

CELLULAR JS

By FEUKOUA GERAUD CHRINAER

Red words:

Null grid: when all the cells in the grid are death

Introduction

Few Weeks Ago, I made a Discovery upon researching on **Cellular Automatons**, an how us humans have struggled to simulate living things in a computer.

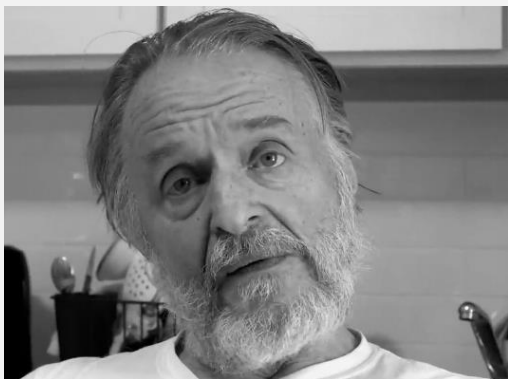
A CELLULAR AUTOMATON is a grid of cells of any shape and size where the activity and future of one cell, is inherently determined by its neighborhood. Scientist have had troubles in trying to guess what can be the relationship between those cells that can make them to look alive when their effects are all combined in a 2D grid.

Very Earlier approaches where to use complex mathematical Equations to represent those relations but that approach was computationally, and knowledgeably expensive in those early days of computers where the most costly game was **Atari (which is pixelated)**. The approach was very complex and do not return any helpful outcome.

But a little tiny advantage of that approach was that, **COMPLEXITY ARISED**. This so called complexity is not computational complexity but the Unpredictable nature of the output such fractals exist again today and some are even infinite e.g. Dragon Fractal, Mandelbrot Set and more.

THE GAME OF "LIFE"

That problem was solved by a Mathematician John Conway () who proposed the **Game of Life**

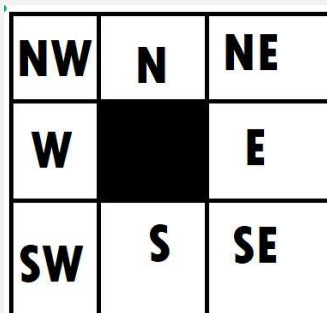


The Ultimate Goal of John Conway, was to make complexity to arise from Simplicity i.e. in his Game means simple rules.

THE Conway's GAME OF LIFE RULE

Considering each cell in a grid has a neighborhood of **8** surrounding Cells **N,E,S,W,NW,NE,SW,SE**.

As illustrated in *fig1a* below



NW	N	NE
W		E
SW	S	SE

FIG1A

The Cell in Highlighted in black is either active or inactive according to the following rules

Activation (BORN):

ACTIVATED NEIGHBOURS EQUALS 3 AND HE IS INACTIVE

Survival:

ACTIVATED NEIGHBOURS EQUALS 3 or 2 AND HE IS ALIVE

DEATH (By loneliness):

ACTIVATED NEIGHBOURS SMALLER THAN 2 AND HE IS ALIVE

DEATH (By Congestion):

ACTIVATED NEIGHBOURS GREATER THAN 3 AND HE IS ALIVE

These rules are then executed for all the cells in the grid each having their Neighborhood. It's important to note that but "If while executing a cell die reaching its neighbor the cell will have already being death and it will cause global loneliness causing in a **null grid**"

To fix that problem, John Conway Proposed that, we can create an empty array of all zeroes, let's call it **At+1**. **At+1** at any index **I** is updated with the new state of the cell at **At** at the same index **I**

i.e., $A_{t+1}[i][j] = A_t[i][j]$

This then causes the values of A_t to remain constant during the evolution of the cells at time t .

But now that we know how to avoid that problem we talked about above, how then is evolution coming along? Well to make evolution visible, after we updated A_{t+1} , A_{t+1} has the new grid state.

Hence A_{t+1} is the future state of the grid. The finishing the filling of the array at A_{t+1} for step t ,

$A_t[i][j] = A_{t+1}[i][j]$

And the Evolution goes for $t, t+1$, and all integer values of t

Putting now this in a Pseudo Code, we have,

```
while Evolution do

  initialise At+1 to zeroes

  for i = 0 to gridWidth
    for j = 0 to gridHeight
      At[i][j] = At+1[i][j]
    endFor
  endFor

  for i = 0 to gridWidth
    for j = 0 to gridHeight
      At+1[i][j] = At[i][j]
    endFor
  endFor

end while
```

This is pretty short for a 2 decade research but as John Conway wanted, **Complexity arises out of simplicity.**

EXPERIMENTS:

The outcome led to the discovery of species like,

Gliders, Rotors, Multipliers, Yes and even Logic Gates can be simulated in

Conway's Game of life is said to be Turing Complete

CELLULAR JS IN THIS FIELD

Over the concept above, CELLULAR JS was built to facilitate the usage of cellular automata. The user will not have to write code when he will be using the Library's file and when he will be using the app **CELLULAR**.

In **CELLULAR**, the user will be able to create his own rules and to do more discoveries without even have read this document

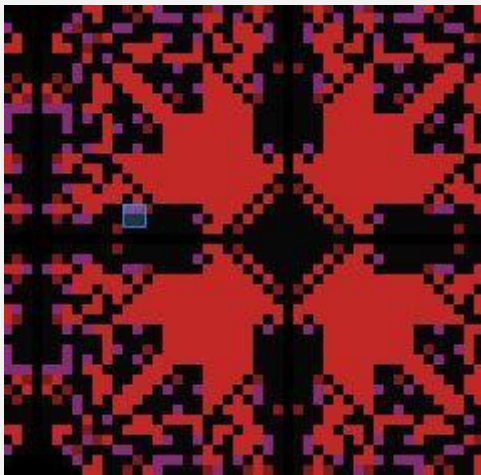
WHAT MAKES CELLULAR JS SPECIAL

- The cells are simulated in any colour of your choice,
- The user can create their own patterns and rules
- The cells evolution is endless, and no errors signaled
- The grid can be rectangular
- I made in such a way that it can be added to any website, or other apps
- There is a bloom effect bringing your grid to lightness

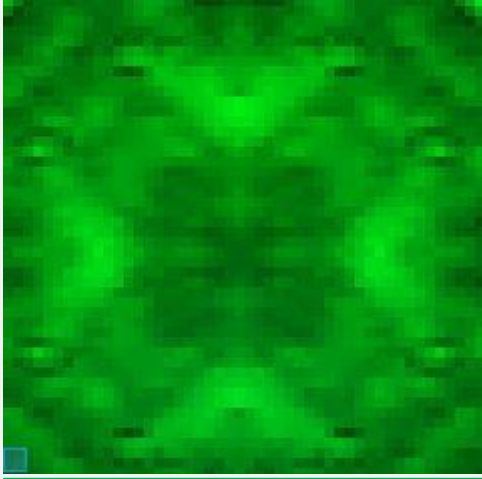
The user can create graphics with the use of existing rules. I edited such rules include,

```
"coagulation", "maze", "coral", "serviettes", "replicator",  
  "flakes", "longlife", "assimilation", "sac", "seeds", "group",  
  "maze2", "born"
```

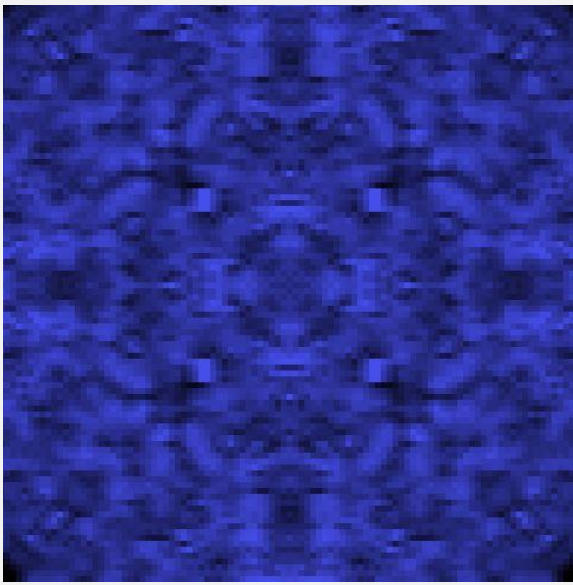
IMAGES CREATED IN CELLULAR JS



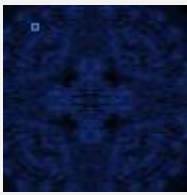
BLADE BROS



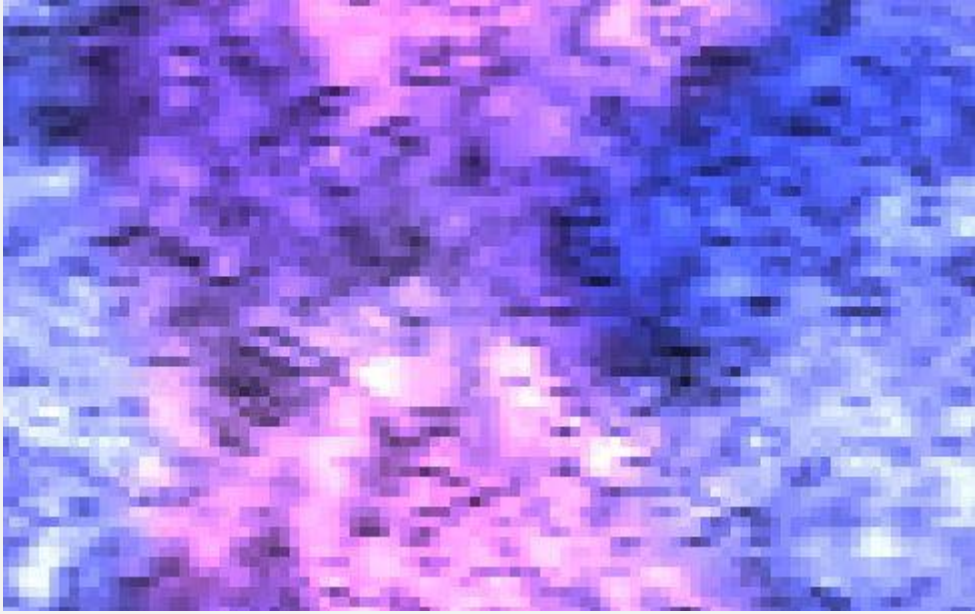
GREEN SYMMETRY



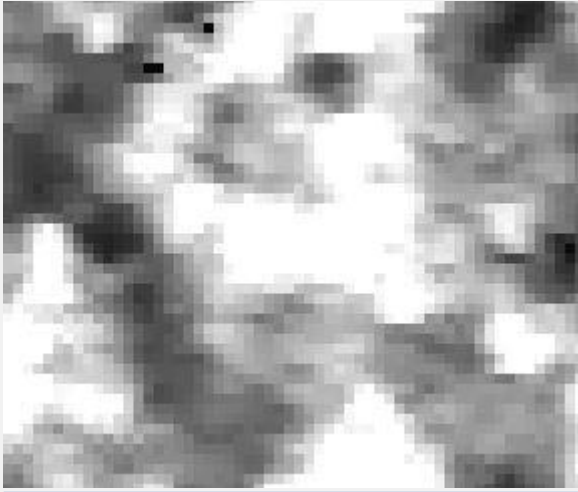
LUCKY BLUE FLOWER



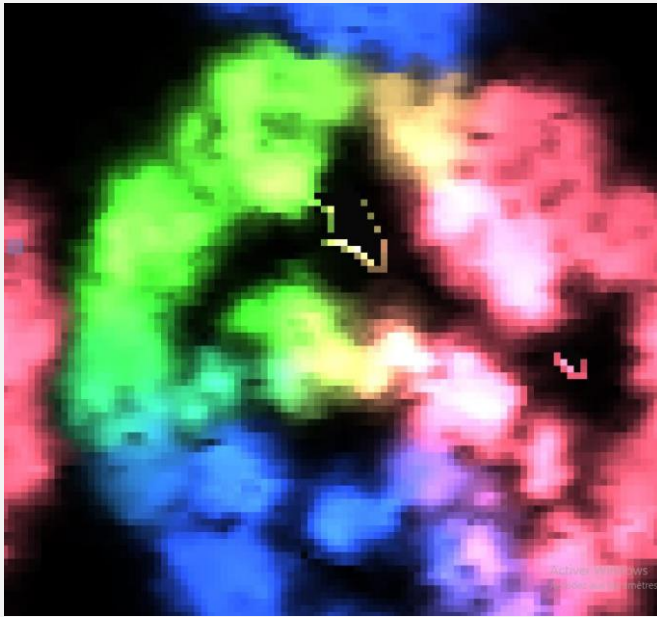
Tiny one



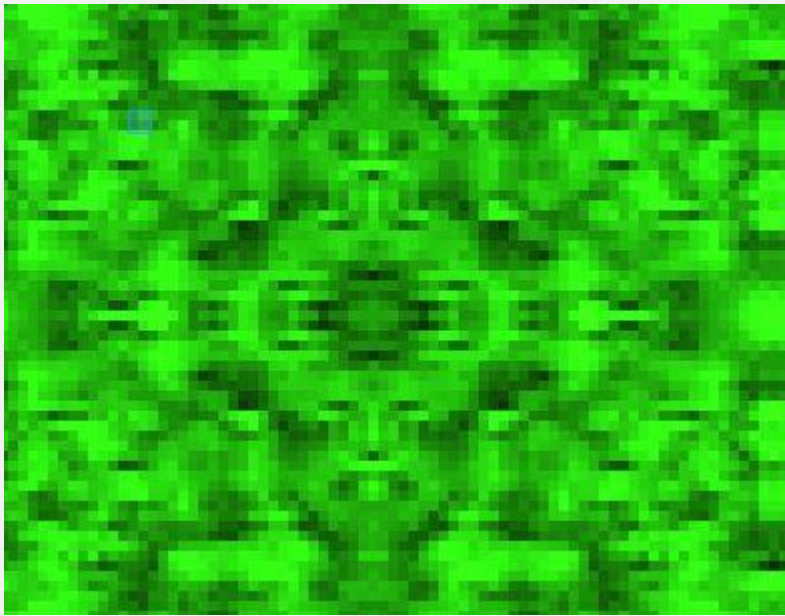
NEBULA



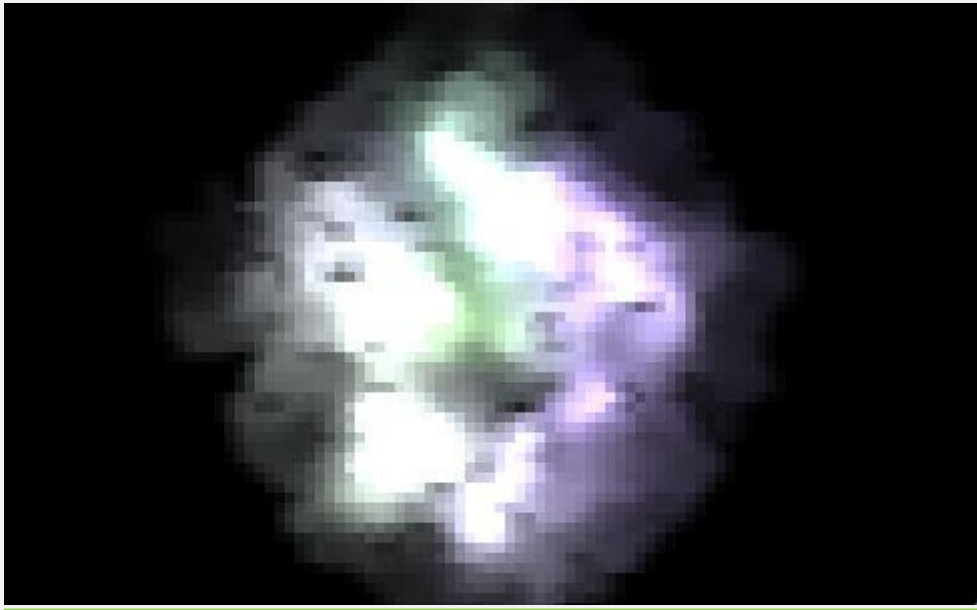
CLOUDS



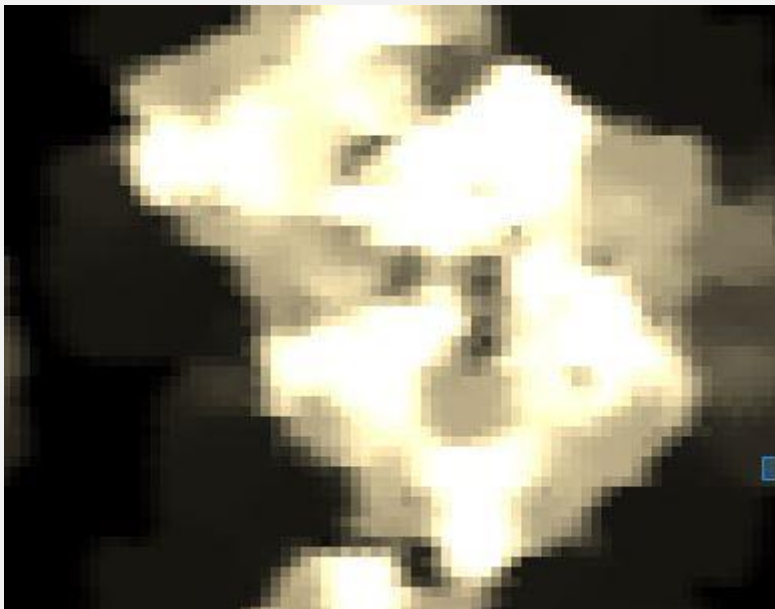
COLOURFULL SMOOKES



GREEN SYMMETRY 2



EXPLOSION



WOW

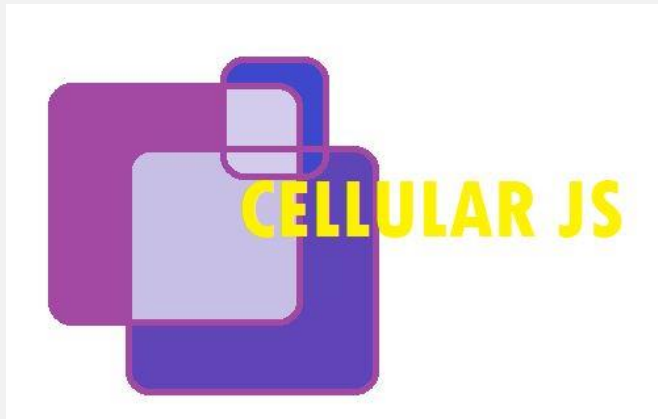


GLIDERS SHOW

THANKS FOR READING, SUBSCRIBE TO MY YOUTUBE CHANNEL

ChrinaerDev

And Follow me on GitHub



GSOFT

At the core of technology



