



Instituto Politécnico Nacional

ESCUELA SUPERIOR DE CÓMPUTO

JUEGO DE LA VIDA

Autor:

Juan Carlos Garcia Medina

PROFESOR:

Dr. Genaro Juárez Martínez

MATERIA:

Sistemas Complejos

, 24 de abril de 2020



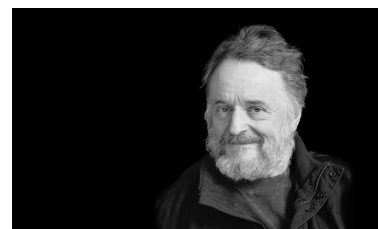
Índice general

1. John Horton Conway (1937-2020)	1
2. Game of Life	2
2.1. Patrones de Life	3
2.1.1. R-pentomino	3
2.1.2. Objetos Still Life	3
2.1.3. Oscillators	4
2.1.4. Gliders	5
2.1.5. The Queen Bee Shuttle	5
3. Implementación Game of Life	6
3.1. Archivo de configuración	12
3.1.1. Con una nueva distribución de Células vivas	13
3.1.2. Conservando la matriz que representa a la cuadrícula	13
3.2. Gráfica	14
3.3. Regla de evolución	18
3.3.1. R(2,2,2,2)	19
3.3.2. R(3,3,3,3)	20
3.3.3. R(1,2,3,4)	21
3.3.4. R(2,4,6,8)	22
3.3.5. R(2,3,3,4)	23
Bibliografía	25

Capítulo 1

John Horton Conway (1937-2020)

«Esta pequeña sección está dedicada a John Horton Conway, Por sus contribuciones a la matemática y particularmente por su conocido Juego de la vida que fue crucial para el entendimiento de lo que se conoce como emergent complexity o sistemas auto-organizables contribuyendo a lo que hoy se conoce como sistemas complejos.»



John Conway hizo otras importantes contribuciones en teoría de grupos, teoría de números, álgebra, geometría, topología, teoría de nudos, combinatoria, teoría de juegos y física teórica entre otras. Fue autor de más de 10 libros y cerca de 150 artículos de alto nivel. Siempre estuvo preocupado por difundir sus conocimientos no solo entre la comunidad matemática sino que realmente creía en la transdisciplinariedad y en ayudar a los más jóvenes, porque ellos son los encargados, en primera instancia, de hacer que se continúe en el progreso del conocimiento.

Como persona los que tuvieron la oportunidad de conocerlo incluyendo a quien sería autora de su biografía (Siobhan Roberts) coinciden en que era una persona muy amable a quien le encantaba contar historias e incapaz de contestar a una pregunta con un simple "sí" o "no", era un genio, un gigante como muchos lo describieron, La universidad de Princeton a través de su sitio web dio la noticia de su lamentable fallecimiento, incluyendo un obituario de las personas que lo conocían personalmente, De Diana Conway podemos leer esto: **“John fue el ser humano más fascinante que he conocido, él no solo estaba interesado en las matemáticas, estaba interesado en todo”**. [1]

La sucesión que se muestra a continuación es muy vista en páginas de acertijos o incluso he conocido personas que han tenido que programar para calcular el n -ésimo término, sin embargo pocos saben que esta sucesión tiene nombre propio: Sucesión de Conway:

3, 13, 1113, 3113, 132113, 1113122113, 311311222113, ...

Capítulo 2

Game of Life

Game of Life o juego de la vida en español, es un autómata celular desarrollado por John Conway en 1970. Consiste en una cuadrícula infinita de células cuadradas y su evolución es determinada solo por el estado inicial.

Las células pueden estar vivas o muertas, cada célula tiene 8 vecinas adyacentes.

Las reglas son simples y describen la evolución de la cuadrícula:

- **Nacimiento:** Una célula que está muerta en el tiempo t estará viva en el tiempo $t + 1$ si exactamente 3 de sus ocho vecinas estuvieron vivas en el tiempo t .
- **Muerte:** Una célula puede morir por:
 - **Sobrepoblación:** Si una célula está viva en el tiempo t y 4 o más de sus vecinas están también vivas en el tiempo t , la célula pasará a estar muerta en el tiempo $t + 1$.
 - **Soledad:** Si una célula viva en el tiempo t solo tiene una vecina o ninguna vecina viva, estará muerta en el tiempo $t + 1$.
- **Supervivencia:** Una célula sobrevive del tiempo t al tiempo $t + 1$ si y solo si 2 o 3 de sus vecinas están vivas en el tiempo t .

2.1. Patrones de Life

2.1.1. R-pentomino

Inicia con 5 células, comienza a complicarse rápidamente, con este patrón se pueden descubrir algunos otros más simples.

El R-pentomino es el primer patrón que Conway encontró que desafió sus intentos por simularlo a mano. De hecho, el patrón eventualmente se vuelve 'estable' o fácil de predecir, pero esto no sucede sino hasta que han pasado 1103 unidades de tiempo. [2]

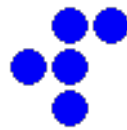


Figura 2.1: R-pentomino

2.1.2. Objetos Still Life

Algunos de los objetos más comunes en el juego de la vida permanecen igual en cada paso. Ninguna célula viva muere y no nacen nuevas células. Conway llamó a este tipo de objetos Still Life. El más común de ellos es simplemente un cuadrado de 2x2 células vivas. [3]



Figura 2.2: Block

Algunos otros tipos de Still Life que se pueden encontrar se muestran en la siguiente Figura.

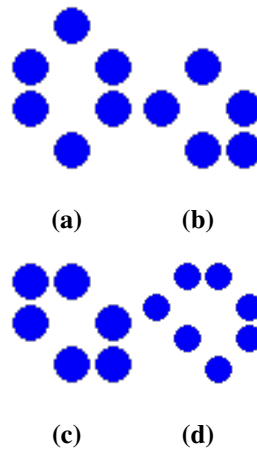


Figura 2.3: (a) Beehive (b) Boat (c) Ship (d) Loaf

2.1.3. Oscillators

Los osciladores son objetos que cambian de un paso al otro, pero eventualmente se repiten. El tipo más simple es el oscillator de periodo 2, o aquellos que se repiten después de dos pasos. El más común es el blinker, que consiste de 3 células.



Figura 2.4: Blinker

Otro oscillator es el Toad, que también aparece en el R-pentomino después de 737 pasos, es destruido por una explosión después de solo 14 pasos.



Figura 2.5: Toad

2.1.4. Gliders

Uno de los descubrimientos más emocionantes que se encontraron primero en el juego de la vida son los objetos que se mueven. Estos patrones en movimiento se llaman gliders. Los gliders se repiten cada 4 generaciones, pero se desplaza diagonalmente una célula.



Figura 2.6: Glider

2.1.5. The Queen Bee Shuttle

Un patrón muy importante que aparece después de correr el R-pentomino durante 774 unidades de tiempo es **The Queen Bee Shuttle**.



Figura 2.7: The Queen Bee Shuttle

Aunque en el entorno del R-pentomino no dura mucho tiempo, por sí sola primero se mueve a la derecha, pero después de unos pocos pasos, se produce un Still Life llamado Beehive a su izquierda y se da la vuelta de nuevo. Conway la nombró **The Queen Bee Shuttle**. por esta razón. Aunque después choca en la siguiente unidad de tiempo con la primer Behive que creó.

Existen otros patrones no presentes en el R-Pentomino como The orthogonal spaceships en [3].

Capítulo 3

Implementación Game of Life

El programa se implementó en JavaScript que se muestra gráficamente en un canvas sobre un documento HTML, el código fuente está disponible en este enlace: **Ver código y reporte**

Además, al ser un programa orientado web decidí lanzarla alojada gratuitamente por los servidores de Github, por lo que es posible acceder a ella y probar su funcionamiento, Enlace:

Aplicación Web

La rutina principal del código se puede ver abajo, donde es posible notar una pequeña optimización para no copiar todos los estados para la siguiente iteración, el array states de la línea 10 tiene como objetivo almacenar temporalmente únicamente aquellas células que presentan algún cambio en la matriz, lo que hace que la complejidad en memoria pase de $O(w * h)$ a $O(Y)$ y que en promedio llega a ser menor que lineal, siendo Y el número máximo de cambios en una iteración y w y h lo ancho y alto de la cuadrícula.

```
1  routine(evt) {  
2  
3      for(var i = 0; i < this.width; i++) {  
4          for(var j = 0; j < this.height; j++) {  
5  
6              if(this.matrix[i][j] == 1)  
7                  setColor();  
8              else  
9                  setWhiteColor();  
10  
11              context.fillRect(cellSizeBordered*j, cellSizeBordered*i,  
12                  cellSize, cellSize);  
          }  
      }  
  }
```



```

13     }
14
15     var me = this;
16     var waitTime = 50;
17
18     if( this.size >= 500)
19         waitTime = 0;
20     var counter = 0;
21
22
23     var drawGrid = function(me) {
24
25         var states = [];
26         var count_ones = 0;
27         for(var i = 0; i < me.width; i++) {
28             count_ones = 0;
29             for(var j = 0; j < me.height; j++) {
30                 let r_vals = document.getElementById("R_evol").value.
split(",");
31                 if(r_vals.length != 4)
32                     return;
33                 //console.log(r_vals);
34                 if(me.matrix[i][j] == 1) {
35                     count_ones++;
36                     var count = me.countLivingNeighbours(i, j);
37                     if(!(count >= parseInt(r_vals[0]) && count <=
parseInt(r_vals[1]))) {
38                         states.push([i, j, 0]);
39                     }
40                 }
41                 else if(me.matrix[i][j] == 0) {
42                     var count = me.countLivingNeighbours(i, j);
43                     if( count >= parseInt(r_vals[2]) && count <= parseInt
(r_vals[3]) ) {
44                         states.push([i, j, 1]);

```

```

45         }
46     }
47 }
48 // myChart.options.data.push(count_ones);
49 counter++;
50 if(counter % 100 == 0)
51     addData(myChart, counter, count_ones);
52 }
53 for(var i = 0; i < states.length; i++) {
54     me.matrix[states[i][0]][states[i][1]] = states[i][2];
55     if(states[i][2] == 1)
56         setColor();
57     else
58         setWhiteColor();
59     context.fillRect(cellSizeBordered * states[i][1],
60 cellSizeBordered*states[i][0], cellSize, cellSize);
61     context.stroke();
62 }
63 time--;
64 if(time >= 0) {
65     setTimeout(drawGrid, waitTime, me);
66 }
67 };
68 drawGrid(me);
69 }

```

Al abrir el programa inmediatamente aparecen campos para darle una configuración al sistema, como se muestra en la Figura 3.1

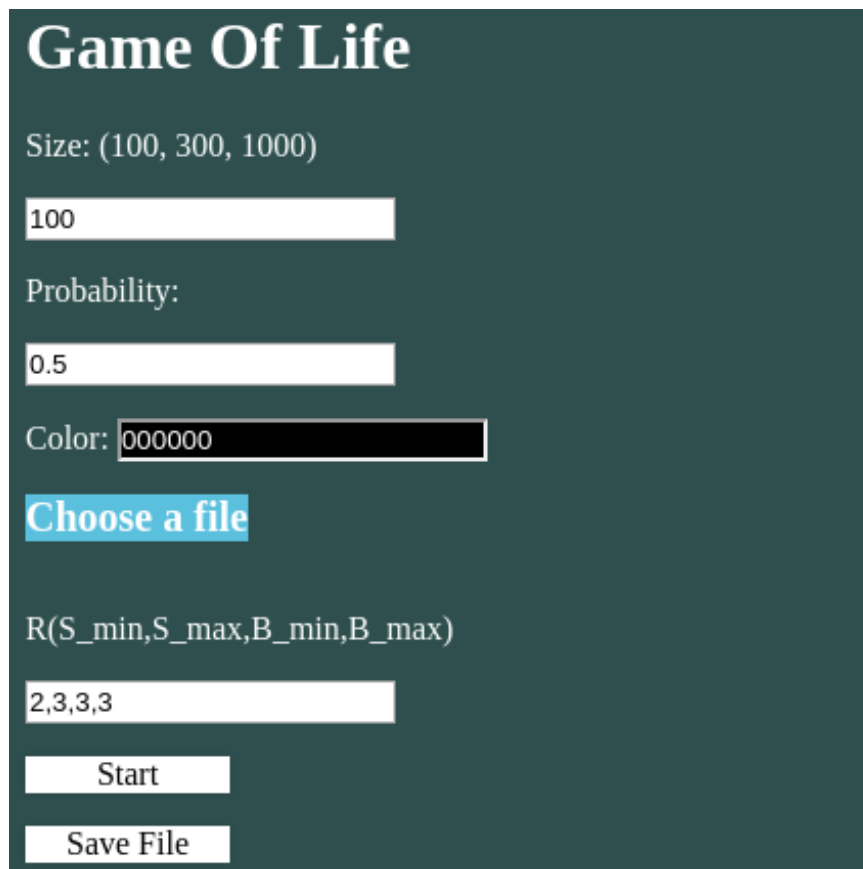


Figura 3.1: Parámetros del programa

- **Size:** Se puede ingresar un tamaño desde 10, a 1000 denotando el tamaño cuadrado de la cuadrícula.
- **Probability:** Es un valor entre 0 y 1 que indica la cantidad de unos en porcentaje iniciales de la cuadrícula.
- **Color:** Al hacer click se mostrará una paleta de colores, con la que se podrá manipular en todo momento el color de los objetos en la cuadrícula.
- **Choose a File:** Al iniciar un programa podemos cargar un archivo de configuración cuya estructura se describe en la siguiente sección.
- **Start:** Una vez elegida una configuración podremos dar click a 'Start' y ver el comportamiento de la cuadrícula.
- **Save File** Podremos descargar un archivo de configuración en todo momento y posteriormente podremos cargarlo en otra instancia.
- **R(S_min,S_max,B_min,B_max):** Permite configurar la regla de evolución, con los valores ingresados separados por coma.

La cuadrícula se muestra en la parte inferior al menú de configuración:



Figura 3.2: Cuadrícula inicial con 50% de células vivas.

La Figura 3.3 muestra una selección de color que se ve reflejada en la cuadrícula.

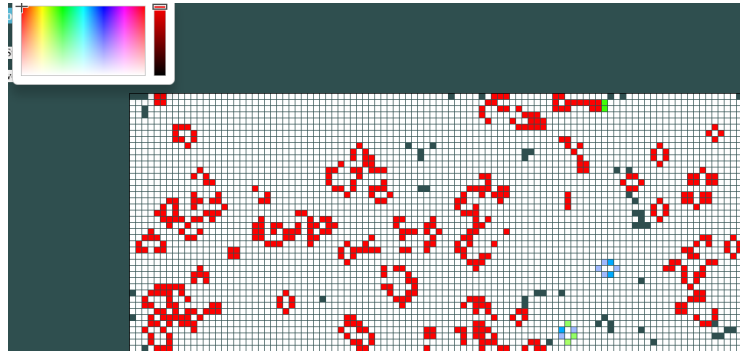


Figura 3.3: Cuadrícula con cambio de color instantáneo

También es posible dibujar nuestra propia configuración inicial con el puntero del ratón mientras se presiona la tecla **shift**.

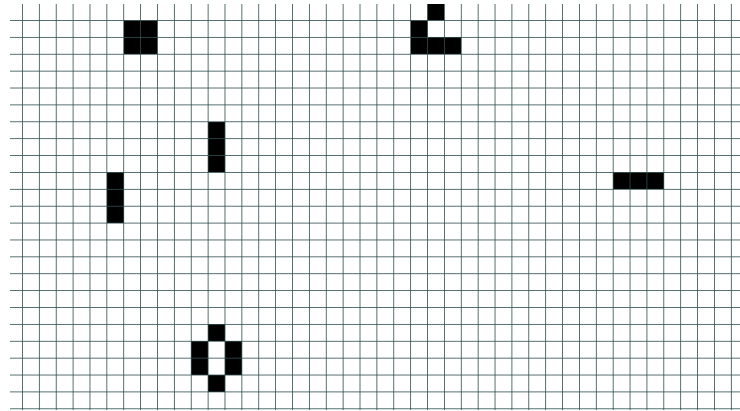


Figura 3.4: Patrones insertados manualmente

Podemos apreciar que hay 3 Oscillator, 1 block, 1 Glider y 1 Beehive en la parte inferior, por lo que al darle iniciar podremos ver su rastro por la cuadrícula.

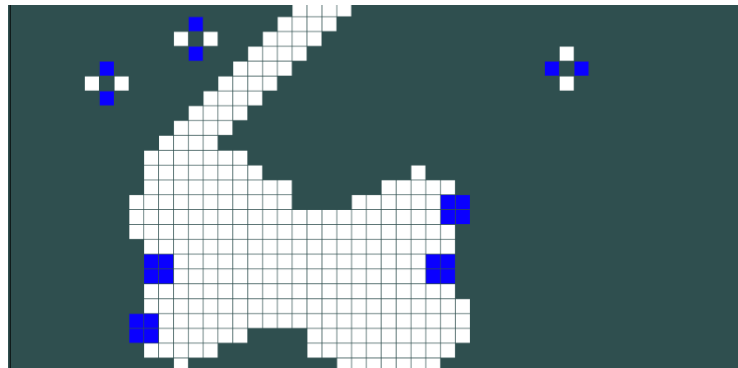


Figura 3.5: Rastro de los objetos, el glider colisiona con un oscillator y forma 4 blocks.

3.1. Archivo de configuración

El archivo de configuración se compone de 4 partes, separadas por un salto de línea, la primera especifica el tamaño de la cuadrícula, siguiendo con el ejemplo anterior, 50x50.

La siguiente línea hace que la configuración del archivo tenga dos posibilidades (0 o 1):

3.1.1. Con una nueva distribución de Células vivas

Si el primer valor es 1, indica que la cuadrícula se va a llenar nuevamente con la proporción especificada en el segundo valor siendo un flotante entre 0 y 1 y omitiendo la lectura de la matriz que representa a la cuadrícula.

3.1.2. Conservando la matriz que representa a la cuadrícula

Si el primer valor es 0, indica que se va a seguir con la matriz existente en el archivo, por lo que omitirá el valor de proporción y leerá la matriz para cargarla en el programa:

1	50
2	0 0
3	100000

La siguiente línea representa las unidades de tiempo discretas que vivirá el programa, para este caso en la línea 3 podemos ver que es de 100000 unidades.

La matriz viene a continuación con valores de 0 y 1, el valor 0 indica una célula muerta y el valor 1, una célula viva, como se ve a continuación representando un segmento del ejemplo anterior donde se aprecia el block, el glider y los 3 oscillators, dos verticales y uno horizontal.

[illegible]

3.2. Gráfica

El número de células vivas por generación se grafica en la parte inferior de la cuadrícula, el eje horizontal representa a las generaciones, mientras que el eje vertical la cantidad de células vivas en cada generación.

A continuación se muestra un ejemplo de 300x300 con un 50% de células vivas:

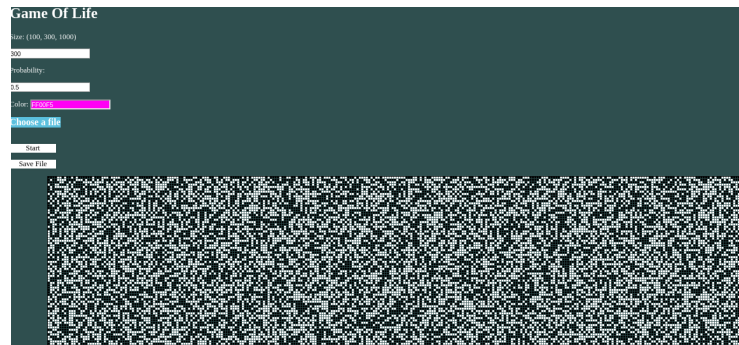


Figura 3.6: Rastro de los objetos, el glider colisiona con un oscillator y forma 4 blocks.

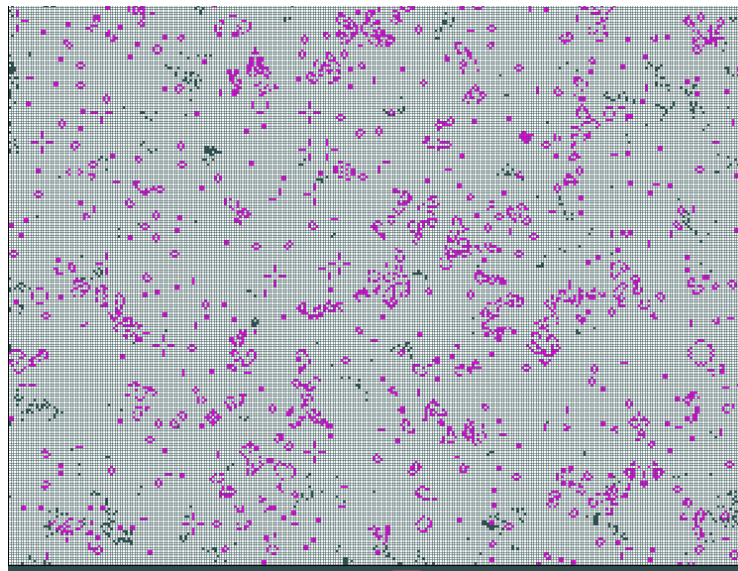


Figura 3.7: Cuadrícula de 300x300.

La gráfica de células vivas muestra un comportamiento repentino decreciente al inicio.

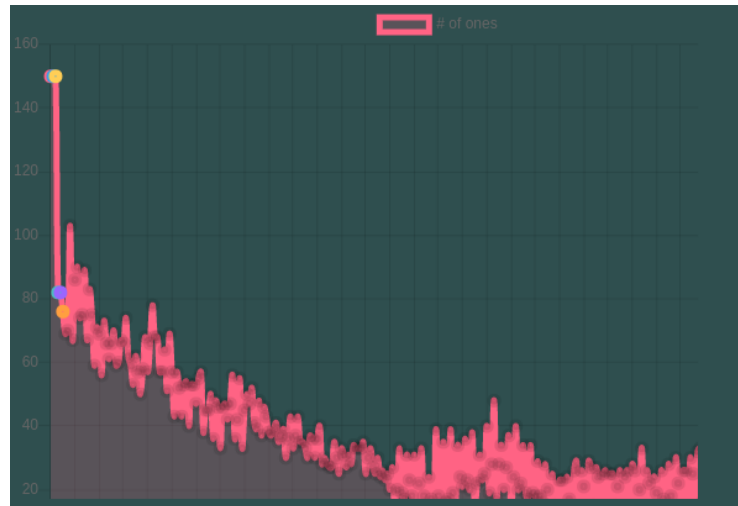


Figura 3.8: Gráfica de cuadrícula de 300x300.

Donde después de cierto tiempo muestra signos de estabilizarse para mostrar una monotonía constante.

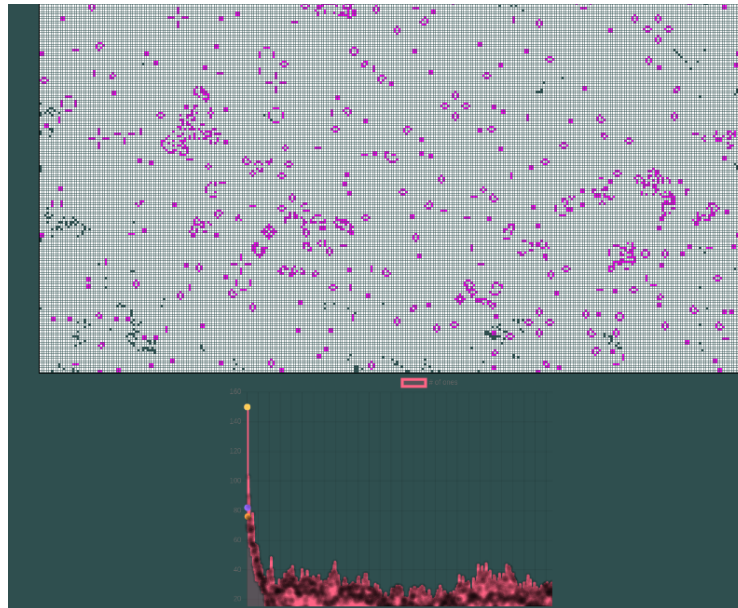


Figura 3.9: Cuadrícula y su gráfica

Al ser de 300x300 el poder lograr ver esa monotonía constante en la gráfica requiere de muchas generaciones o unidades de tiempo, sin embargo en una cuadrícula de 100x100 es más rápido ver este comportamiento como se muestra en la Figura 3.10 y en la Figura 3.11

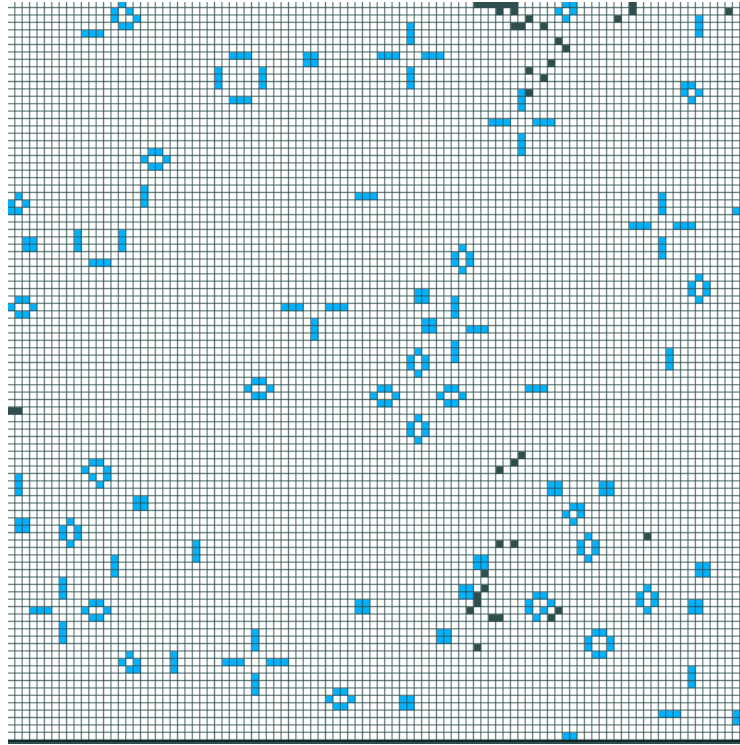


Figura 3.10: Cuadrícula de 100x100 con patrones constantes

Podemos ver como esta tiende a una función constante y que se puede ver debido a que los patrones constan de oscillators y blocks que mantienen el número de células vivas muy constante por periodos.

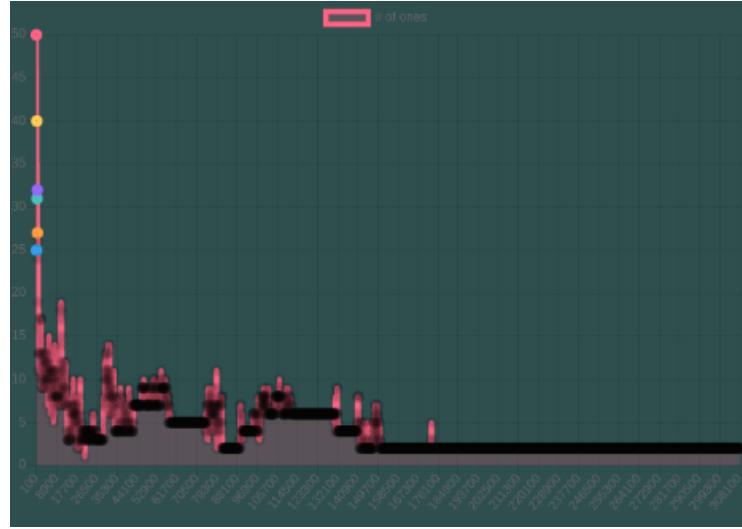


Figura 3.11: Gráfica de la cuadrícula de 100x100

3.3. Regla de evolución

Por defecto el programa carga la regla **R(2, 3, 3, 3)**, que es la propuesta originalmente, sin embargo es posible modificar estos valores y observar el comportamiento del autómata.

Algo interesante es ver el caso en el que $S_{min} = S_{max} = B_{min} = B_{max}$, podría pensarse que esta generalidad da lugar a un resultado similar, sin embargo al hacer pruebas se puede concluir que no es generalizable del todo, ya que si $S_{min} = S_{max} = B_{min} = B_{max} = i$ con $i \in \{1, 2\}$ hay una constancia periódica del número de células vivas, mientras que para $S_{min} = S_{max} = B_{min} = B_{max} \geq 3$ prácticamente todas las células vivas mueren con gran aceleración:

3.3.1. $R(2,2,2,2)$

El comportamiento de la regla cuando los parámetros son igual a 2, tiene también un comportamiento similar cuando son igual a 1, en la Figura 3.12.

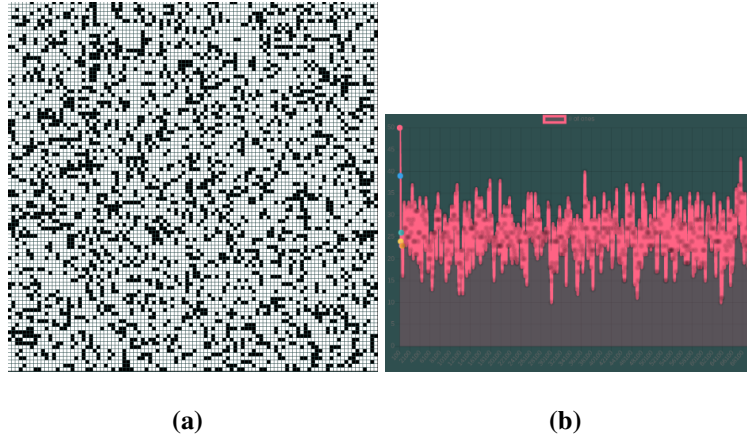


Figura 3.12: (a) Cuadrícula con regla $R(2,2,2,2)$ (b) Gráfica de células vivas.

3.3.2. $R(3,3,3,3)$

Para la regla con valores iguales mayores que 2, el patrón hace que haya una tasa muy alta de células muertas en cada generación, dejando en la mayoría de los casos un tablero con todas las células muertas.

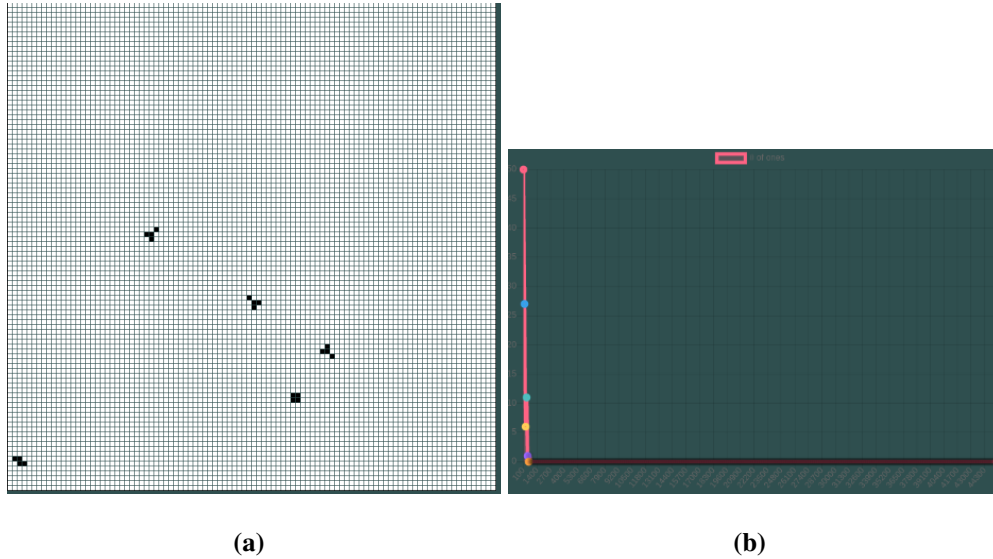


Figura 3.13: (a) Cuadrícula con regla $R(3,3,3,3)$ (b) Gráfica de células vivas.

3.3.3. R(1,2,3,4)

Esta regla produce patrones constantes en cuanto a periodicidad, lo que genera gráficas relativamente constantes en cuanto a periodo.

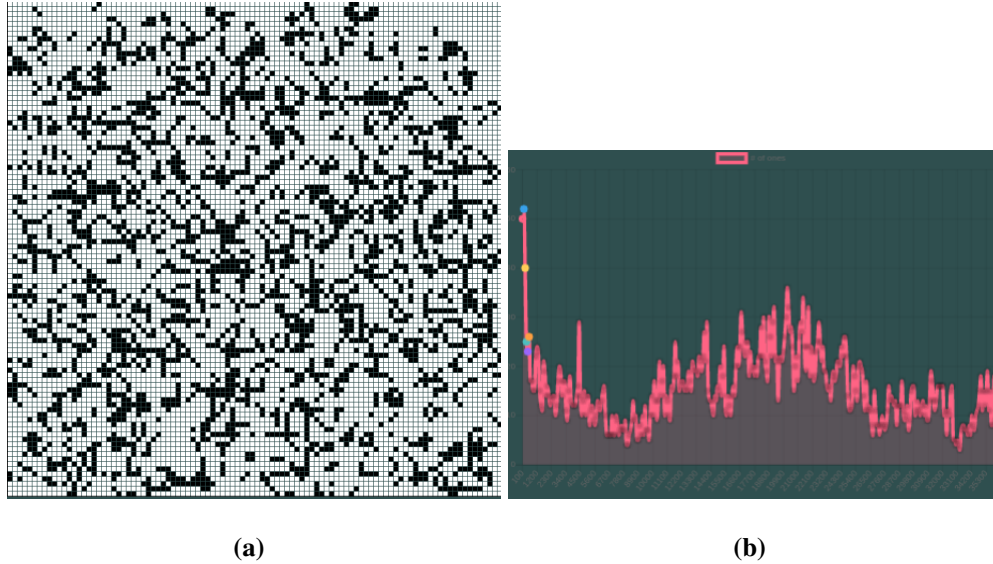


Figura 3.14: (a) Cuadrícula con regla R(1,2,3,4) (b) Gráfica de células vivas.

3.3.4. $R(2,4,6,8)$

En el caso de esta regla, después de pocas generaciones se puede ver una constancia en cada una de las generaciones siguientes, produciendo en su mayoría oscillators de periodo simple y patrones tipo Still Life que se ve reflejada en la gráfica.

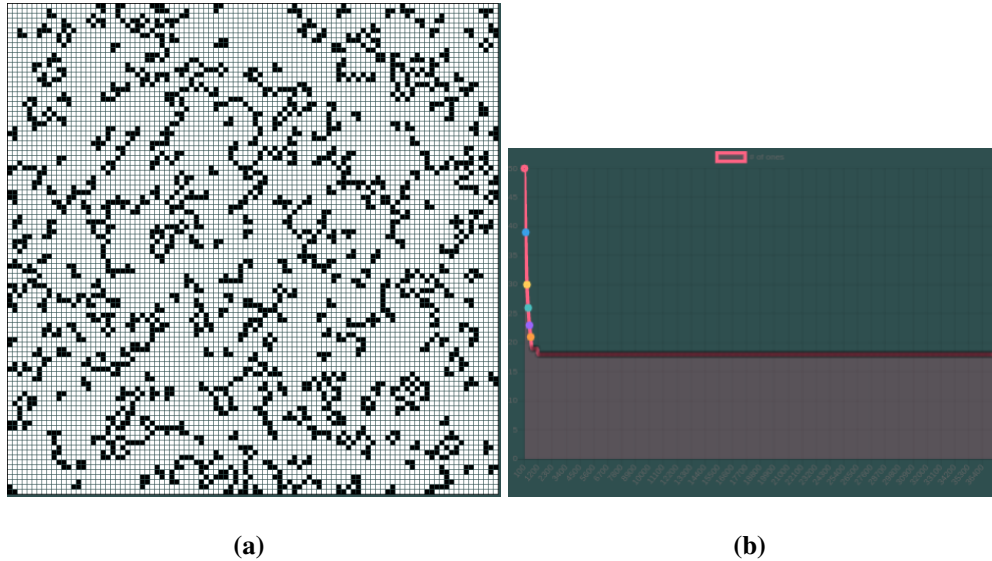


Figura 3.15: (a) Cuadrícula con regla $R(2,4,6,8)$ (b) Gráfica de células vivas.

3.3.5. $R(2,3,3,4)$

Esta regla de evolución aunque muy parecida a la $R(12,3,4)$, parece que agrega volumen a los patrones, produciendo un buen efecto visual, también se debe mencionar que mantiene en términos numéricos una cantidad de células vivas y muertas aunque variable parece garantizar que el autómata conservará su densidad impidiendo que los objetos Still Life duren por mucho tiempo.

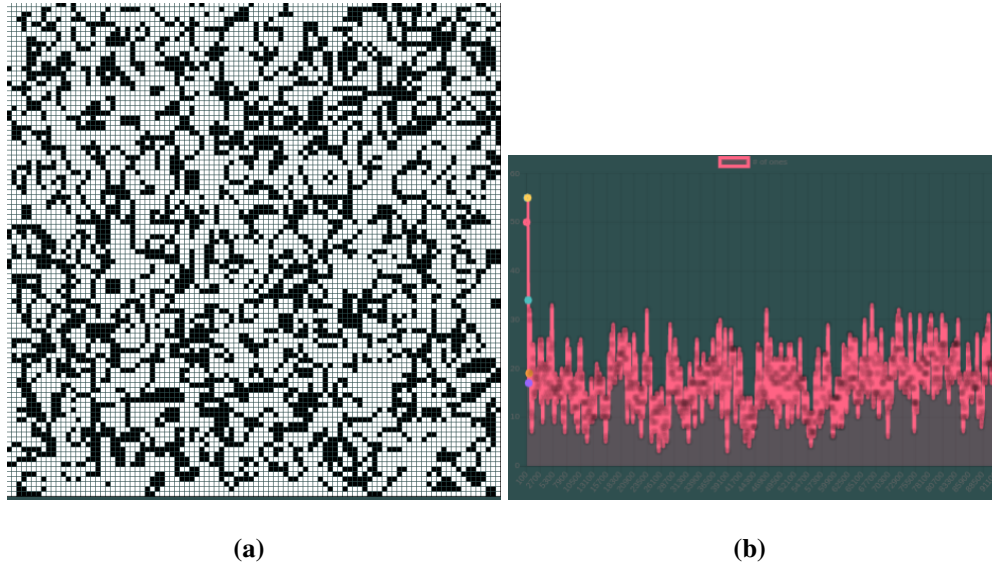


Figura 3.16: (a) Cuadrícula con regla $R(2,3,3,4)$ (b) Gráfica de células vivas.

El programa cumple con las características propuestas que se enlistan a continuación:

No.	Característica
1	Evaluar espacios de 300x300, 500x500 y máximo de 1000x1000 células
2	Animación en dos dimensiones
3	Poder cambiar los colores del estado
4	Poder inicializar el espacio de evoluciones con diferentes densidades
5	Poder editar el espacio de evoluciones para dibujar configuraciones particulares
6	Poder salvar y levantar archivos con configuraciones en específico
7	Graficar el número de unos de cada generación
8	Editar la regla de evolución $R(S_{min}, S_{max}, B_{min}, B_{max})$

Características del programa del juego de la vida.

Bibliografía

- [1] C. Zandonella, “Mathematician john horton conway, a ‘magical genius’ known for inventing the ‘game of life,’ dies at age 82.” <https://www.princeton.edu/news/2020/04/14/mathematician-john-horton-conway-magical-genius-known-inventing-game-life-dies-age-82>, 2020. 1
- [2] M. Gymrek, “Conway’s game of life.” <http://web.mit.edu/sp.268/www/2010/lifeSlides.pdf>, 2010. 3
- [3] P. Callahan, “What is the game of life?.” <http://www.math.com/students/wonders/life/life.html>. 3