Game of Life

Documentation

Contents

[About 2](#_Toc100263806)

[Rules 3](#_Toc100263807)

[Documentation 4](#_Toc100263808)

[Structure 4](#_Toc100263809)

[Launching 4](#_Toc100263810)

[Main logic 4](#_Toc100263811)

[Algorithm 5](#_Toc100263812)

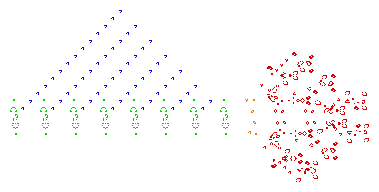
[Source code 5](#_Toc100263813)

# About

The Game of Life is a cellular-automaton, zero player game, developed by Cambridge mathematician John Conway in 1970.

The game is played on an infinite grid of square cells, and its evolution is only determined by its initial state.

This game became widely known when it was mentioned in an article published by Scientific American in 1970. It consists of a collection of cells which, based on a few mathematical rules, can live, die or multiply. Depending on the initial conditions, the cells form various patterns throughout the course of the game.



A screenshot of a puffer-type breeder (red) that leaves   
glider guns (green) in its wake, which in turn create gliders (blue)   
(by Wikipedia.org)

## Rules

* **For a space that is populated:**
  + Each cell with one or no neighbors dies, as if by solitude.



* + Each cell with four or more neighbors dies, as if by overpopulation.



* + Each cell with two or three neighbors survives.



* **For a space that is empty or unpopulated**
  + Each cell with three neighbors becomes populated.



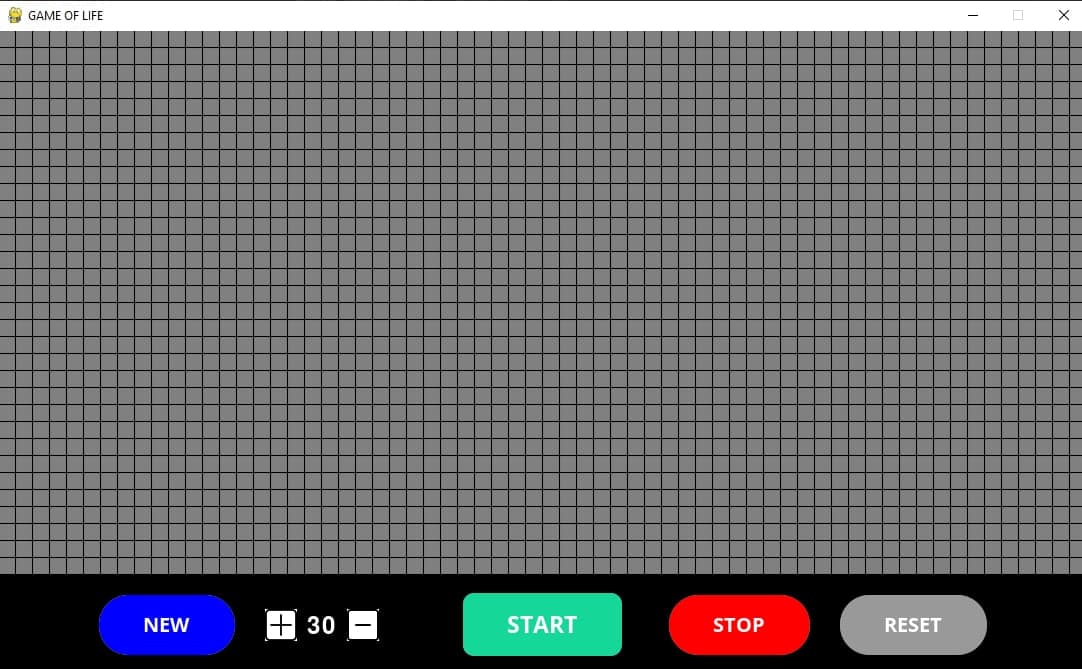
# Documentation

## Structure

The source code is written in the one file: main.py. The Assets folder contains all the images used for the project.

## Launching

The game is launched via main.py file. You need to have PyGame library installed to launch it.



## Main logic

The program has only one page. The page contains:

* *The Grid* - consists of 64x32 initially dead (gray) cells. By clicking on the cells becomes alive if it's dead and dead if it's alive. After setting up the desired board press START to initiate the simulation
* *Buttons*:
  + «NEW» — wipes the grid (changes all the cells status to dead);
  + «START» — initiates the simulation;
  + «STOP» — stops/pauses the simulation;
  + «RESET » — resets the grid to the state that it was before the START button was clicked;
  + «+» — fps speed boost;
  + «-» — fps reduction in speed;

## Algorithm

The solution is very basic. The board is a 2-dimensional board filled with ones (alive cell) and zeros (dead cell). To get the next generation, it loops through every cell calculating alive cells around them and based on the defined rules the function creates and returns grid of the next generation followed by screen refresh, to display the newly returned grid. The process goes on as long as the user hasn’t pressed the button STOP or closed the window.

## Source code

The game uses pygame library as it’s core. We set the size, title, font size of the window then we define several global variables to make our work easier. Such variables are the color variables, default fps the game will run on and 2 matrices (initial\_grid and grid) holding all the dead cells.

## 

We also load all the images needed for the game, while giving them their coordinates.

nextGen := takes as parameters the current grid and the size of it. Returns future generation as grid.

screen\_refresh := redraws the screen.

array\_copy := as first parameter takes an array from which we copy the values, as second parameter takes an array to which we copy the values. As last parameters it takes the size of the arrays. Arrays must have similar sizes.

gol := main function of game of life. Takes the current grid and its size as parameters. Uses nextGen to get the future grid, then uses array\_copy to replace the current grid with the future one. After which redraws the board with screen\_refresh.

draw\_board := takes a grid and its size as parameters and draws it on the board.

draw\_buttons := draws the buttons on the board.

main := keeps the game running while refreshing the information on the board based on FPS. Manages the logic of the buttons that are being pressed.