------------------------------------------------------|

| `GameOfLife\_CustomPlace.py`| Original Game of Life implementation with custom cell placement|

| `GameOfLife\_SelfishRules.py`| Implementation where cells exhibit selfish behavior |

| `GameOfLife\_SacrificeRules.py`| Implementation where cells exhibit sacrifice behavior |

| `stochastic\_GOL.py` | Implementation with stochastic growth properties |

## How to Play

### Game of Life (Original)

- Run `GameOfLife\_CustomPlace.py`.

- Left-click to place live cells, right-click to remove.

- Press spacebar to start/stop the simulation.

- Press 'r' to reset the grid.

### Selfish Rules

- Run `GameOfLife\_SelfishRules.py`.

- The cells exhibit selfish behavior with a specified level of selfishness.

- Specify the level of selfishness when prompted.

### Sacrifice Rules

- Run `GameOfLife\_SacrificeRules.py`.

- The cells exhibit sacrifice behavior where they sacrifice themselves if they have a certain number of neighbors.

- Specify the number of neighbors for sacrifice when prompted.

### Stochastic Growth

- Run `stochastic\_GOL.py`.

- Cells exhibit self-healing and self-controlled growth properties.

- Specify the probability of cell death when prompted.

## Installation

1. \*\*Clone this repository:\*\*

```bash

git clone https://github.com/yourusername/GameOfLife.git

```

2. \*\*Navigate to the directory:\*\*

```bash

cd GameOfLife

```

3. \*\*Run the desired implementation file as mentioned above.\*\*

### Dependencies

The main dependencies for running the various versions of Conway's Game of Life are:

1. \*\*Python\*\*: Python is the primary programming language used for implementing the game and its variations. Ensure you have Python installed on your system. You can download it from the [official Python website](https://www.python.org/downloads/).

2. \*\*pygame\*\*: Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries. To install pygame, you can use pip, Python's package manager.

3. \*\*numpy\*\*: Numpy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Install numpy using pip.

4. \*\*matplotlib\*\*: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is used for plotting the generation vs. alive cells graph. Install matplotlib using pip.

5. \*\*csv\*\* (optional): The csv module is used for reading and writing CSV files. It is used in one of the implementations to save data to a CSV file.

### Installation Instructions

You can install the dependencies using pip, which is the package installer for Python. Open your terminal or command prompt and execute the following commands:

```bash

pip install pygame numpy matplotlib

```

If you choose to run the version that utilizes the `csv` module, you can install it with:

```bash

pip install csv

```

These commands will install all the necessary dependencies required to run Conway's Game of Life and its extended implementations.

Once the dependencies are installed, you can run each version of the game by executing the corresponding Python script. For example:

```bash

python GameOfLife\_CustomPlace.py

```

Ensure that you are in the correct directory where the Python scripts are located when running the commands. With the dependencies installed, you should be able to enjoy exploring the different implementations of Conway's Game of Life!

### Functionality

1. **Game of Life Original**

This file is the original implementation of the game of Life following all the natures rules.The Upgrade Grid Function is designed to continuously update the grid by checking the rules of Conway's Game of Life. These rules are as follows:

- If a living cell has zero or one neighbors, it will die due to loneliness. It will also die if it has more than four neighbors.

- If a dead cell is surrounded by exactly three neighbors, a birth will occur, and the cell will become alive.

**Features**

1. Stable patterns
2. Can produce highly complex patterns depending on the initial configurations
3. It is Turing Complete, it can simulate any Turing machine
4. **Game of Life with Sacrifice Rules**

This file implements the Game of Life with modified rules where the alive cells play a pregame called “n-die game”. The cells play the pregame where the cell sacrifices itself if it has “n neighbours and then the nature rules apply. The Upgrade Grid function incorporates the sacrifice rules as described in the paper by Roland Mühlenbernd (2014)., et al

The following is a description of the game of life. The set of alive cells in the current round of play is denoted by CL. For each cell ci in CL, its neighborhood Ni is defined as the set of all neighboring cells. The non-deterministic n-die game is a game with three phases and is defined for any natural number n between 1 and 8.

To begin with, a list called AL needs to be initialized. This can be done by randomly selecting all living cells ci from CL and adding them to the list. Once the list has been created, a decision needs to be made about which cells should be sacrificed. For each cell ck in the list AL, if the number of its neighbors is equal to n, it is labeled as dead and removed from the neighbor list of all cells ci in CL. Finally, the rules of nature of the game of life are applied to determine the next generation of cells

**Features**

1. Number of neigbours when sacrifice happens can be given as input “n”
2. Optimization of population for n values
3. When n = 2, new stable patterns other than the defined ones in conways game of life.
4. **Game of Life with Selfish Rules**

This file implements the Game of Life with modified rules where a cell can exhibit selfishness based on a given percentage. The Upgrade Grid function incorporates the selfish rules as described in the paper by M.R.Lauer, et al.

The rules are defined as:

1. The first rule states that a chip can survive by eliminating its neighbors if it is surrounded by four or more chips.
2. The second rule allows a chip to survive even if it has no or one neighbor, as long as its vitality factor is greater than or equal to one.
3. The third rule allows for births in an empty cell with three or four neighbors.
4. Finally, the fourth rule designates a new chip as selfish based on a given level of selfishness that is a parameter of the population.

**Features**

1. Level of selfishness can be given in terms of percentage.
2. Optimization of population for some levels of selfishness
3. **Game of Life with Probability**

This file implements the Game of Life with modified rules where probability is introduced into the cell death and additional weighted masks are introduced to neighbours count.

The upgrade Grid function upgrades the rules as mentioned in the paper The modifications made to the standard Conway's model are done in two ways:

1. The "Deaths" of "live" cells are made stochastic with a certain probability 1/Pdeath, which is a model parameter.

2. The number of "live" cells in the 8-neighborhood of each cell is counted using a weighted summation with rounding up the summation result instead of a simple summation of their binary values.

**Features:**

1. The probability and mask can be given as input.
2. Cells expand infinitely for some probabilities and can heal themselves.
3. Can show some coherent growth properties.

### Acknowledgments

1. John Horton Conway for creating Conway's Game of Life.

2. The Python community for developing and maintaining the libraries used in this project.

Enjoy exploring the various behaviors of cellular automata in Conway's Game of Life!

---