

Reporte ETS Especial

Alumno: Monroy Martos Elioth

Boleta: 2016630258

Profesor: Genaro Juárez Martínez

Materia: Computing Selected Topics

Grupo: 3CM7

25 de enero de 2019

Índice

1. Máquina de Turing	4
1.1. Introducción	4
1.2. Planteamiento de la práctica	5
1.3. Desarrollo	6
1.4. Pruebas y resultados	9
1.5. Conclusiones	12
2. Autómata Celular (Juego de la Vida y Regla de Difusión)	13
2.1. Introducción	13
2.1.1. Juego de la Vida	13
2.2. Planteamiento de la práctica	14
2.3. Desarrollo	16
2.4. Pruebas y resultados	27
2.4.1. Análisis de poblaciones	36
2.5. Conclusiones	56
3. Hormiga de Langton	57
3.1. Introducción	57
3.2. Planteamiento de la práctica	57
3.2.1. Hormiga de Langton Original	57
3.2.2. Hormiga de Langton Modificada	58
3.3. Desarrollo	59
3.3.1. Hormiga de Langton Original	59
3.3.2. Hormiga de Langton Modificada	71
3.4. Pruebas y resultados	89
3.4.1. Hormiga de Langton Original	89
3.5. Hormiga de Langton Modificada	92
3.6. Conclusiones	96
4. Grafos	97
4.1. Introducción	97
4.2. Planteamiento de la práctica	97
4.3. Desarrollo	98
4.4. Pruebas y resultados	101
4.5. Conclusiones	103

5. Autómata Celular con Memoria	104
5.1. Introducción	104
5.2. Planteamiento de la práctica	104
5.3. Desarrollo	106
5.4. Pruebas y resultados	120
5.4.1. Regla: 2 7 4 6. Densidad: 20%	120
5.4.2. Regla: 3 6 3 4 (Cruz en el centro)	125
5.4.3. Regla: 1 6 1 6. Densidad: 10%	132
5.4.4. Regla: 3 3 1 8. Densidad: 10%	138
5.4.5. Regla: 3 3 1 7. Línea en el centro de 7 cuadros	145
5.5. Conclusiones	153
Referencias	154

1. Máquina de Turing

1.1. Introducción

Las máquinas de Turing son un dispositivo que permite idealmente resolver cualquier problema, debido a que tiene una memoria infinita, sin embargo está implementación no es posible de realizar y deben ser acotadas a una memoria o cinta finita, de tal forma que se dice que si una máquina de Turing es capaz de resolver un problema, este problema es computable (quiere decir que una computadora lo puede resolver). En caso contrario, se dice que el problema no es computable.[1]

De manera formal la máquina de Turing se define de la siguiente manera:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

donde:

Q es el conjunto finito de estados de la unidad de control.

Σ es el conjunto finito de símbolos de entrada

Γ es el conjunto completo de símbolos de cinta; Σ siempre es un subconjunto de Γ

δ Es la función de transición. Los argumentos de $\delta(q, X)$ son un estado q y un símbolo de cinta X . El valor de $\delta(q, X)$, si está definido, es (p, Y, D) donde:

1. p es el siguiente estado de Q
2. Y es el símbolo de Γ , que se escribe en la casilla que señala la cabeza y que sustituye a cualquier símbolo que se encontrara en ella.
3. D es una dirección y puede ser L o R , lo que nos indica la dirección en que la cabeza se mueve, ‘izquierda’ (L) o ‘derecha’ (R), respectivamente

q_0 es el estado inicial, un elemento de Q , en el que inicialmente se encuentra la unidad de control.

B es el símbolo espacio en blanco. Este símbolo pertenece a Γ pero no a Σ ; es decir, no es un símbolo de entrada.

F es el conjunto de los estados finales, un subconjunto de Q

1.2. Planteamiento de la práctica

El ejercicio a realizar en esta actividad, es diseñar e implementar una máquina de Turing que duplique la cantidad de unos de una cadena ingresada, y que además, se muestre el procedimiento en una animación y se escriba en un archivo el historial de los pasos realizados por la máquina de Turing.

Para la elaboración del programa, se realizó el siguiente diagrama que modela los estados y las transiciones que tiene la máquina.

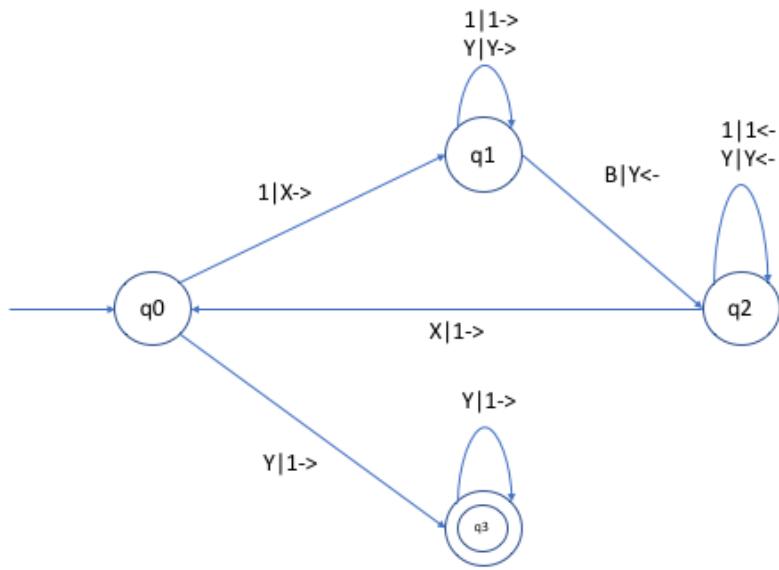


Figura 1: Estados y transiciones usados

1.3. Desarrollo

Para el modelado de la máquina de Turing se realizó la siguiente implementación en Python 3.7, la cual es una clase que sirve para trabajar con la misma:

maquina.py:

```
1 class MaquinaTuring(object):
2     def __init__(self, estados, alfabeto, transiciones, inicial,
3                  final, cadena):
4         self.estados=estados
5         self.alfabeto=alfabeto
6         self.transiciones=transiciones
7         self.inicial=inicial
8         self.estado_actual=self.inicial
9         self.final=final
10        self.apuntador=0
11        self.blanco="B"
12        self.cadena=list(cadena)
13        self.direccion=""
14    def evaluar(self):
15        if len(self.cadena)-1<self.apuntador:
16            caracter=self.blanco
17        else:
18            caracter=self.cadena[self.apuntador]
19
20        if caracter in self.alfabeto:
21            transicion=(self.estado_actual, caracter)
22            if transicion in self.transiciones:
23                siguiente=self.transiciones[transicion]
24                if len(self.cadena)-1<self.apuntador:
25                    self.cadena.append(self.blanco)
26                if self.apuntador<0:
27                    self.cadena.insert(0,self.blanco)
28                self.cadena[self.apuntador]=siguiente[1]
29                if siguiente[2]=="R":
30                    self.apuntador+=1
31                else:
32                    self.apuntador=self.apuntador-1
33
34            self.estado_actual=siguiente[0]
35            self.direccion=siguiente[2]
36            if self.estado_actual in self.estados:
37                return True
38            else:
```

```

38         return False
39     else:
40         return False
41     else:
42         return False
43 def esFinal(self):
44     if self.estado_actual in self.final:
45         if len(self.cadena)-1<self.apuntador:
46             return True
47     return False

```

Por si sola está máquina no tiene funcionalidad, por lo cual fue necesario el uso de la misma en otro archivo, el cual se encarga de manejar todo el funcionamiento del programa, como recibir la entrada (cadena de unos), y utilizar tkinter para realizar la animación de la máquina de Turing.

main.py:

```

1 from tkinter import *
2 from tkinter import font as tkfont
3 import time
4 from maquina import MaquinaTuring
5
6 entrada = input("Ingrese la cadena a duplicar: ")
7 estados=[ "q0" , "q1" , "q2" , "q3" ]
8 alfabeto=[ "1" , "Y" , "X" , "B" ]
9 transiciones={
10     ("q0" , "1") : ("q1" , "X" , "R"),
11     ("q0" , "Y") : ("q3" , "1" , "R"),
12     ("q1" , "1") : ("q1" , "1" , "R"),
13     ("q1" , "Y") : ("q1" , "Y" , "R"),
14     ("q1" , "B") : ("q2" , "Y" , "L"),
15     ("q2" , "1") : ("q2" , "1" , "L"),
16     ("q2" , "X") : ("q0" , "1" , "R"),
17     ("q2" , "Y") : ("q2" , "Y" , "L"),
18     ("q3" , "Y") : ("q3" , "1" , "R"),
19 }
20
21 maquina = MaquinaTuring(estados , alfabeto , transiciones , estados
22 [0] , estados[3] , entrada)
23
24 master = Tk()
25 master.title("Turing")
26 c = Canvas(master , width=400, height=300)
27 c.pack()
28 tipo_letra = tkfont.Font(family="Helvetica" , size=20)

```

```

29 flecha = c.create_line(175, 150, 175, 175, arrow=LAST, width=2)
30 texto = c.create_text(165, 200, text="".join(maquina.cadena),
31     font=tipo_letra, anchor=W)
32 estado = c.create_text(120, 150, text=maquina.estado_actual,
33     font=tipo_letra, anchor=W)
34 archivo = open("salida.txt", "w")
35 while not maquina.esFinal():
36     print("Cadena: "+"".join(maquina.cadena))
37     print("Estado actual: "+str(maquina.estado_actual)+",
38         apuntador: "+str(maquina.apuntador)+"\n")
39     archivo.write("Cadena: "+"".join(maquina.cadena)+"\n")
40     archivo.write("Estado actual: "+str(maquina.estado_actual)+",
41         apuntador: "+str(maquina.apuntador)+"\n")
42     if not maquina.evaluar():
43         print("_____")
44         archivo.write("_____")
45         archivo.write("\n")
46     print("Siguiente estado: "+str(maquina.estado_actual))
47     print("_____")
48     archivo.write("Siguiente estado: "+str(maquina.estado_actual))
49     archivo.write("\n")
50     archivo.write("_____")
51     master.update()
52     time.sleep(.8)
53     c.itemconfigure(texto, text="".join(maquina.cadena))
54     c.itemconfigure(estado, text=str(maquina.estado_actual))
55     if maquina.direccion == 'R':
56         c.move(flecha, 10, 0)
57     else:
58         c.move(flecha, -10, 0)
59
60
61 print("Resultado: "+"".join(maquina.cadena))
62 archivo.write("Resultado: "+"".join(maquina.cadena)+"\n")
63 archivo.flush()
64 master.mainloop()

```

Cabe señalar que el código que se muestra anteriormente, fue cargado mediante el paquete de LaTex “listings”.

1.4. Pruebas y resultados

Para comprobar la funcionalidad del programa, se realizó la siguiente prueba, se ingresó la cadena “11”.

Los resultados obtenidos fueron los siguientes:

En la figura 2 se muestra la salida en consola del programa anteriormente ejecutado, en consola se despliega el log de los pasos que va haciendo la máquina así como la cadena que se encuentra actualmente en la cinta, el estado actual y el apuntador.

```
[MacBook-Pro-de-Elioth:MT eliothmonroy$ python3 main.py
Ingrese la cadena a duplicar: 11
Cadena: 11
Estado actual: q0, apuntador: 1

Siguiente estado: q1
-----
Cadena: X1
Estado actual: q1, apuntador: 2

Siguiente estado: q1
-----
Cadena: X1
Estado actual: q1, apuntador: 3

Siguiente estado: q2
-----
Cadena: X1Y
Estado actual: q2, apuntador: 2

Siguiente estado: q2
-----
Cadena: X1Y
```

Figura 2: Resultados prueba (1)

En la figura 3 se muestra un archivo de log que es generado por el programa al terminar su ejecución, el cual muestra todos los pasos que realizó la máquina, tal y como se mostraron en consola.

```
Cadena: 11
Estado actual: q0, apuntador: 1
Siguiente estado: q1-----
Cadena: X1
Estado actual: q1, apuntador: 2
Siguiente estado: q1-----
Cadena: X1
Estado actual: q1, apuntador: 3
Siguiente estado: q2-----
Cadena: X1Y
Estado actual: q2, apuntador: 2
Siguiente estado: q2-----
Cadena: X1Y
Estado actual: q2, apuntador: 1
Siguiente estado: q0-----
Cadena: 11Y
Estado actual: q0, apuntador: 2
Siguiente estado: q1-----
Cadena: 1XY
Estado actual: q1, apuntador: 3
Siguiente estado: q1-----
Cadena: 1XY
Estado actual: q1, apuntador: 4
Siguiente estado: q2-----
Cadena: 1XXY
Estado actual: q2, apuntador: 3
Siguiente estado: q2-----
Cadena: 1XXY
Estado actual: q2, apuntador: 2
Siguiente estado: q0-----
Cadena: 1YY
Estado actual: q0, apuntador: 3
Siguiente estado: q3-----
Cadena: 11Y
Estado actual: q3, apuntador: 4
Siguiente estado: q3-----
Resultado: 1111
```

Figura 3: Resultados prueba (2)

En las figuras 4 y 5 se muestra la ventana del programa en la que se realiza la animación de la máquina de Turing.

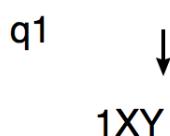


Figura 4: Resultados prueba (3)

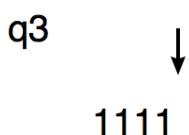
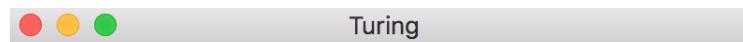


Figura 5: Resultados prueba (4)

1.5. Conclusiones

El uso de autómatas finitos deterministas y no deterministas nos permite validar que el orden en cadenas de entrada sea correcto. Esto tiene diversas aplicaciones, desde muy sencillas (como validar el formato de un CURP) hasta aplicaciones más complejas como realizar parte del análisis léxico de un lenguaje de entrada en un compilador.

Los cuales nos ayudan a saber si una entrada pertenece a nuestro lenguaje o no, siendo de gran utilidad al momento de validar cadenas apartir de una expresión regular.

2. Autómata Celular (Juego de la Vida y Regla de Difusión)

2.1. Introducción

Los autómatas celulares(AC) surgen en la década de 1940 con John Von Neumann, que intentaba modelar una máquina que fuera capaz de auto-rePLICARSE, llegando así a un modelo matemático de dicha maquina con reglas complicadas sobre una red rectangular. Inicialmente fueron interpretados como conjunto de células que crecían, se reproducían y morían a medida que pasaba el tiempo. A esta similitud con el crecimiento de las células se le debe su nombre.[2]

Un autómata celular se caracteriza por contar con los siguientes elementos:

- Arreglo regular. Ya sea un plano de dos dimensiones o un espacio n-dimensional, este es el espacio de evoluciones, y cada división homogénea del arreglo es llamada célula.
- Conjunto de estados. Es finito y cada elemento o célula del arreglo toma un valor de este conjunto de estados. También se denomina alfabeto. Puede ser expresado en valores o colores.
- configuración inicial. Consiste en asignar un estado a cada una de las células del espacio de evolución inicial del sistema.
- Vecindades. Define el conjunto contiguo de células y posición relativa respecto a cada una de ellas. A cada vecindad diferente corresponde un elemento del conjunto de estados.
- Función local. Es la regla de evolución que determina el comportamiento del A. C. Se conforma de una célula central y sus vecindades. Define como debe cambiar de estado cada célula dependiendo de los estados anteriores de sus vecindades. Puede ser una expresión algebraica o un grupo de ecuaciones.

2.1.1. Juego de la Vida

El juego de la vida fue desarrollado por John Horton Conway, quien fuera un matemático estadounidense que trabajaba en la Universidad de Cambrid-

ge. Él desarrollo un “juego” al cual llamaba vida, debido a su parecido con la forma en que las sociedades de organismos vivos se levantan y caen.[3]

Este juego se considera como un simulador, ya que se asemeja a la vida real. Originalmente, se planteó como un juego de mesa, pero con el pasar de los años fue usado en otras ramas (como la computación) debido a las posibilidades que este juego brinda.

La idea básica del juego, es iniciar con una configuración simple de organismos vivientes, cada uno asignado a una celda dentro de un tablero (el cual se considera un plano infinito), para así observar como está cambia según se aplican las leyes genéticas de Conway, las cuales determinan el nacimiento, muerte o supervivencia de cada organismo. Estás reglas son tres:

1. Supervivencia: Cada organismo con dos o tres vecinos vivos sobrevivirá a la siguiente generación.
2. Muerte: Cada organismo con cuatro o más vecinos muere por sobrepopulation, así mismo cada organismo con solo un vecino o ninguno morirá por aislamiento.
3. Nacimiento: En cada celda vacía que este rodeada por exactamente tres vecinos, nacerá un organismo. [3]

Es importante señalar que cada muerte, nacimiento o supervivencia debe ser simultaneo durante cada salto de generación. Para realizar la evaluación de cada una de las celdas, estás son divididas a su vez en grupos de 9 celdas, la célula que se evaluará constituye el centro del ahora cuadrado. Dentro de este cuadrado, son aplicadas las reglas ya descritas anteriormente.

El programa que ha sido desarrollado para está actividad, simula este juego. Son dados como parámetros el total de la población y la probabilidad de que existan organismos vivos en la misma. Y con base a esto se realizada la simulación del juego de la vida aplicando las reglas de Conway. Además se gráfica el histórico de la cantidad de organismos vivos que han existido durante cada una de las generaciones.

2.2. Planteamiento de la práctica

Este programa implementa la simulación de un autómata celular mediante el uso de las reglas conocidas como Life y Difusión. A su vez, el programa debe tener una interfaz gráfica donde se muestren las células vivas y muertas

mediante el uso de una cuadrilla. Las características de este simulador son las siguientes:

- Permitir seleccionar el tamaño de la población de la matriz.
- Permitir seleccionar la regla que se utilizara en cada iteración de la simulación.
- Se podrá elegir la distribución de células que habrá en la matriz.
- Se podrá cambiar los colores de las células vivas y muertas.
- Se mostrará el cambio de unos que hay a lo largo de cada iteración.
- Se graficará a lo largo del tiempo la cantidad de células vivas.
- Cuando el usuario de click a una de las células está deberá cambiarse al estado contrario al que se encuentra en ese momento (de viva a muerta y viceversa).

El objetivo que se tiene es mostrar una matriz de hasta 1000 por 1000 para poder observar un comportamiento que nos proporcione información.

2.3. Desarrollo

El desarrollo de este programa, se llevo acabo usando Javascript, debido a que el entorno web y del navegador permiten que se puedan desarrollar interfaces para usuario de una forma bastante eficiente. Además, los programas que requieren de usar animaciones o que hacen un uso extensivo de dibujar en un canvas (como es el caso), se ven beneficiados del entorno web que permite todo se ejecute más rápido.

La parte de presentación para el usuario, se encuentra en el siguiente archivo html.

Archivo: gol.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Game of life</title>
6     <!-- <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous"> -->
7     <link rel="stylesheet" type="text/css" href="estilos.css">
8   </head>
9   <body>
10    <div id="contenedor">
11      <!-- <canvas id="myCanvas" width="1260" height="680" style="border:1px solid #000000;"> -->
12      <div>
13        <canvas onclick="clickCanvas(event)" id="myCanvas" width="680" height="680" style="border:1px solid #000000;">
14        </canvas>
15      </div>
16      <div id="controles">
17        <div id="botones">
18          <!-- <div> -->
19          <button onclick="iniciar()">Iniciar</button>
20          <button onclick="continuar()">Continuar</button>
21          <button onclick="pausa()">Pausar</button>
22          <button onclick="siguiente()">Siguiente</button>
23          <!-- </div> -->
24        </div>
25      <div>
```

```

26      <button id="descargar" onclick="descargar()">Descargar
27  log</button>
28  </div><br>
29  <div>
30      <label id="generacion">Generación: 0</label>
31  </div><br>
32  <div>
33      <label for="tam">Tamaño población:</label>
34      <input type="number" id="tam" value="100"><br>
35  </div><br>
36  <div>
37      <label for="b1">B1:</label>
38      <input type="number" id="b1" value="2"><br>
39      <label for="b2">B2:</label>
40      <input type="number" id="b2" value="3"><br>
41      <label for="s1">S1:</label>
42      <input type="number" id="s1" value="3"><br>
43      <label for="s2">S2:</label>
44      <input type="number" id="s2" value="3"><br>
45  </div><br>
46  <div>
47      <label for="distribucion">
48          Distribución de vivos:
49          <span id="distribucion_valor">50</span>%
50      </label>
51      <input type="range" min="0" max="100" value="50" id="distribucion">
52  </div><br>
53  <div>
54      <label for="color_vivo">Color de vivos:</label>
55      <input type="color" value="#66ff66" id="color_vivo">
56  <br>
57      <label for="color_muerto">Color de muertos:</label>
58      <input type="color" value="#FF0000" id="color_muerto">
59  <br>
60  </div>
61  </div>
62  <canvas id="myChart" width="500" height="500"></canvas><br>
63  <div>Matriz auxiliar:</div>
64  <script src="Chart.min.js"></script>
65  <script type="text/javascript" src="gol.js"></script>
66  </body>
67 </html>

```

La lógica con la que funciona el programa, se encuentra en el siguiente

archivo de Javascript.

Archivo: gol.js

```
1 var c=document.getElementById("myCanvas");
2 var chart = document.getElementById("myChart");
3 var longitud=680;
4 var tam=100;
5 var escala=longitud/tam;
6 var ctx=c.getContext("2d");
7 ctx.lineWidth = "0.1";
8 var random;
9 var x,y;
10 var regla1=2;
11 var regla2=3;
12 var regla3=3;
13 var regla4=3;
14 var generacion=0;
15 var numero_vivos=0;
16 var numero_muertos=0;
17 var color_vivo="#66ff66";
18 var color_muerto="#FF0000";
19 var vecindad=new Array(8);
20 var intervalo;
21 var log;
22 var generacion_grafica=[];
23 var vivos_grafica=[];
24 var total_vivos=0;
25 var array;
26 var array2;
27 var coordenadas;
28 var distribucion=50;
29
30 //Función para descargar
31 function descargar(){
32     var fileContents = log;
33     var fileName = "log.txt";
34     var pp = document.createElement('a');
35     pp.setAttribute('href', 'data:text/plain;charset=utf-8,' +
36         encodeURIComponent(fileContents));
37     pp.setAttribute('download', fileName);
38     pp.click();
39 }
40 //Modificar valor del slider en pantalla
```

```

41 document.getElementById("distribucion_valor").innerHTML =
42     document.getElementById("distribucion").value;
43 document.getElementById("distribucion").oninput = function(e) {
44     e.preventDefault();
45     document.getElementById("distribucion_valor").innerHTML = this
46         .value;
47 }
48
49 function Create2DArray(rows) {
50     var arr = new Array(rows);
51     for (var i=0;i<rows; i++) {
52         arr[i] = new Array(rows);
53     }
54     return arr;
55 }
56
57 function evaluar(i,j){
58     var estado=array2[i][j];
59     var cantidad_vivos=0;
60     var cantidad_muertos=0;
61     if (i==0){
62         vecindad[0]=array2[tam-1][j];//superior
63         vecindad[1]=array2[i+1][j];//inferior
64         if (j==0){
65             vecindad[2]=array2[tam-1][tam-1];//superior izquierdo
66             vecindad[3]=array2[tam-1][j+1];//superior derecho
67             vecindad[4]=array2[i][tam-1];//izquierdo
68             vecindad[5]=array2[i][j+1];//derecho
69             vecindad[6]=array2[i+1][tam-1];//inferior izquierdo
70             vecindad[7]=array2[i+1][j+1];//inferior derecho
71         }
72         else if (j==tam-1){
73             vecindad[2]=array2[tam-1][j-1];//superior izquierdo
74             vecindad[3]=array2[tam-1][0];//superior derecho
75             vecindad[4]=array2[i][j-1];//izquierdo
76             vecindad[5]=array2[i][0];//derecho
77             vecindad[6]=array2[i+1][j-1];//inferior izquierdo
78             vecindad[7]=array2[i+1][0];//inferior derecho
79     }
80     else{
81         vecindad[2]=array2[tam-1][j-1];//superior izquierdo
82         vecindad[3]=array2[tam-1][j+1];//superior derecho
83         vecindad[4]=array2[i][j-1];//izquierdo
84         vecindad[5]=array2[i][j+1];//derecho
85         vecindad[6]=array2[i+1][j-1];//inferior izquierdo

```

```

84         vecindad[7]=array2[i+1][j+1];//inferior derecho
85     }
86 }
87 else if (i==tam-1){
88     vecindad[0]=array2[i-1][j];//superior
89     vecindad[1]=array2[0][j];//inferior
90     if (j==0){
91         vecindad[2]=array2[i-1][tam-1];//superior izquierdo
92         vecindad[3]=array2[i-1][j+1];//superior derecho
93         vecindad[4]=array2[i][tam-1];//izquierdo
94         vecindad[5]=array2[i][j+1];//derecho
95         vecindad[6]=array2[0][tam-1];//inferior izquierdo
96         vecindad[7]=array2[0][j+1];//inferior derecho
97     }
98     else if (j==tam-1){
99         vecindad[2]=array2[i-1][j-1];//superior izquierdo
100        vecindad[3]=array2[i-1][0];//superior derecho
101        vecindad[4]=array2[i][j-1];//izquierdo
102        vecindad[5]=array2[i][0];//derecho
103        vecindad[6]=array2[0][j-1];//inferior izquierdo
104        vecindad[7]=array2[0][0];//inferior derecho
105    }
106    else{
107        vecindad[2]=array2[i-1][j-1];//superior izquierdo
108        vecindad[3]=array2[i-1][j+1];//superior derecho
109        vecindad[4]=array2[i][j-1];//izquierdo
110        vecindad[5]=array2[i][j+1];//derecho
111        vecindad[6]=array2[0][j-1];//inferior izquierdo
112        vecindad[7]=array2[0][j+1];//inferior derecho
113    }
114 }
115 else{
116     vecindad[0]=array2[i-1][j];//superior
117     vecindad[1]=array2[i+1][j];//inferior
118     if (j==0){
119         vecindad[2]=array2[i-1][tam-1];//superior izquierdo
120         vecindad[3]=array2[i-1][j+1];//superior derecho
121         vecindad[4]=array2[i][tam-1];//izquierdo
122         vecindad[5]=array2[i][j+1];//derecho
123         vecindad[6]=array2[i+1][tam-1];//inferior izquierdo
124         vecindad[7]=array2[i+1][j+1];//inferior derecho
125     }
126     else if (j==tam-1){
127         vecindad[2]=array2[i-1][j-1];//superior izquierdo
128         vecindad[3]=array2[i-1][0];//superior derecho

```

```

129     vecindad[4]=array2[i][j-1];//izquierdo
130     vecindad[5]=array2[i][0];//derecho
131     vecindad[6]=array2[i+1][j-1];//inferior izquierdo
132     vecindad[7]=array2[i+1][0];//inferior derecho
133 }
134 else{
135     vecindad[2]=array2[i-1][j-1];//superior izquierdo
136     vecindad[3]=array2[i-1][j+1];//superior derecho
137     vecindad[4]=array2[i][j-1];//izquierdo
138     vecindad[5]=array2[i][j+1];//derecho
139     vecindad[6]=array2[i+1][j-1];//inferior izquierdo
140     vecindad[7]=array2[i+1][j+1];//inferior derecho
141 }
142 }
143 for(var k=0;k<8;k++){
144     if(vecindad[k]==1){
145         cantidad_vivos++;
146     }else{
147         cantidad_muertos++;
148     }
149 }
150 if(estado==1){
151     if(!((cantidad_vivos >= regla1) && (cantidad_vivos <=
152         regla2))){
153         array[i][j]=0;
154         numero_vivos--;
155         numero_muertos++;
156         ctx.fillStyle=color_muerto;
157         ctx.fillRect(0+(j*(escala)),0+(i*(escala)),escala,escala);
158     }
159 }else{
160     if((cantidad_vivos >= regla3) && (cantidad_vivos <= regla4)){
161         array[i][j]=1;
162         numero_vivos++;
163         numero_muertos--;
164         ctx.fillStyle=color_vivo;
165         ctx.fillRect(0+(j*(escala)),0+(i*(escala)),escala,escala);
166     }
167     ctx.stroke();
168 }
169
170 function copiar(array){
171     var aux=Create2DArray(tam);

```

```

172     for ( var i=0; i<tam; i++){
173         for ( var j=0;j<tam; j++){
174             aux [ i ] [ j]=array [ i ] [ j ];
175         }
176     }
177     return aux;
178 }
179
180 function obtenerTamano(){
181     tam = parseInt(document.getElementById("tam").value);
182     //console.log(tam);
183     array=Create2DArray(tam);
184     array2=Create2DArray(tam);
185     coordenadas=Create2DArray(tam);
186     escala=longitud/tam;
187 }
188
189 function obtenerRegla(){
190     regla1 = parseInt(document.getElementById("b1").value);
191     regla2 = parseInt(document.getElementById("b2").value);
192     regla3 = parseInt(document.getElementById("s1").value);
193     regla4 = parseInt(document.getElementById("s2").value);
194 }
195
196 function obtenerDistribucion(){
197     distribucion = parseInt(document.getElementById("distribucion")
198         ).value)/100;
199     distribucion = parseFloat(distribucion.toFixed(2));
200 }
201
202 function obtenerColores(){
203     color_vivo=document.getElementById("color_vivo").value;
204     color_muerto=document.getElementById("color_muerto").value;
205 }
206
207 function obtenerConfiguracion(){
208     obtenerTamano();
209     obtenerRegla();
210     obtenerDistribucion();
211     obtenerColores();
212 }
213
214 function limpiar(){
215     ctx.clearRect(0, 0, ctx.width, ctx.height);
216     console.clear();

```

```

216 new Chart(chart,{});  

217 numero_vivos=0;  

218 numero_muertos=0;  

219 generacion=0;  

220 document.getElementById('generacion').innerHTML = "Generación:  

221     0";  

222 generacion_grafica=[];  

223 vivos_grafica=[];  

224 log="";  

225 }  

226 function iniciar(){  

227     limpiar();  

228     obtenerConfiguracion();  

229     for(var i=0; i<tam; i++){  

230         for(var j=0;j<tam; j++){  

231             //Checamos la distribución  

232             if(Math.random() < distribucion){  

233                 array[i][j]=1;  

234                 array2[i][j]=1;  

235                 ctx.fillStyle=color_vivo;  

236                 numero_vivos++;  

237             }  

238             else{  

239                 array[i][j]=0;  

240                 array2[i][j]=0;  

241                 ctx.fillStyle=color_muerto;  

242                 numero_muertos++;  

243             }  

244             x=0+j*escala;  

245             y=0+i*escala;  

246             coordenadas[i][j]=x+", "+y+", "+(x+escala)+", "+(y+escala);  

247             ctx.fillRect(0+(j*(escala)),0+(i*(escala)), escala, escala);  

248             ctx.stroke();  

249         }  

250     }  

251     generacion_grafica.push(generacion);  

252     vivos_grafica.push(numero_vivos);  

253     console.log(numero_vivos+", "+generacion);  

254     log=numero_vivos+", "+generacion+"\n";  

255 }  

256  

257 function ejecutar(){  

258     obtenerRegla();  

259     obtenerColores();  


```

```

260   for ( var i=0; i<tam; i++){
261     for ( var j=0;j<tam; j++){
262       evaluar(i ,j );
263     }
264   }
265   array2=copiar (array );
266   generacion++;
267   generacion_grafica .push(generacion );
268   vivos_grafica .push(numero_vivos );
269   total_vivos+=numero_vivos ;
270   console .log (numero_vivos+ " "+generacion );
271   document .getElementById( 'generacion' ).innerHTML =" Generación :
272   "+generacion ;
273   log+=numero_vivos+ " "+generacion+"\n" ;
274 }
275 function continuar (){
276   intervalo=setInterval(ejecutar , 0 );
277   intervalo2=setInterval(graficar , 1000 );
278 }
279
280 function pausa (){
281   clearInterval(intervalo );
282   clearInterval(intervalo2 );
283 }
284 function siguiente (){
285   ejecutar();
286   graficar();
287 }
288
289 function graficar (){
290   new Chart(chart , {
291     type: 'line' ,
292     data: {
293       labels: generacion_grafica ,
294       datasets: [{
295         data: vivos_grafica ,
296         label: "Generación: "+generacion+ " Número de vivos: "
297         + numero_vivos+ " Promedio: "+(total_vivos/generacion)+"
298         Densidad: "+(total_vivos/generacion)/(tam*tam) ,
299         borderColor: "red" ,
300         fill: false
301       }]
302     },
303     options: {

```

```

302     title: {
303         display: true ,
304         text: 'Gráfica de vivos'
305     },
306     elements: {
307         line: {
308             tension: 0, // disables bezier curves
309         }
310     },
311     animation: {
312         duration: 0, // general animation time
313     },
314     hover: {
315         animationDuration: 0, // duration of animations when
316         hovering an item
317     },
318     responsiveAnimationDuration: 0, // animation duration
319     after a resize
320     events: [] , //Para que no se pueda interactuar con la
321     gráfica
322     maintainAspectRatio:true ,
323     responsive: true ,
324     }
325 );
326 }
327
328 function obtenerCoordenadas(i ,j){
329     var texto=coordenadas[i ][j ];
330     return texto .split(",");
331 }
332
333 function clickCanvas(event){
334     x=event .offsetX;
335     y=event .offsetY;
336     // console.log(x + "," +y);
337     for( var i=0;i<tam ;i++){
338         for(var j=0;j<tam ;j++){
339             var arr=obtenerCoordenadas(i ,j );
340             if(x >= arr [0] && x <= arr [2] && y >= arr [1] && y <= arr
341             [3]){
342                 x=0+j *escala ;
343                 y=0+i *escala ;
344                 if(array [i ][j ]==0){
345                     array [i ][j ]=1;
346                     array2 [i ][j ]=1;

```

```

343     numero_vivos++;
344     numero_muertos--;
345     ctx.fillStyle=color_vivo;
346 }else{
347     array[i][j]=0;
348     array2[i][j]=0;
349     numero_vivos--;
350     numero_muertos++;
351     ctx.fillStyle=color_muerto;
352 }
353 ctx.fillRect(x,y,escala,escala);
354 ctx.stroke();
355 //console.log("Soy: i:"+i+" j:"+j);
356 console.log("Número vivos: "+numero_vivos);
357 break;
358 }
359 }
360 }
361 }
362
363 //iniciar();

```

2.4. Pruebas y resultados



Figura 6: Interfaz gráfica del programa

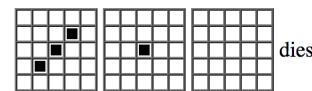


Figura 7: Prueba 1

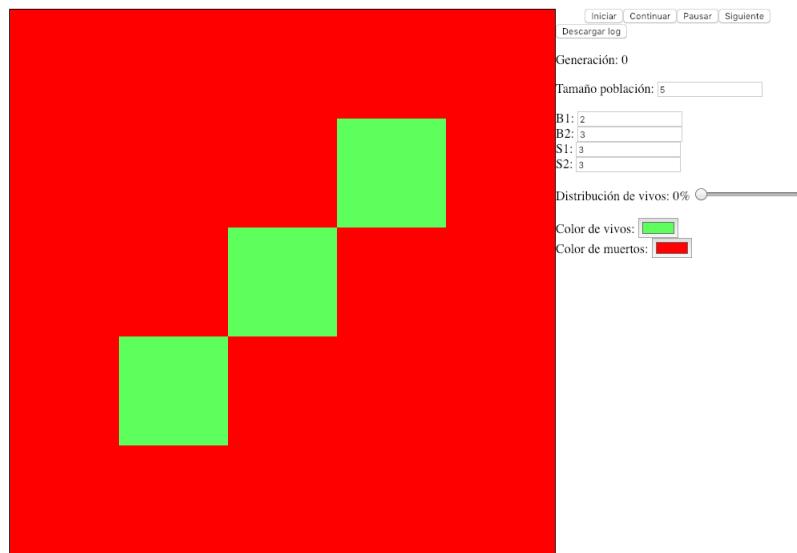


Figura 8: Resultado para la prueba 1 (1)

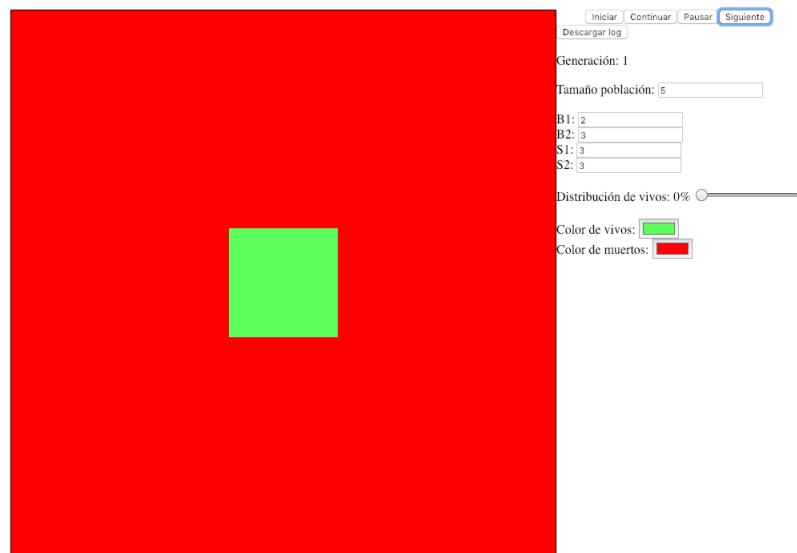


Figura 9: Resultado para la prueba 1 (2)



Figura 10: Resultado para la prueba 1 (3)



Figura 11: Prueba 2

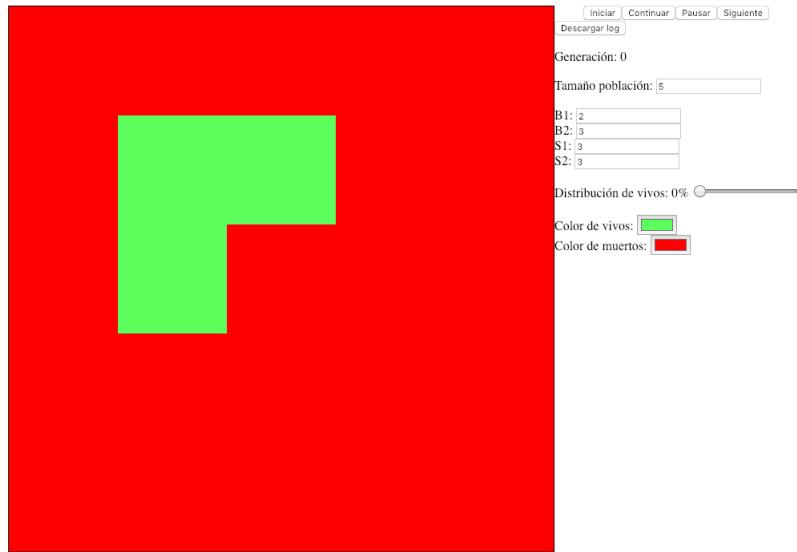


Figura 12: Resultado para la prueba 2 (1)

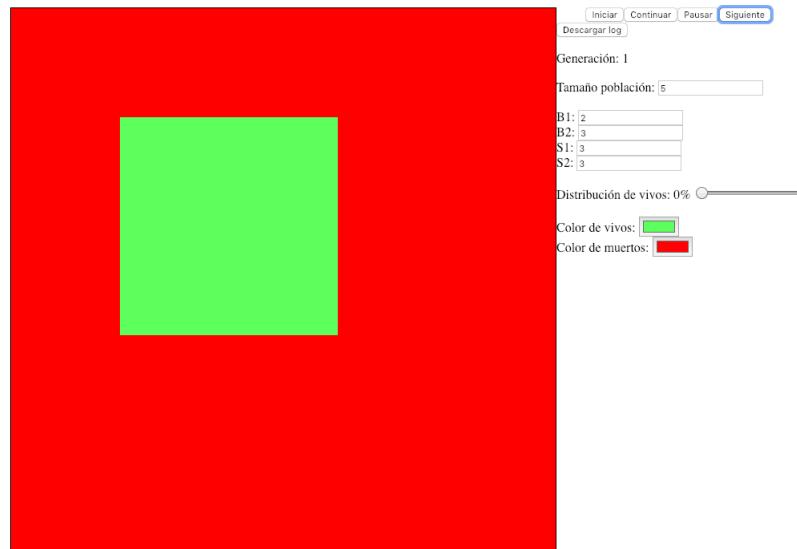


Figura 13: Resultado para la prueba 2 (2)

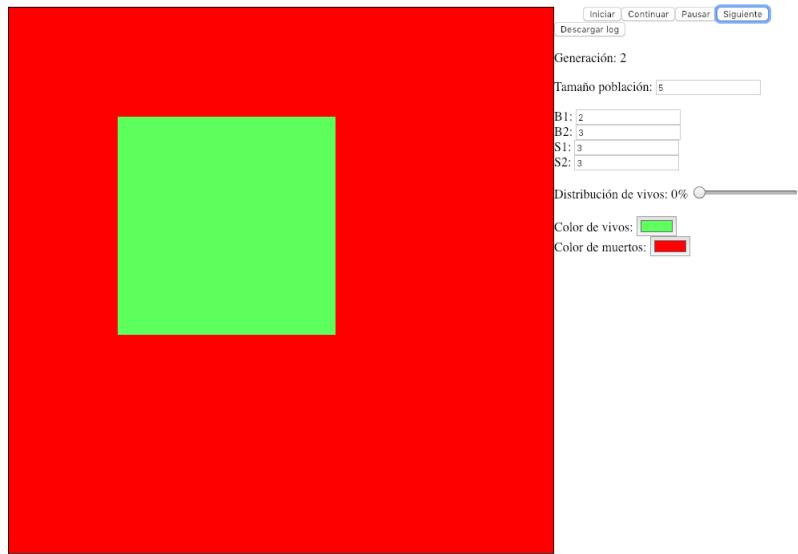


Figura 14: Resultado para la prueba 2 (3)

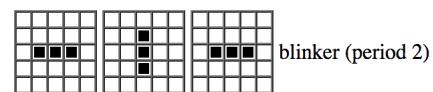


Figura 15: Prueba 3



Figura 16: Resultado para la prueba 3 (1)

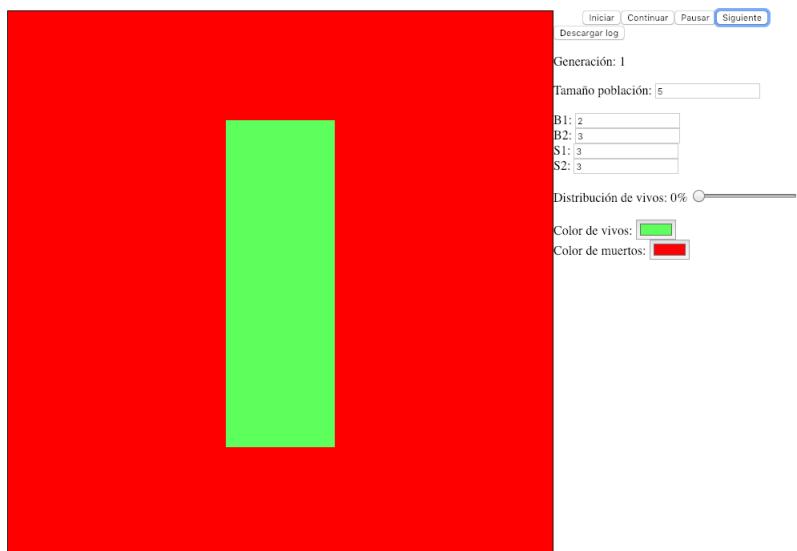


Figura 17: Resultado para la prueba 3 (2)



Figura 18: Resultado para la prueba 3 (3)

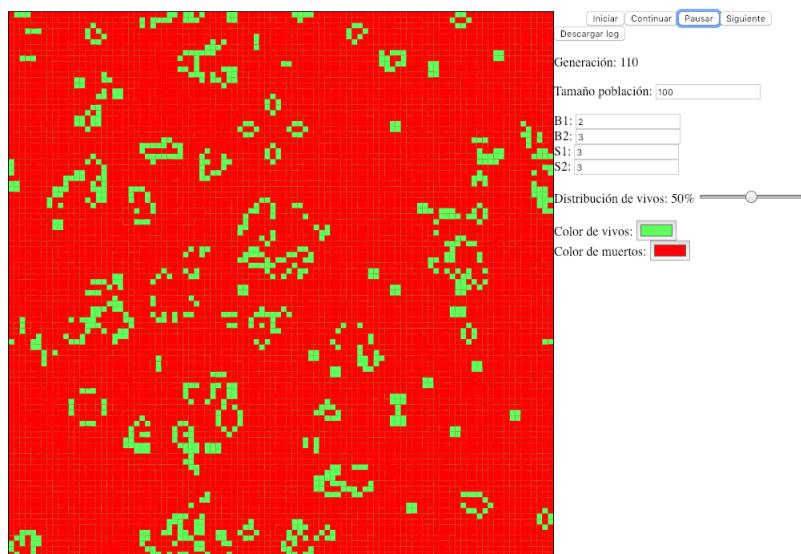


Figura 19: Juego de la vida después de 100 generaciones

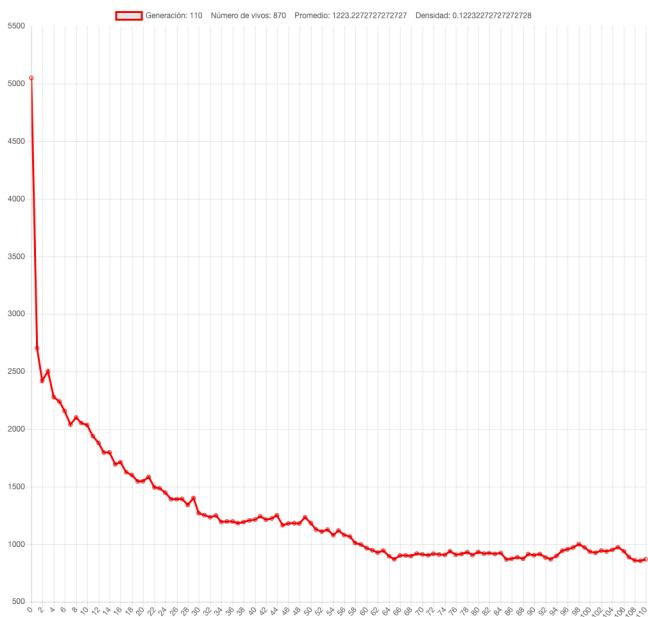


Figura 20: Total células vivas en juego de la vida después de 100 generaciones

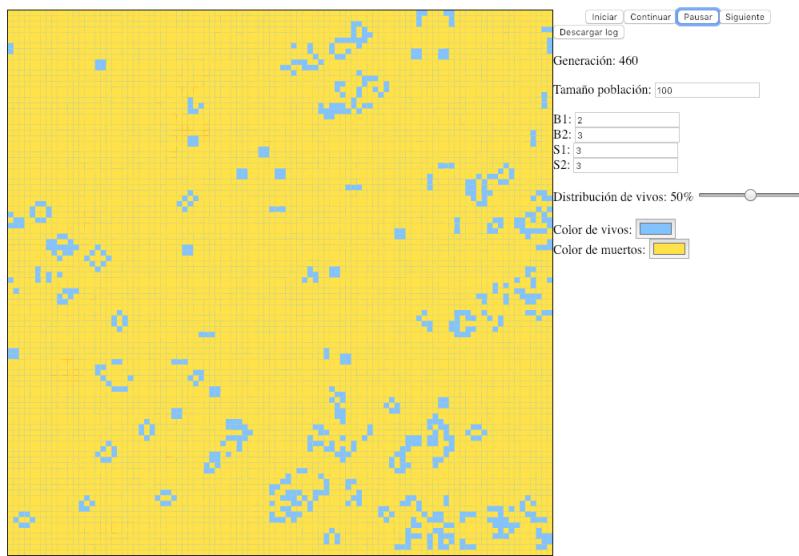


Figura 21: Juego de la vida con distintos colores

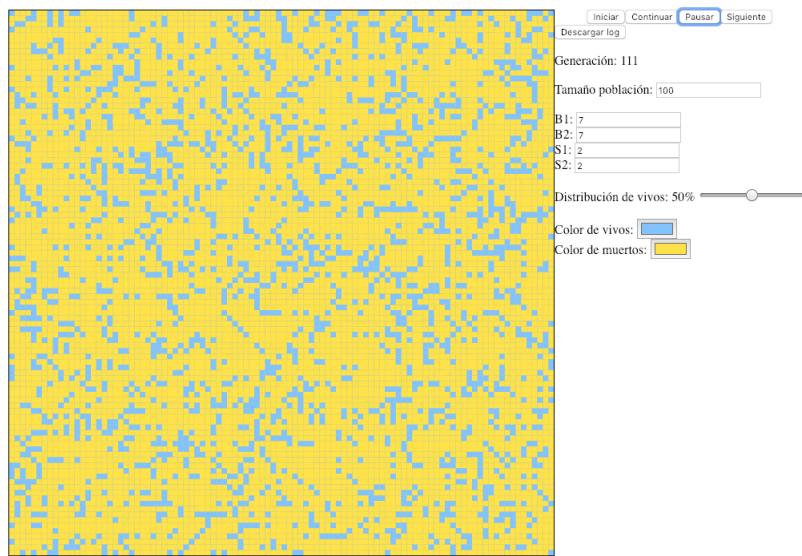


Figura 22: Regla de difusión después de 100 generaciones

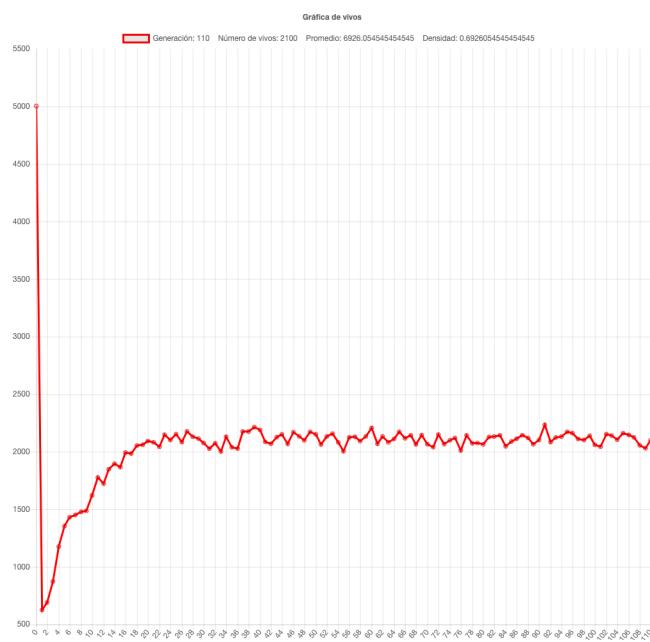


Figura 23: Total células vivas con regla difusión después de 100 generaciones

2.4.1. Análisis de poblaciones

En esta parte se hicieron pruebas con la regla de life y difusión aumentando la densidad de la población de 10 en 10 por ciento hasta llegar al máximo de 90 por ciento debido que en un 100 por ciento no se aprecia nada al igual que en cero, las pruebas se realizaron tras 1000 generaciones en una matriz de 100 por 100.

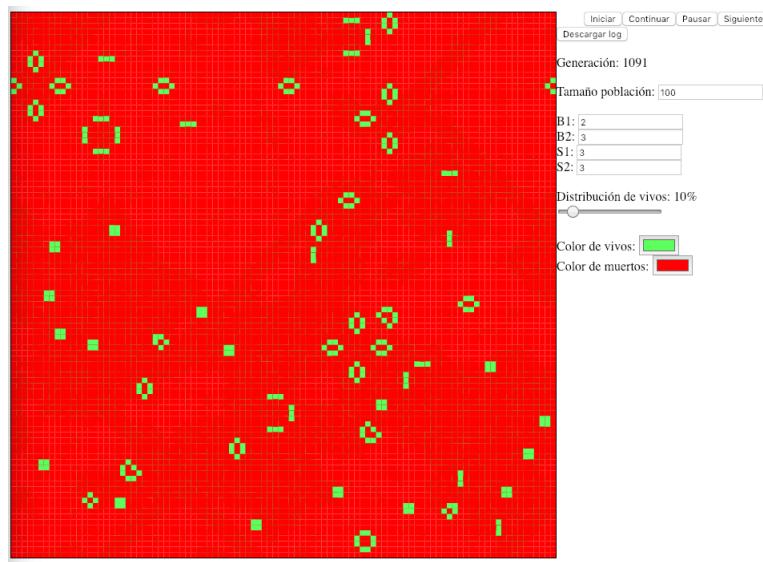


Figura 24: Regla de life con probabilidad de 10 %



Figura 25: Comportamiento de la población de la simulación anterior

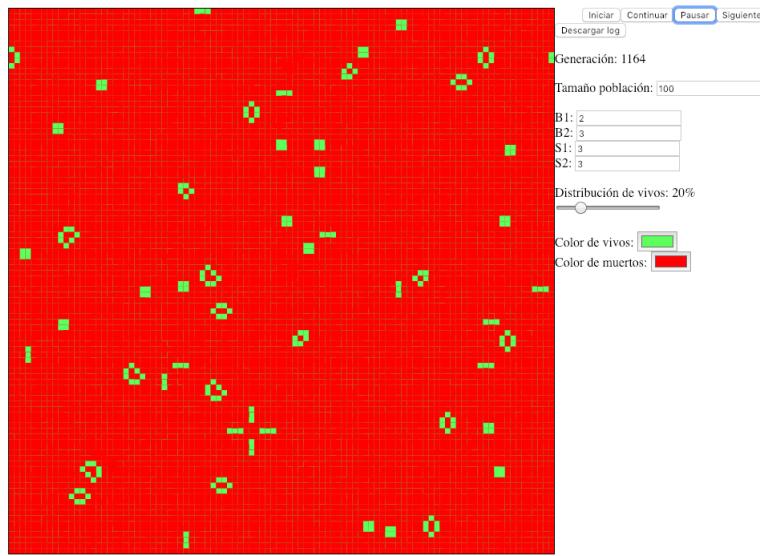


Figura 26: Regla de life con probabilidad de 20 %

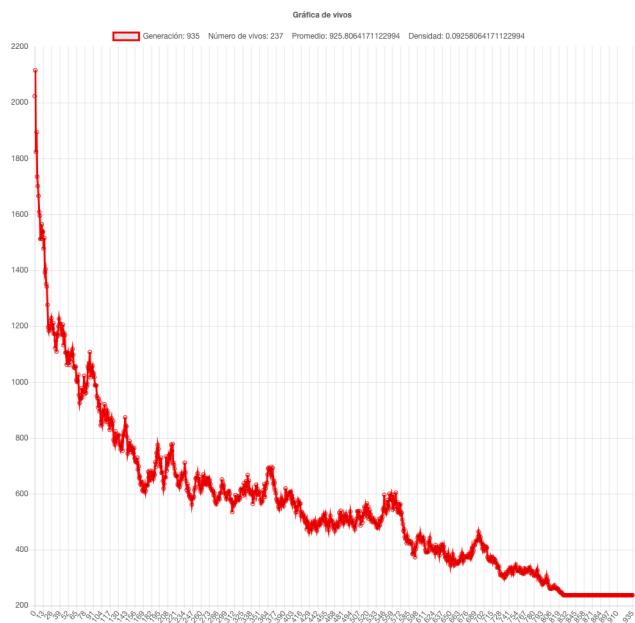


Figura 27: Comportamiento de la población de la simulación anterior

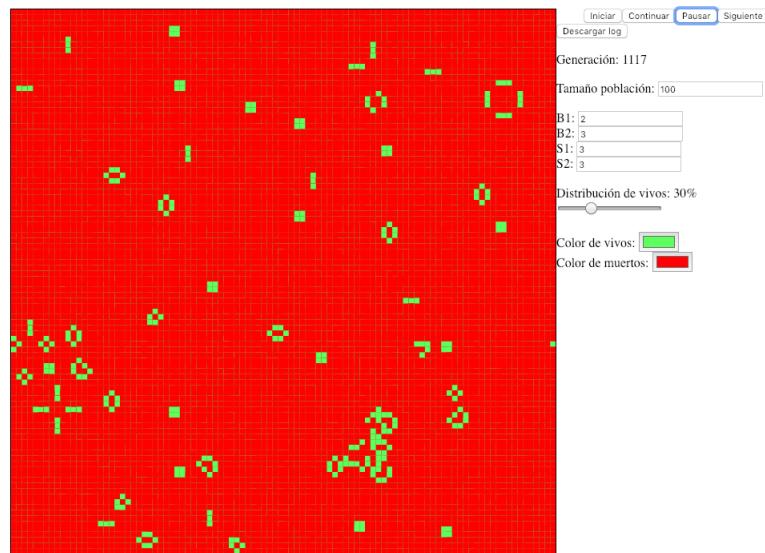


Figura 28: Regla de life con probabilidad de 30 %

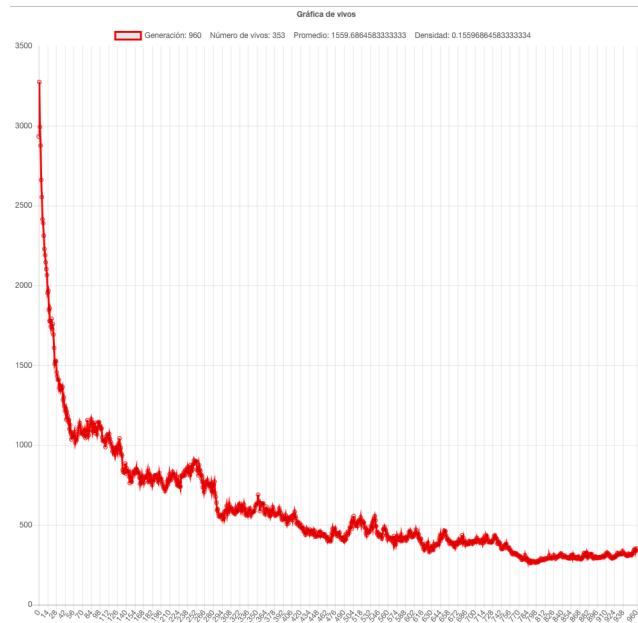


Figura 29: Comportamiento de la población de la simulación anterior

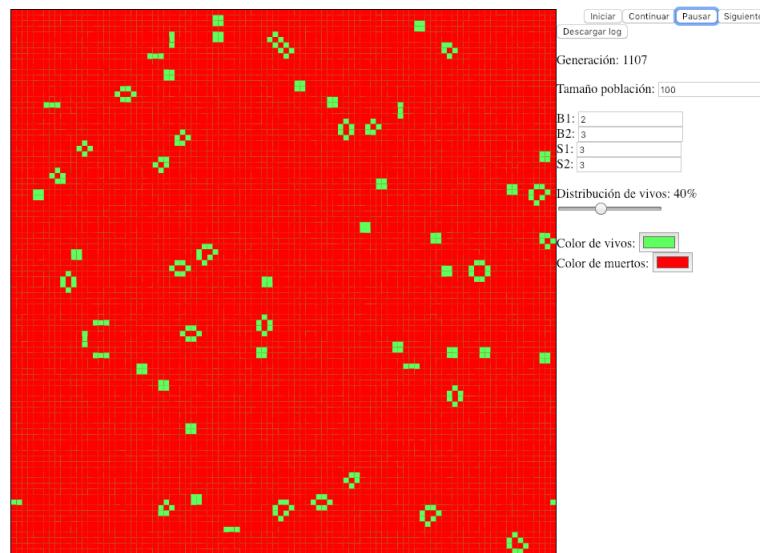


Figura 30: Regla de life con probabilidad de 40 %



Figura 31: Comportamiento de la población de la simulación anterior

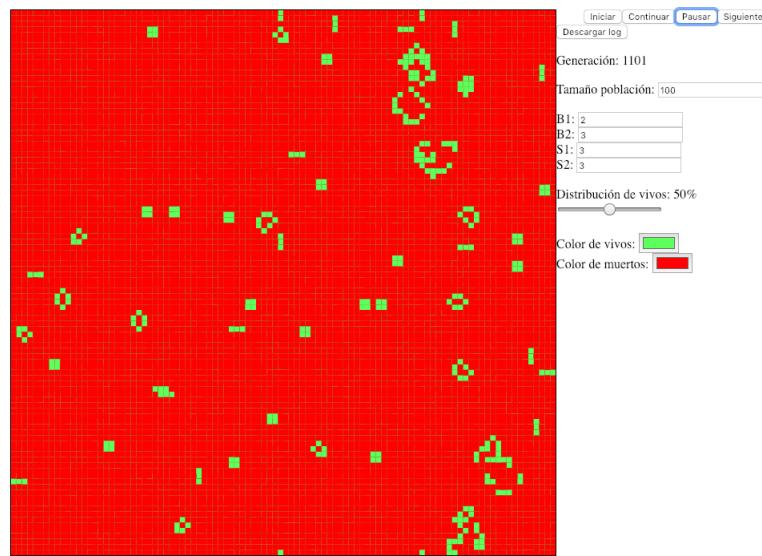


Figura 32: Regla de life con probabilidad de 50 %



Figura 33: Comportamiento de la población de la simulación anterior

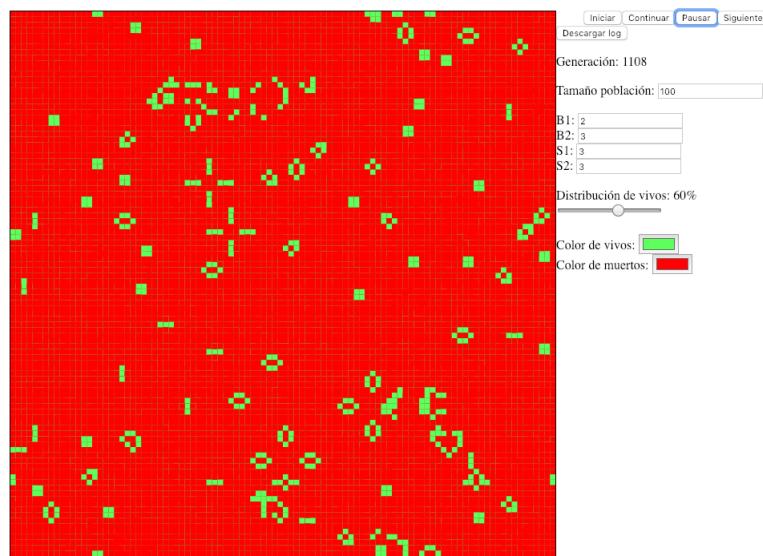


Figura 34: Regla de life con probabilidad de 60 %



Figura 35: Comportamiento de la población de la simulación anterior

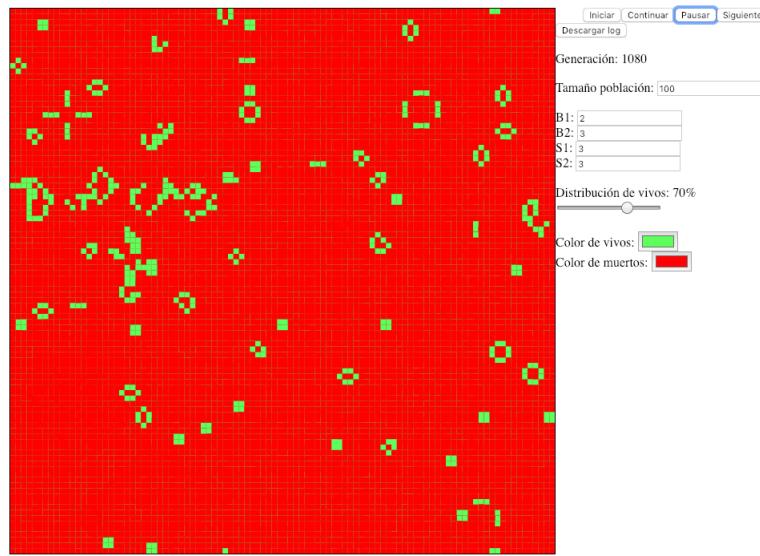


Figura 36: Regla de life con probabilidad de 70 %

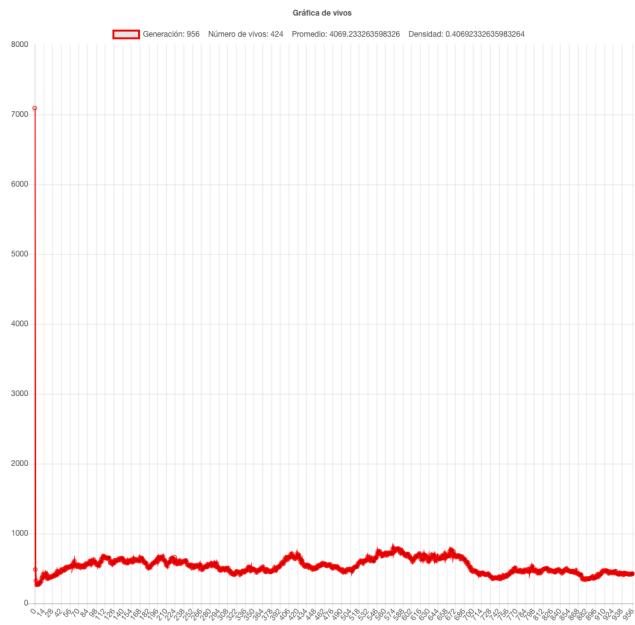


Figura 37: Comportamiento de la población de la simulación anterior

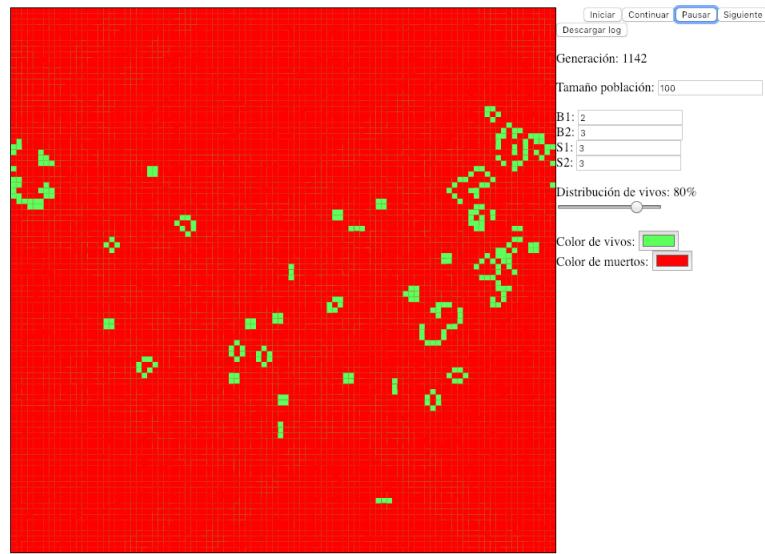


Figura 38: Regla de life con probabilidad de 80 %

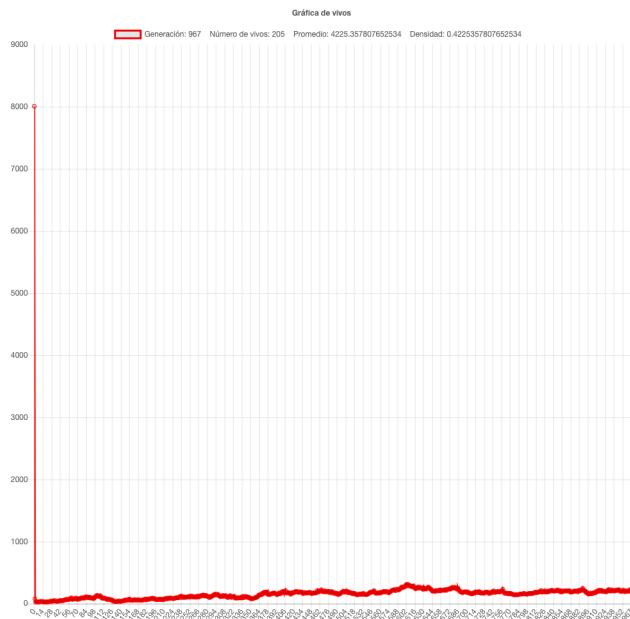


Figura 39: Comportamiento de la población de la simulación anterior



Figura 40: Regla de life con probabilidad de 90 %

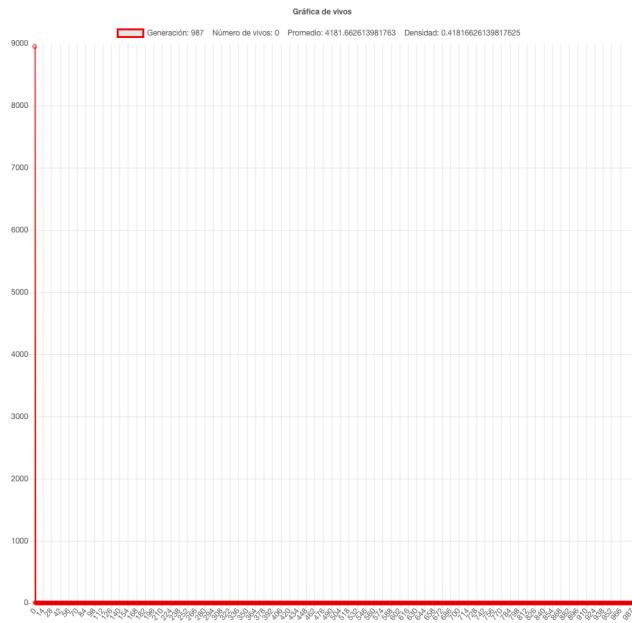


Figura 41: Comportamiento de la población de la simulación anterior

En todas las simulaciones se encuentra un comportamiento similar en el cual la población inicial es muy alta y de manera brusca disminuye y apartir de ahí decrece de manera lenta. En general, la regla de life hace que las poblaciones tiendan a una densidad de .03.

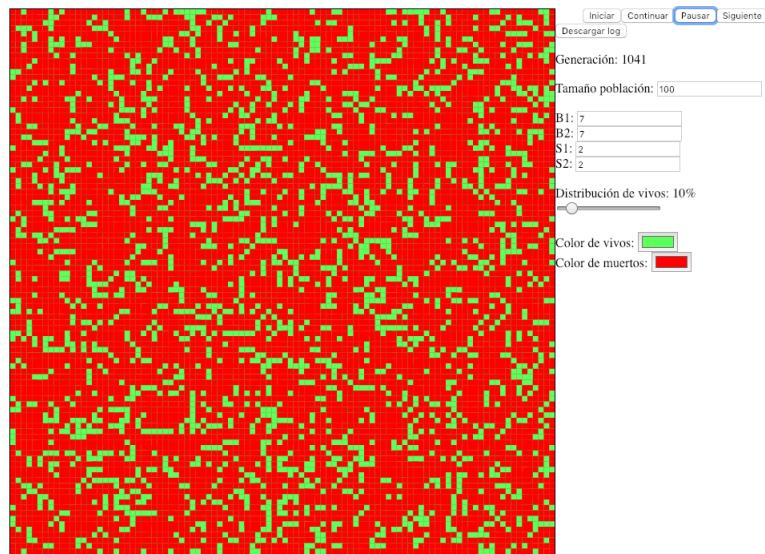


Figura 42: Regla de difusión con probabilidad de 10 %

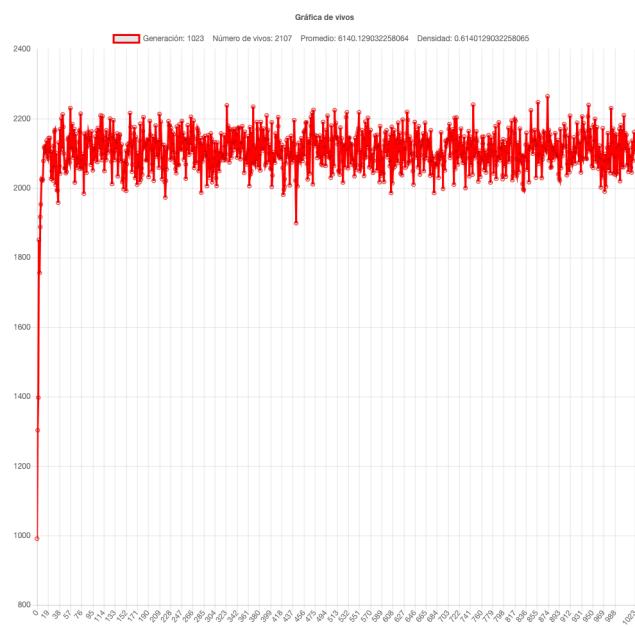


Figura 43: Comportamiento de la población de la simulación anterior

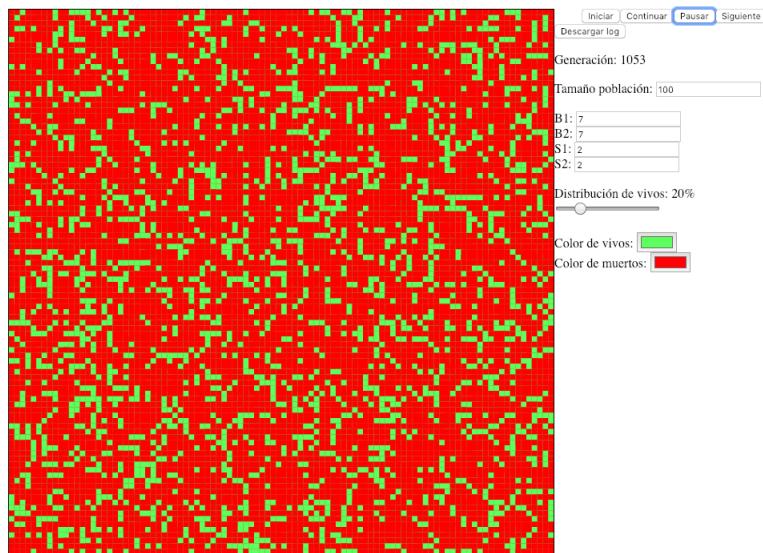


Figura 44: Regla de difusión con probabilidad de 20 %

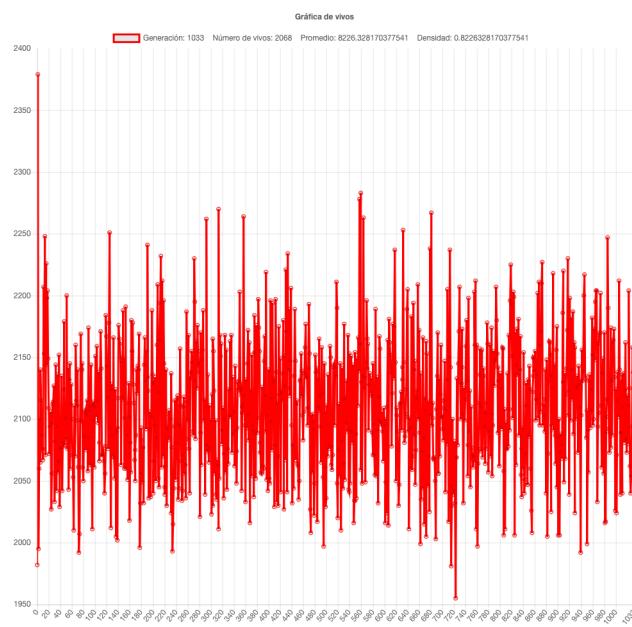


Figura 45: Comportamiento de la población de la simulación anterior

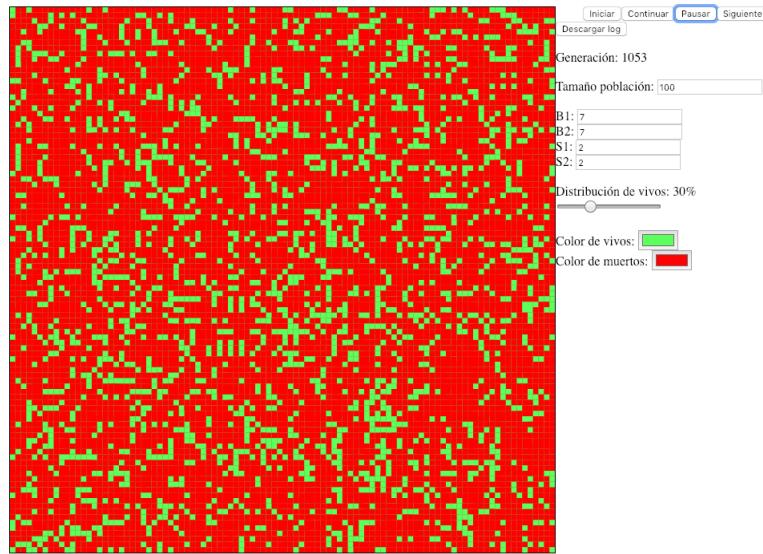


Figura 46: Regla de difusión con probabilidad de 30 %

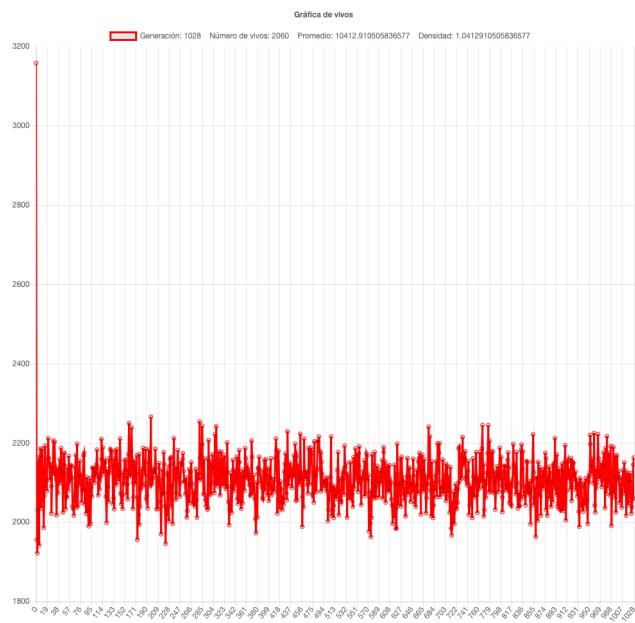


Figura 47: Comportamiento de la población de la simulación anterior

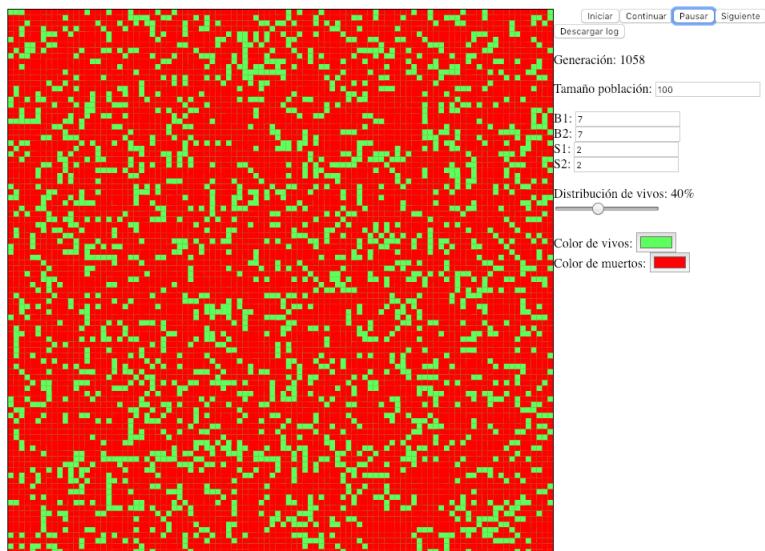


Figura 48: Regla de difusión con probabilidad de 40 %

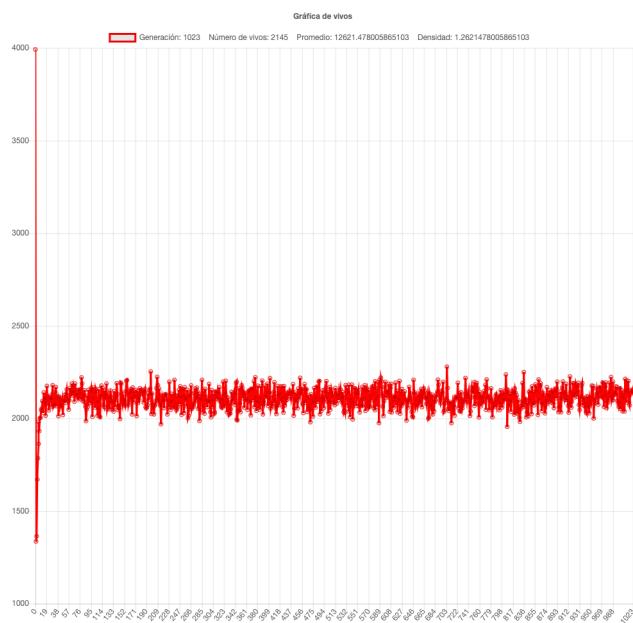


Figura 49: Comportamiento de la población de la simulación anterior

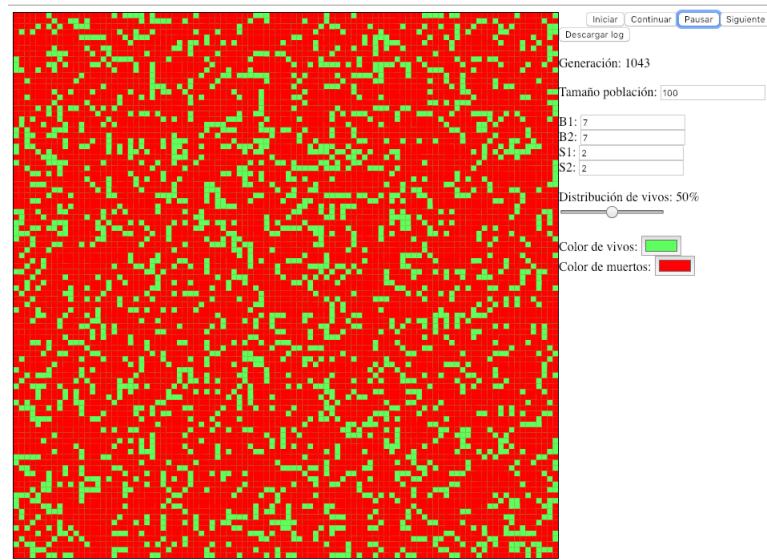


Figura 50: Regla de difusión con probabilidad de 50 %

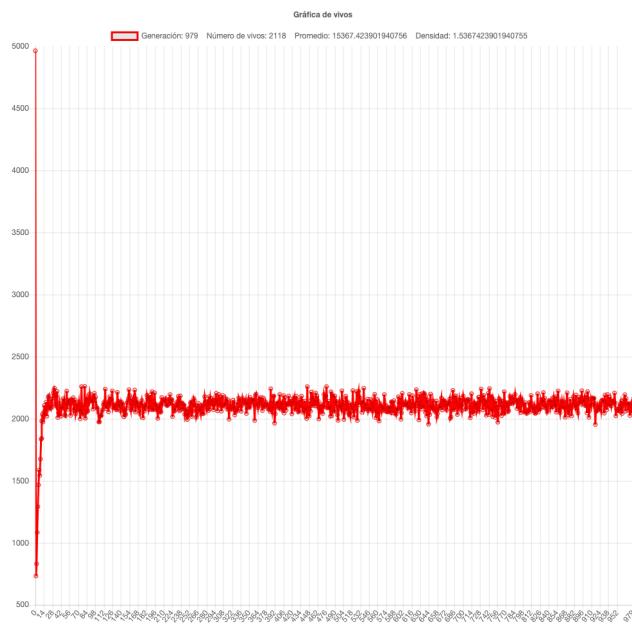


Figura 51: Comportamiento de la población de la simulación anterior

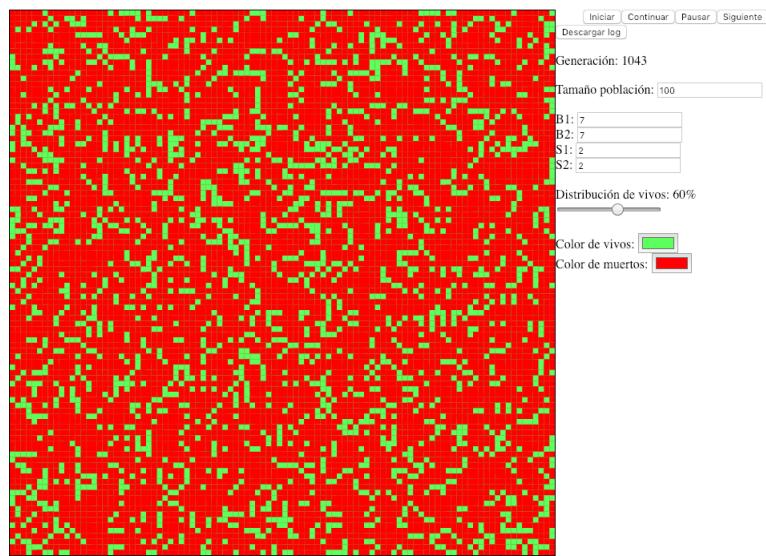


Figura 52: Regla de difusión con probabilidad de 60 %

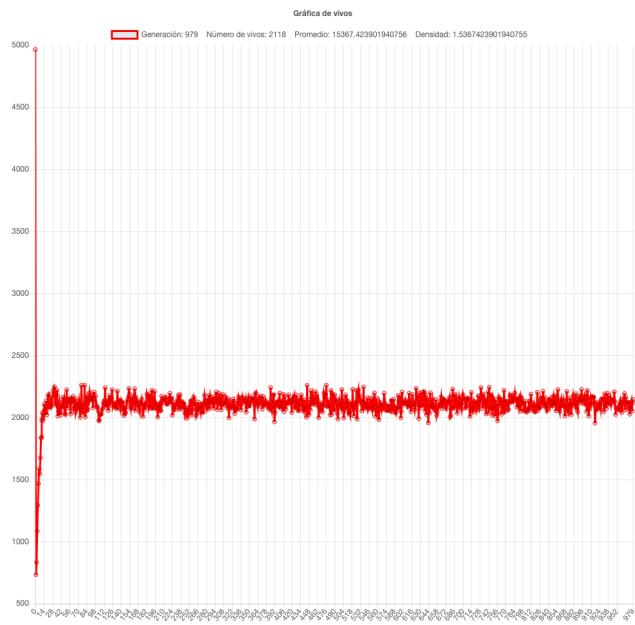


Figura 53: Comportamiento de la población de la simulación anterior

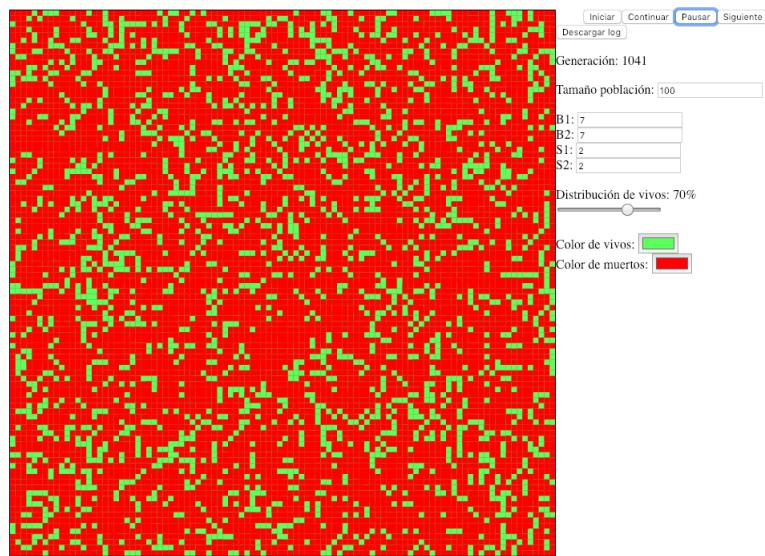


Figura 54: Regla de difusión con probabilidad de 70 %

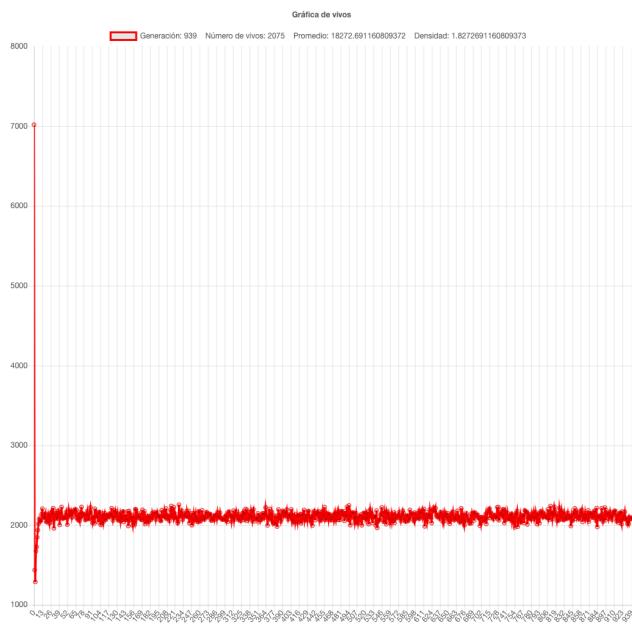


Figura 55: Comportamiento de la población de la simulación anterior

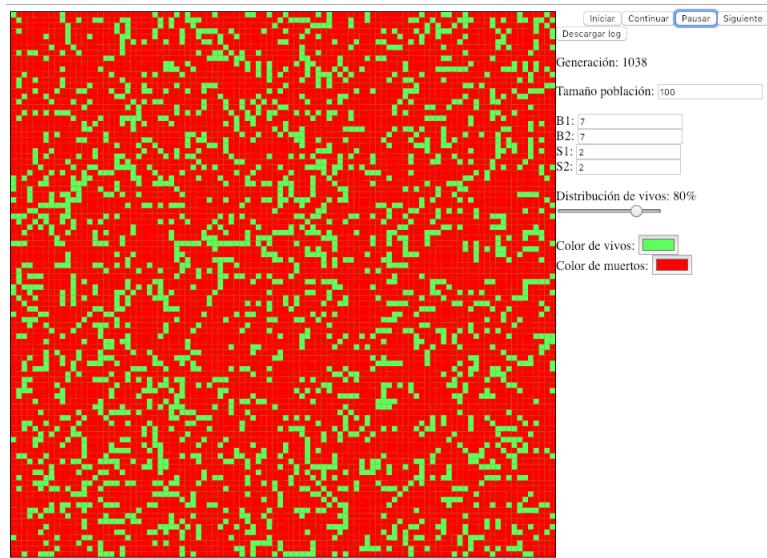


Figura 56: Regla de difusión con probabilidad de 80 %

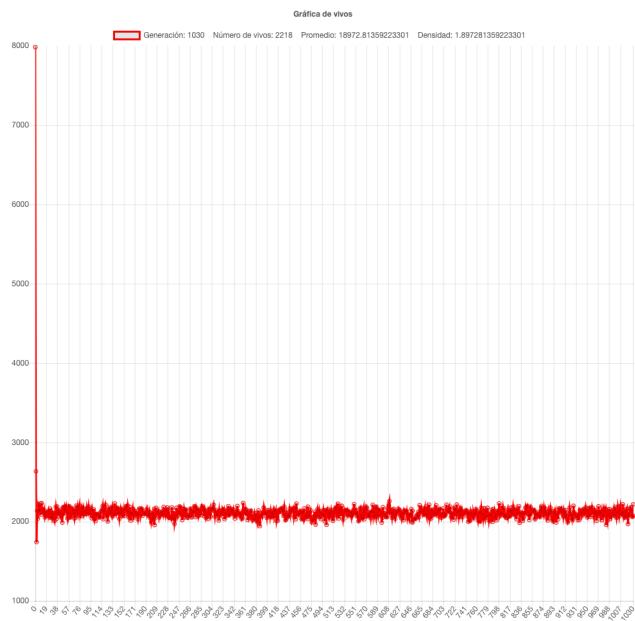


Figura 57: Comportamiento de la población de la simulación anterior

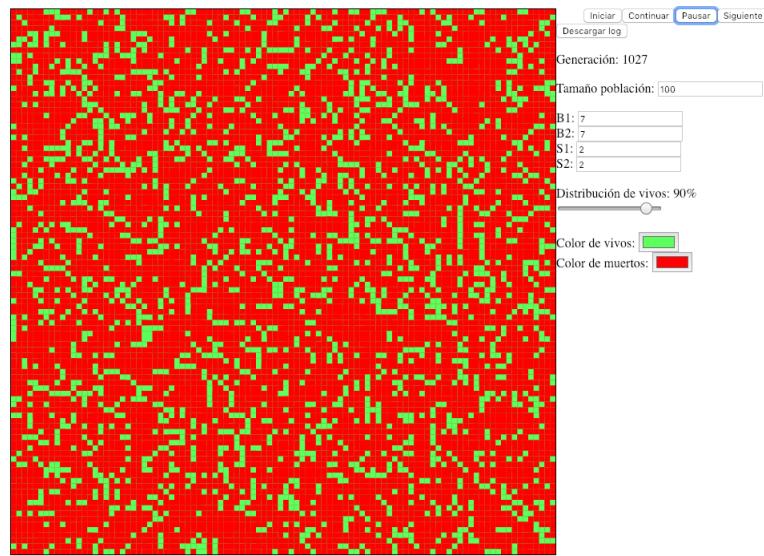


Figura 58: Regla de difusión con probabilidad de 90 %

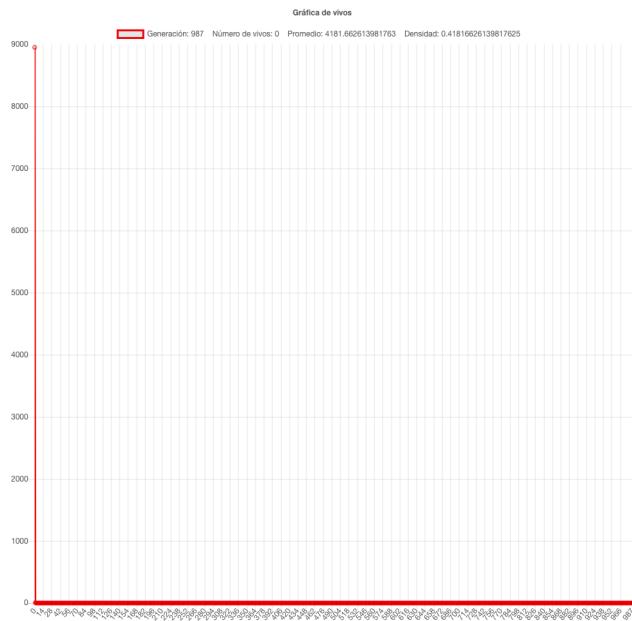


Figura 59: Comportamiento de la población de la simulación anterior

Para la regla de difusión se podria decir que el comportamiento de la población es más estable que la de life, debido a que en todas las simulaciones la cantidad de vivos que se graficaban no era tan variable desde un inicio y el comportamiento de la gráfica es igual en todas las probabilidades de uno que se probaron en donde la población oscila entre un límite mayor y uno menor.

2.5. Conclusiones

Al hacer y probar este programa se pudieron solucionar dos cuestiones, la primera es si existe una configuración en la cual el numero de células no se termine, y la respuesta a esto es que exista un oscilador el cual cambia su posición a lo largo del tiempo, otra posible opción es que haya figuras como un bloque formado por 4 cuadros en el cual no hay cambios.

La siguiente cuestión es si existe una configuración en la cual la población crezca indefinidamente. Para lograr esto es indispensable tener un espacio que sea infinito en el cual se puedan propagar las células a través del tiempo, ya que si no se cuenta con esto en algún punto se tendrán tantos elementos vivos que empezaran a morir por sobre población.

3. Hormiga de Langton

3.1. Introducción

La hormiga de Langton es un autómata celular desarrollado por Chris Langton en 1968, Langton se inspiró en la bioquímica, exploró la posibilidad de implementar la lógica molecular del estado viviente generando así una bioquímica artificial basada en la interacción entre moléculas artificiales. Langton dijo que el comportamiento global de una sociedad es un fenómeno emergente que surge de todas las interacciones locales de sus miembros.

El comportamiento complejo puede surgir de la interacción de las partes muy simples. Por lo que utilizó una colonia de hormigas como modelo para una forma variante de un autómata celular, en donde cada célula puede cambiar de estado, en virtud de los estados de las otras células. [4]

Las hormigas del modelo clásico se mueven en un entorno que consta de células en donde cada una de estas células se encuentra en uno de los dos posibles estados (viva o muerta, 0 o 1), la hormiga viaja en línea recta en el espacio siguiendo las siguientes reglas:

Si se encuentra con una célula muerta, hace un giro a la derecha y sale de la célula invirtiendo el estado de la misma.

Si se encuentra con una célula viva, gira a la izquierda y sale de la célula invirtiendo su estado.

De esta forma la hormiga deja un rastro a medida que se mueve.

3.2. Planteamiento de la práctica

En esta práctica se trabajaron dos versiones diferentes de la hormiga de Langton, la primera que es la implementación clásica de este autómata celular y una segunda versión, en la cual se cuenta con tres tipos de hormigas y algunas otras restricciones que modifican el comportamiento del autómata original.

3.2.1. Hormiga de Langton Original

Las principales características de esta versión es la posibilidad de insertar hormigas por parte del usuario en la posición que se desee además de poder cambiar el color del camino generado por la hormiga. Para poder apreciar de

una forma más clara el comportamiento de la hormiga los movimientos que se realizan tienen un color realizado.

- Si la hormiga esta viendo hacia el sur el color de la hormiga es azul.
- El color sera rojo si la hormiga se encuentra viendo hacia el norte.
- Para el oeste se tiene el color verde.
- Y para el este el color asociado sera el amarillo.

Finalmente, se tiene la posibilidad de generar una cantidad de hormigas basada en una probabilidad de nacimiento, por lo que cada hormiga que se genera tendrá un color diferente. Es importante señalar que el tamaño del espacio en donde se trabaja puede cambiar y las dimensiones de cada célula se ajustan al tamaño del espacio.

3.2.2. Hormiga de Langton Modificada

Esta versión modificada tiene todo la funcionalidad de la versión con excepción de que cada hormiga tenga un color diferente, en este caso los colores generados por las hormigas depende del tipo de hormiga del que se trate. Para las hormigas normales el color asociado es el blanco, para las soldado sera el naranja y para las reinas sera el morado.

El comportamiento es el mismo que en la versión clásica, sin embargo se tienen las siguientes restricciones:

- Cada tipo de hormiga tiene una probabilidad de nacimiento asociada, los valores por defecto son 90 % para las normales, 8 % para las soldado y 2 % para las reinas.
- También se tiene la posibilidad de cambiar estos valores en la interfaz.
- Si una hormiga reina y una soldado se encuentran, se reproducen y nace una nueva hormiga dependiendo de las probabilidades anteriores.
- Se tiene un programa para poder visualizar la población de hormigas separando por tipo de hormiga.

3.3. Desarrollo

El desarrollo de estos programas, se llevo acabo usando Javascript, debido a que el entorno web y del navegador permiten que se puedan desarrollar interfaces para usuario de una forma bastante eficiente. Además, los programas que requieren de usar animaciones o que hacen un uso extensivo de dibujar en un canvas (como es el caso), se ven beneficiados del entorno web que permite todo se ejecute más rápido.

3.3.1. Hormiga de Langton Original

La parte de presentación para el usuario, se encuentra en el siguiente archivo html.

Archivo: hormiga-original.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Hormiga de Langton Original</title>
6     <link rel="stylesheet" type="text/css" href="estilos.css">
7   </head>
8   <body>
9     <div id="contenedor">
10       <!-- <canvas id="myCanvas" width="1260" height="680"
11           style="border:1px solid #000000;"> -->
12       <div>
13         <canvas onclick="clickCanvas(event)" id="myCanvas" width=
14             "680" height="680" style="border:1px solid #000000;">
15         </canvas>
16       </div>
17       <div id="controles">
18         <div id="botones">
19           <!-- <div> -->
20           <button onclick="iniciar()">Iniciar</button>
21           <button onclick="continuar()">Continuar</button>
22           <button onclick="pausa()">Pausar</button>
23           <button onclick="siguiente()">Siguiente</button>
24           <!-- </div> -->
25         </div>
26         <div>
27           <button id="descargar" onclick="descargar()">Descargar
28           log</button>
```

```

26     </div><br>
27     <div>
28         <label id="generacion">Generación: 0</label><br>
29         <label id="vivas">Total de hormigas: 0</label>
30     </div><br>
31     <div>
32         <label for="tam">Tamaño población: </label>
33         <input type="number" id="tam" value="100"><br>
34     </div><br>
35     <div>
36         <label for="distribucion">
37             Distribución de hormigas:
38             <span id="distribucion_valor">50</span>%
39         </label>
40         <input type="range" min="0" max="100" value="50" id=">
41 distribucion">
42     </div><br>
43     <div>
44         <label for="color_norte">Color hormiga norte:</label>
45         <input type="color" value="#ff0000" id="color_norte">
46     <br>
47         <label for="color_este">Color hormiga este:</label>
48         <input type="color" value="#ffff00" id="color_este">
49     <br>
50         <label for="color_oeste">Color hormiga oeste:</label>
51         <input type="color" value="#00cc00" id="color_oeste">
52     <br>
53         <label for="color_sur">Color hormiga sur:</label>
54         <input type="color" value="#0000ff" id="color_sur">
55     <br>
56     </div>
57     </div>
58 <!— <script src="https://ajax.googleapis.com/ajax/libs/jquery/
59 /2.1.1/jquery.min.js">—>
60 <script type="text/javascript" src="hormiga-original.js"></
61 script>
62 </body>
63 </html>

```

La lógica con la que funciona el programa, se encuentra en el siguiente archivo de Javascript.

Archivo: hormiga-original.js

```

1 var c=document.getElementById("myCanvas");
2 var ctx=c.getContext("2d");
3 var longitud=680;
4 var tam=100;
5 var escala=longitud/tam;
6 var random;
7 var generacion=0;
8 var numero_blancos=0;
9 var numero_negros=0;
10 var numero_hormigas=0;
11 var distribucion=50;
12 var intervalo ;
13 var log ;
14 var array ;
15 var array2 ;
16 var x ;
17 var y ;
18 var coordenadas ;
19 var hormigas ;
20 var hormigas2 ;
21 var direccion ;
22 var direccion2 ;
23 var matar ;
24 var color_negro="#000000";
25 var color_blanco="#ffffff";
26 var color_norte="#ff0000";
27 var color_este="#ffff00";
28 var color_sur="#0000ff";
29 var color_oeste="#00cc00";
30
31 //Direcciones :
32 //0->norte , 1->este , 2->south , 3->west
33
34 //Función para descargar
35 function descargar(){
36     var fileContents = log;
37     var fileName = "log.txt";
38     var pp = document.createElement('a');
39     pp.setAttribute('href', 'data:text/plain;charset=utf-8,' +
        encodeURIComponent(fileContents));
40     pp.setAttribute('download', fileName);
41     pp.click();
42 }
43
44 //Modificar valor del slider en pantalla

```

```

45 document.getElementById("distribucion_valor").innerHTML =
46   document.getElementById("distribucion").value;
47 document.getElementById("distribucion").oninput = function(e) {
48   e.preventDefault();
49   document.getElementById("distribucion_valor").innerHTML = this
50     .value;
51 }
52
53 function Create2DArray(rows) {
54   var arr = new Array(rows);
55   for (var i=0;i<rows;i++) {
56     arr[i] = new Array(rows);
57   }
58   return arr;
59 }
60
61 function evaluar(i,j){
62   var estado=array2[i][j];
63   if(hormigas2[i][j].length>0){
64     for(var k=0;k<hormigas2[i][j].length;k++){
65       hormigas[i][j].splice(0, 1);
66       direccion[i][j].splice(0, 1);
67       //Blanco
68       if(estado==1){
69         if(k==0){
70           numero_negros++;
71         }
72         array[i][j]=0;
73         if(hormigas[i][j].length<1){
74           ctx.clearRect(0+(j*(escala)),0+(i*(escala)),escala ,
75           escala);
76           ctx.fillStyle=color_negro;//Negro
77           ctx.fillRect(0+(j*(escala)),0+(i*(escala)),escala ,
78           escala);
79           ctx.stroke();
80         }
81         if(direccion2[i][j][k]==0){
82           if(j==0){
83             hormigas[i][tam-1].push(1);
84             direccion[i][tam-1].push(3);
85             ctx.clearRect(0+((tam-1)*(escala)),0+(i*(escala)),
86             escala , escala);
87             ctx.fillStyle=color_oeste;
88             ctx.fillRect(0+((tam-1)*(escala)),0+(i*(escala)),
89             escala , escala);

```

```

84 } else{
85     hormigas[ i ][ j -1].push(1);
86     direccion[ i ][ j -1].push(3);
87     ctx.clearRect(0+((j-1)*( escala )),0+( i *( escala )), 
88     escala , escala );
89     ctx.fillStyle=color_oeste;
90     ctx.fillRect(0+((j-1)*( escala )),0+( i *( escala )), 
91     escala , escala );
92 } else if(direccion2[ i ][ j ][ k]==1){
93     if( i ==0){
94         hormigas[ tam-1 ][ j ].push(1);
95         direccion[ tam-1 ][ j ].push(0);
96         ctx.clearRect(0+((j)*( escala )),0+((tam-1)*( escala )), 
97         escala , escala );
98         ctx.fillStyle=color_norte;
99         ctx.fillRect(0+((j)*( escala )),0+((tam-1)*( escala )), 
100        escala , escala );
101    } else{
102        hormigas[ i -1 ][ j ].push(1);
103        direccion[ i -1 ][ j ].push(0);
104        ctx.clearRect(0+((j)*( escala )),0+((i-1)*( escala )), 
105        escala , escala );
106        ctx.fillStyle=color_norte;
107    } else if(direccion2[ i ][ j ][ k]==2){
108        if( j ==tam-1){
109            hormigas[ i ][ 0 ].push(1);
110            direccion[ i ][ 0 ].push(1);
111            ctx.clearRect(0+((0)*( escala )),0+(( i )*( escala )), 
112            escala , escala );
113            ctx.fillStyle=color_este;
114        } else{
115            hormigas[ i ][ j +1].push(1);
116            direccion[ i ][ j +1].push(1);
117            ctx.clearRect(0+((j+1)*( escala )),0+(( i )*( escala )), 
118            escala , escala );
119            ctx.fillStyle=color_este ;

```

```

119         ctx.fillRect(0+((j+1)*( escala )) ,0+(( i )*( escala )) ,
120         escala , escala );
121         }
122         ctx.stroke();
123     }else if(direccion2 [ i ] [ j ] [ k]==3){
124         if( i==tam-1){
125             hormigas [ 0 ] [ j ].push(1);
126             direccion [ 0 ] [ j ].push(2);
127             ctx.clearRect(0+((j)*( escala )) ,0+((0)*( escala )) ,
128             escala , escala );
129             ctx.fillStyle=color_sur;
130             ctx.fillRect(0+((j)*( escala )) ,0+((0)*( escala )) ,
131             escala , escala );
132         }else{
133             hormigas [ i +1][ j ].push(1);
134             direccion [ i +1][ j ].push(2);
135             ctx.clearRect(0+((j)*( escala )) ,0+(( i +1)*( escala )) ,
136             escala , escala );
137             ctx.fillStyle=color_sur;
138             ctx.fillRect(0+((j)*( escala )) ,0+(( i +1)*( escala )) ,
139             escala , escala );
140             }
141             ctx.stroke();
142         }
143     }else { //Negro
144         if(k==0){
145             numero_negros--;
146         }
147         array [ i ] [ j ]=1;
148         if(hormigas [ i ] [ j ].length <1){
149             ctx.clearRect(0+(j*( escala )) ,0+( i *( escala )) ,escala ,
150             escala );
151             ctx.fillStyle=color_blanco ; //Blanco
152             ctx.fillRect(0+(j*( escala )) ,0+( i *( escala )) ,escala ,
153             escala );
154             ctx.stroke();
155         }
156     }if( direccion2 [ i ] [ j ] [ k]==0{
157         if(j==tam-1{
158             hormigas [ i ] [ 0 ].push(1);
159             direccion [ i ] [ 0 ].push(1);
160             ctx.clearRect(0+((0)*( escala )) ,0+(( i *( escala )) ,escala
161             , escala );
162             ctx.fillStyle=color_este ;
163

```

```

155     ctx.fillRect(0+((0)*( escala )),0+( i*( escala )), escala ,
156     escala );
157     } else{
158         hormigas[ i ][ j+1].push(1);
159         direccion[ i ][ j+1].push(1);
160         ctx.clearRect(0+(( j+1)*( escala )),0+( i*( escala )), escala , escala );
161         ctx.fillStyle=color_este;
162         ctx.fillRect(0+(( j+1)*( escala )),0+( i*( escala )), escala , escala );
163         }
164         ctx.stroke();
165         } else if(direccion2[ i ][ j ][ k]==1){
166             if( i==tam-1){
167                 hormigas[ 0 ][ j ].push(1);
168                 direccion[ 0 ][ j ].push(2);
169                 ctx.clearRect(0+(( j)*( escala )),0+((0)*( escala )), escala , escala );
170                 ctx.fillStyle=color_sur;
171                 ctx.fillRect(0+(( j)*( escala )),0+((0)*( escala )), escala , escala );
172             } else{
173                 hormigas[ i+1 ][ j ].push(1);
174                 direccion[ i+1 ][ j ].push(2);
175                 ctx.clearRect(0+(( j)*( escala )),0+(( i+1)*( escala )), escala , escala );
176                 ctx.fillStyle=color_sur;
177                 ctx.fillRect(0+(( j)*( escala )),0+(( i+1)*( escala )), escala , escala );
178             }
179             ctx.stroke();
180             } else if(direccion2[ i ][ j ][ k]==2){
181                 if( j==0){
182                     hormigas[ i ][ tam-1].push(1);
183                     direccion[ i ][ tam-1].push(3);
184                     ctx.clearRect(0+(( tam-1)*( escala )),0+(( i)*( escala )), escala , escala );
185                     ctx.fillStyle=color_oeste;
186                     ctx.fillRect(0+(( tam-1)*( escala )),0+(( i)*( escala )), escala , escala );
187                 } else{
188                     hormigas[ i ][ j-1].push(1);
189                     direccion[ i ][ j-1].push(3);

```

```

190         ctx.fillStyle=color_oeste;
191         ctx.fillRect(0+((j-1)*(escala)),0+((i)*(escala)),
192         escala,escala);
193         }
194         ctx.stroke();
195     }else if(direccion2[i][j][k]==3){
196     if(i==0){
197         hormigas[tam-1][j].push(1);
198         direccion[tam-1][j].push(0);
199         ctx.clearRect(0+((j)*(escala)),0+((tam-1)*(escala)),
200         escala,escala);
201         ctx.fillStyle=color_norte;
202         ctx.fillRect(0+((j)*(escala)),0+((tam-1)*(escala)),
203         escala,escala);
204     }else{
205         hormigas[i-1][j].push(1);
206         direccion[i-1][j].push(0);
207         ctx.clearRect(0+((j)*(escala)),0+((i-1)*(escala)),
208         escala,escala);
209         ctx.fillStyle=color_norte;
210         ctx.fillRect(0+((j)*(escala)),0+((i-1)*(escala)),
211         escala,escala);
212     }
213 }
214
215 function copiar(arr){
216     var aux=Create2DArray(tam);
217     for(var i=0; i<tam; i++){
218         for(var j=0;j<tam; j++){
219             aux[i][j]=arr[i][j];
220         }
221     }
222     return aux;
223 }
224
225 function copiar2(arr){
226     var aux=Create2DArray(tam);
227     for(var i=0; i<tam; i++){
228         for(var j=0;j<tam; j++){
229             aux[i][j]=[];

```

```

230     for ( var k=0;k<arr [ i ][ j ].length ;k++){
231         aux [ i ][ j ][ k]=arr [ i ][ j ][ k ];
232     }
233 }
234 }
235 return aux;
236 }

237
238 function obtenerTamano(){
239     tam = parseInt(document .getElementById("tam") .value );
240     escala=longitud/tam;
241     array=Create2DArray(tam);
242     array2=Create2DArray(tam);
243     hormigas=Create2DArray(tam);
244     hormigas2=Create2DArray(tam);
245     coordenadas=Create2DArray(tam);
246     direccion=Create2DArray(tam);
247     direccion2=Create2DArray(tam);
248     matar=Create2DArray(tam);
249     for ( var i=0;i<tam ; i++){
250         for ( var j=0;j<tam ; j++){
251             hormigas [ i ][ j ]=[];
252             hormigas2 [ i ][ j ]=[];
253             direccion [ i ][ j ]=[];
254             direccion2 [ i ][ j ]=[];
255         }
256     }
257 }

258
259 function obtenerDistribucion(){
260     distribucion = parseInt(document .getElementById("distribucion")
261         ) .value )/100;
262     distribucion = parseFloat( distribucion .toFixed(2));
263 }

264 function obtenerConfiguracion(){
265     obtenerTamano();
266     obtenerDistribucion();
267 }

268
269 function limpiar(){
270     ctx .clearRect(0, 0, ctx .width , ctx .height );
271     console .clear ();
272     generacion=0;
273     numero_blanco=0;

```

```

274     numero_negros=0;
275     numero_hormigas=0;
276     log="";
277     document.getElementById('generacion').innerHTML ="Generación:
278         0";
279 }
280 function iniciar(){
281     limpiar();
282     obtenerConfiguracion();
283     for (var i = 0; i < tam; i++) {
284         for(var j=0;j<tam;j++){
285             // random=Math.floor((Math.random() * 2) + 0);
286             if (Math.random() < distribucion){
287                 random=Math.floor((Math.random() * 4) + 0);
288                 // random=3;
289                 array [i][j]=1;
290                 array2 [i][j]=1;
291                 numero_hormigas++;
292                 hormigas [i][j].push(1);
293                 hormigas2 [i][j].push(1);
294                 direccion [i][j].push(random);
295                 direccion2 [i][j].push(random);
296                 matar [i][j]=0;
297                 if (random==0){
298                     //Norte -> Rojo
299                     ctx.fillStyle=color_norte;
300
301                 } else if (random==1){
302                     //Este -> Amarillo
303                     ctx.fillStyle=color_este;
304                 } else if (random==2){
305                     //Sur -> Azul
306                     ctx.fillStyle=color_sur;
307                 } else{
308                     //Oeste -> Verde
309                     ctx.fillStyle=color_oeste;
310                 }
311             }
312             else{
313                 array [i][j]=0;
314                 array2 [i][j]=0;
315                 ctx.fillStyle=color_negro;
316                 numero_negros++;
317             }
}

```

```

318     x=0+j*escala ;
319     y=0+i*escala ;
320     coordenadas [ i ][ j ]=x+” , ”+y+” , ”+(x+escala )+” , ”+(y+escala ) ;
321     ctx . fillRect ( x,y , escala , escala );
322     ctx . stroke ();
323   }
324 }
325 console . log ( numero_negros+” , ”+generacion );
326 log=numero_negros+” , ”+generacion+” \n ”;
327 document . getElementById ( ' vivas ' ) . innerHTML =” Total de hormigas
328 : ”+numero_hormigas ;
329
330 function ejecutar () {
331   generacion++;
332   for ( var i=0; i<tam; i++ ){
333     for ( var j=0;j<tam; j++ ){
334       evaluar ( i , j );
335     }
336   }
337   console . log ( numero_negros+” , ”+generacion );
338   document . getElementById ( ' generacion ' ) . innerHTML =” Generación :
339   ”+generacion ;
340   log+=numero_negros+” , ”+generacion+” \n ”;
341   array2=copiar ( array );
342   hormigas2=copiar2 ( hormigas );
343   direccion2=copiar2 ( direccion );
344 }
345 function continuar (){
346   intervalo=setInterval ( ejecutar , 10 );
347 }
348
349 function pausa (){
350   clearInterval ( intervalo );
351 }
352 function siguiente (){
353   ejecutar ();
354 }
355
356 function obtenerCoordenadas ( i , j ){
357   var texto=coordenadas [ i ][ j ];
358   return texto . split ( ” , ” );
359 }
360

```

```

361 function clickCanvas(event){
362     x=event.offsetX;
363     y=event.offsetY;
364     // console.log(x + "," +y);
365     for(var i=0;i<tam;i++){
366         for(var j=0;j<tam;j++){
367             var arr=obtenerCoordenadas(i,j);
368             if(x >= arr[0] && x <= arr[2] && y >= arr[1] && y <= arr[3]){
369                 x=0+j*escala;
370                 y=0+i*escala;
371                 if(hormigas2[i][j].length<1){
372                     console.log("Soy:" +i+"," +j);
373                     random=Math.floor((Math.random() * 4) + 0);
374                     // random=3;
375                     numero_hormigas++;
376                     numero_negros--;
377                     hormigas[i][j].push(1);
378                     hormigas2[i][j].push(1);
379                     direccion[i][j].push(random);
380                     direccion2[i][j].push(random);
381                     matar[i][j]=0;
382                     if(random==0){
383                         //Norte -> Rojo
384                         ctx.fillStyle=color_norte;
385
386                     } else if(random==1){
387                         //Este -> Amarillo
388                         ctx.fillStyle=color_este;
389                     } else if(random==2){
390                         //Sur -> Azul
391                         ctx.fillStyle=color_sur;
392                     } else{
393                         //Oeste -> Verde
394                         ctx.fillStyle=color_oeste;
395                     }
396                 } else{
397                     if(matar[i][j]==4){
398                         hormigas[i][j].pop();
399                         hormigas2[i][j].pop();
400                         direccion[i][j].pop();
401                         direccion2[i][j].pop();
402                         ctx.fillStyle=color_negro;
403                         matar[i][j]=0;
404                         numero_hormigas--;

```

```

405     numero_negros++;
406 } else if(direccion[i][j][0]==0){
407     direccion[i][j][0]=1;
408     direccion2[i][j][0]=1;
409     ctx.fillStyle=color_este ;
410     matar[i][j]++;
411 } else if(direccion[i][j][0]==1){
412     direccion[i][j][0]=2;
413     direccion2[i][j][0]=2;
414     ctx.fillStyle=color_sur ;
415     matar[i][j]++;
416 } else if(direccion[i][j][0]==2){
417     direccion[i][j][0]=3;
418     direccion2[i][j][0]=3;
419     ctx.fillStyle=color_oeste ;
420     matar[i][j]++;
421 } else if(direccion[i][j][0]==3){
422     direccion[i][j][0]=0;
423     direccion2[i][j][0]=0;
424     ctx.fillStyle=color_norte ;
425     matar[i][j]++;
426 }
427 }
428 ctx.fillRect(x,y,escala ,escala );
429 ctx.stroke();
430 break;
431 }
432 }
433 }
434 document.getElementById('vivas').innerHTML ="Total de hormigas
435 : "+numero_hormigas;
}

```

3.3.2. Hormiga de Langton Modificada

La parte de presentación para el usuario, se encuentra en el siguiente archivo html.

Archivo: hormiga.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">

```

```

5   <title>Hormiga de Langton Modificada</title>
6   <link rel="stylesheet" type="text/css" href="estilos.css">
7 </head>
8 <body>
9   <div id="contenedor">
10    <!-- <canvas id="myCanvas" width="1260" height="680"
11      style="border:1px solid #000000;"> -->
12    <div>
13      <canvas onclick="clickCanvas(event)" id="myCanvas" width
14        ="680" height="680" style="border:1px solid #000000;">
15      </canvas>
16    </div>
17    <div id="">
18      <div id="botones">
19        <!— <div> —>
20        <button onclick="iniciar()">Iniciar</button>
21        <button onclick="continuar()">Continuar</button>
22        <button onclick="pausa()">Pausar</button>
23        <button onclick="siguiente()">Siguiente</button>
24        <!— </div> —>
25      </div>
26      <div>
27        <button id="descargar" onclick="descargar()">Descargar
28 log</button>
29      </div><br>
30      <div>
31        <label id="generacion">Generación: 0</label><br>
32        <label id="vivas">Total de hormigas: 0</label>
33      </div><br>
34      <div>
35        <label for="tam">Tamaño población: </label>
36        <input type="number" id="tam" value="100"><br>
37      </div><br>
38      <div>
39        <label for="distribucion">
40          Distribución de hormigas:
41          <span id="distribucion_valor">50</span>%<br>
42        </label>
43        <input type="range" min="0" max="100" value="50" id=">
44 distribucion">
45      </div><br>
46      <div>
47        <label for="obrera">Probabilidad obrera: </label>
48        <input type="number" id="obrera" value="90"><br>
49        <label for="soldado">Probabilidad soldado: </label>

```

```

46      <input type="number" id="soldado" value="8"><br>
47      <label for="reina">Probabilidad reina: </label>
48      <input type="number" id="reina" value="2"><br>
49    </div><br>
50    <div>
51      <label for="color_norte">Color hormiga norte:</label>
52      <input type="color" value="#ff0000" id="color_norte">
53    <br>
54      <label for="color_este">Color hormiga este:</label>
55      <input type="color" value="#ffff00" id="color_este">
56    <br>
57      <label for="color_oeste">Color hormiga oeste:</label>
58      <input type="color" value="#00cc00" id="color_oeste">
59    <br>
60      <label for="color_sur">Color hormiga sur:</label>
61      <input type="color" value="#0000ff" id="color_sur">
62    <br>
63      <label for="color_reina">Color hormiga reina:</label>
64      <input type="color" value="#3366ff" id="color_reina">
65    <br>
66      <label for="color_soldado">Color hormiga soldado:</
67      label>
68      <input type="color" value="#66ff33" id="color_soldado
69    "><br>
70      <label for="color_obrera">Color hormiga obrera:</label
71    >
72      <input type="color" value="#ffffff" id="color_obrera">
73    </div>
74  </div>
75  </div>
76
77  <canvas id="myChart" width="600" height="600"></canvas>
78  <canvas id="myChart2" width="600" height="600"></canvas>
79  <script src="Chart.min.js"></script>
80  <script type="text/javascript" src="hormiga.js"></script>
81
82</body>
83</html>

```

La lógica con la que funciona el programa, se encuentra en el siguiente archivo de Javascript.

Archivo: hormiga.js

```

1 var c=document.getElementById("myCanvas");
2 var chart = document.getElementById("myChart");

```

```

3 var chart2=document.getElementById("myChart2");
4 var ctx=c.getContext("2d");
5 var longitud=680;
6 var tam=100;
7 var escala=longitud/tam;
8 var random;
9 var generacion=0;
10 var numero_blancos=0;
11 var numero_negros=0;
12 var numero_hormigas=0;
13 var numero_reinas=0;
14 var numero_soldados=0;
15 var numero_obreras=0;
16 var intervalo;
17 var log;
18 var array;
19 var array2;
20 var x;
21 var y;
22 var coordenadas;
23 var hormigas;
24 var hormigas2;
25 var direccion;
26 var direccion2;
27 var matar;
28 var generacion_grafica=[];
29 var obreras_grafica=[];
30 var soldados_grafica=[];
31 var reinas_grafica=[];
32 var hormigas_grafica=[];
33 var color_negro="#000000";
34 var color_blanco="#ffffff";
35 var color_norte="#ff0000";
36 var color_este="#ffff00";
37 var color_sur="#0000ff";
38 var color_oeste="#00cc00";
39 var color_reina="#3366ff";
40 var color_soldado="#66ff33";
41 var color_obra=color_blanco;
42 var probabilidad=.05;
43 var colores_direcciones=[color_norte,color_este,color_sur,
    color_oeste];
44
45 //1-> Obrera
46 //2-> Soldado

```

```

47 //3-> Reina
48 var tipos_hormiga=[1,2,3];
49 var colores=[color_obra, color_soldado, color_reina];
50 var probabilidades_hormiga=[.9,.08,.02];
51 probabilidades_hormiga.sort(function(a, b){ return b-a});
52
53 //Modificar valor del slider en pantalla
54 document.getElementById("distribucion_valor").innerHTML =
    document.getElementById("distribucion").value;
55 document.getElementById("distribucion").oninput = function(e) {
56     e.preventDefault();
57     document.getElementById("distribucion_valor").innerHTML = this
        .value;
58 }
59
60 //Direcciones:
61 //0->norte, 1->este, 2->sur, 3->oeste
62
63 //Función para descargar
64 function descargar(){
65     var fileContents = log;
66     var fileName = "log.txt";
67     var pp = document.createElement('a');
68     pp.setAttribute('href', 'data:text/plain;charset=utf-8,' +
        encodeURIComponent(fileContents));
69     pp.setAttribute('download', fileName);
70     pp.click();
71 }
72
73 function Create2DArray(rows) {
74     var arr = new Array(rows);
75     for (var i=0;i<rows; i++) {
76         arr[i] = new Array(rows);
77     }
78     return arr;
79 }
80
81 function evaluar(i,j){
82     var estado=array2[i][j];
83     if(hormigas2[i][j].length>0){
84         if(hormigas2[i][j].length==2){
85             if((hormigas2[i][j][0]==2 && hormigas2[i][j][1]==3) || (
86                 hormigas2[i][j][0]==3 && hormigas2[i][j][1]==2)){
87                 //Hacemos que nazca una nueva hormiga
88                 random=Math.floor((Math.random() * 4) + 0);

```

```

88     numero_hormigas++;
89     direccion[i][j].push(random);
90     direccion2[i][j].push(random);
91     obtenerTipoHormiga(i,j);
92     matar[i][j]=0;
93   }
94 }
95 for(var k=0;k<hormigas2[i][j].length;k++){
96   var tipo_hormiga=hormigas[i][j].splice(0, 1);
97   direccion[i][j].splice(0, 1);
98   //Blanco
99   if(estado==1){
100     if(k==0){
101       numero_negros++;
102     }
103     array[i][j]=0;
104     if(hormigas[i][j].length<1){
105       ctx.clearRect(0+(j*(escala)),0+(i*(escala)),escala,
106 escala);
107       ctx.fillStyle=color_negro;//Negro
108       ctx.fillRect(0+(j*(escala)),0+(i*(escala)),escala,
109 escala);
110       ctx.stroke();
111     }
112     if(direccion2[i][j][k]==0){
113       if(j==0){
114         hormigas[i][tam-1].push(tipo_hormiga);
115         direccion[i][tam-1].push(3);
116         ctx.clearRect(0+((tam-1)*(escala)),0+(i*(escala)),
117 escala, escala);
118         ctx.fillStyle=color_oeste;
119         ctx.fillRect(0+((tam-1)*(escala)),0+(i*(escala)),
120 escala, escala);
121       } else{
122         hormigas[i][j-1].push(tipo_hormiga);
123         direccion[i][j-1].push(3);
124         ctx.clearRect(0+((j-1)*(escala)),0+(i*(escala)),
125 escala, escala);
126       }
127       ctx.stroke();
128     } else if(direccion2[i][j][k]==1){
129       if(i==0){

```

```

127     hormigas[tam-1][j].push(tipo_hormiga);
128     direccion[tam-1][j].push(0);
129     ctx.clearRect(0+((j)*(escala)),0+((tam-1)*(escala)),
130     escala, escala);
131     ctx.fillStyle=color_norte;
132     ctx.fillRect(0+((j)*(escala)),0+((tam-1)*(escala)),
133     escala, escala);
134     }else{
135         hormigas[i-1][j].push(tipo_hormiga);
136         direccion[i-1][j].push(0);
137         ctx.clearRect(0+((j)*(escala)),0+((i-1)*(escala)),
138         escala, escala);
139         ctx.strokeStyle="black";
140     }else if(direccion2[i][j][k]==2){
141         if(j==tam-1){
142             hormigas[i][0].push(tipo_hormiga);
143             direccion[i][0].push(1);
144             ctx.clearRect(0+((0)*(escala)),0+((i)*(escala)),
145             escala, escala);
146             ctx.fillStyle="red";
147             ctx.fillRect(0+((0)*(escala)),0+((i)*(escala)),
148             escala, escala);
149             }else{
150                 hormigas[i][j+1].push(tipo_hormiga);
151                 direccion[i][j+1].push(1);
152                 ctx.clearRect(0+((j+1)*(escala)),0+((i)*(escala)),
153                 escala, escala);
154                 ctx.strokeStyle="black";
155             }else if(direccion2[i][j][k]==3){
156                 if(i==tam-1){
157                     hormigas[0][j].push(tipo_hormiga);
158                     direccion[0][j].push(2);
159                     ctx.clearRect(0+((j)*(escala)),0+((0)*(escala)),
160                     escala, escala);
161                     ctx.fillStyle="blue";
162                     ctx.fillRect(0+((j)*(escala)),0+((0)*(escala)),

```

```

162     } else{
163         hormigas[ i +1][ j ]. push( tipo_hormiga );
164         direccion[ i +1][ j ]. push( 2 );
165         ctx . clearRect( 0+(( j )*( escala )), 0+(( i +1)*( escala )), ,
166         escala , escala );
167         ctx . fillStyle=color_sur ;
168         ctx . fillRect( 0+(( j )*( escala )), 0+(( i +1)*( escala )), ,
169         escala , escala );
170         ctx . stroke();
171     }
172 } else{//Negro
173     if( k==0){
174         numero_negros--;
175     }
176     array[ i ][ j ]=1;
177     if( hormigas[ i ][ j ]. length <1){
178         ctx . clearRect( 0+( j *( escala )), 0+( i *( escala )), escala ,
179         escala );
180         //Aquí se pinta dependiendo del tipo de hormiga
181         ctx . fillStyle=colores[ tipo_hormiga -1];//Color
182         correspondiente
183         ctx . fillRect( 0+( j *( escala )), 0+( i *( escala )), escala ,
184         escala );
185         ctx . stroke();
186     }
187     if( direccion2[ i ][ j ][ k ]==0){
188         if( j ==tam-1){
189             hormigas[ i ][ 0 ]. push( tipo_hormiga );
190             direccion[ i ][ 0 ]. push( 1 );
191             ctx . clearRect( 0+(( 0)*( escala )), 0+(( i *( escala )), escala
192             , escala );
193             ctx . fillStyle=color_este ;
194             ctx . fillRect( 0+(( 0)*( escala )), 0+(( i *( escala )), escala ,
195             escala );
196         } else{
197             hormigas[ i ][ j +1]. push( tipo_hormiga );
198             direccion[ i ][ j +1]. push( 1 );
199             ctx . clearRect( 0+(( j +1)*( escala )), 0+(( i *( escala )), escala ,
200             escala );
201             ctx . fillStyle=color_este ;
202             ctx . fillRect( 0+(( j +1)*( escala )), 0+(( i *( escala )), escala ,
203             escala );
204         }
205     }
206     ctx . stroke();

```

```

198     } else if(direccion2[i][j][k]==1){
199         if(i==tam-1){
200             hormigas[0][j].push(tipo_hormiga);
201             direccion[0][j].push(2);
202             ctx.clearRect(0+((j)*(escala)),0+((0)*(escala)),
203                         escala,escala);
203             ctx.fillStyle=color_sur;
204             ctx.fillRect(0+((j)*(escala)),0+((0)*(escala)),
205                         escala,escala);
205         } else{
206             hormigas[i+1][j].push(tipo_hormiga);
207             direccion[i+1][j].push(2);
208             ctx.clearRect(0+((j)*(escala)),0+((i+1)*(escala)),
209                         escala,escala);
210             ctx.fillStyle=color_sur;
211             ctx.fillRect(0+((j)*(escala)),0+((i+1)*(escala)),
210                         escala,escala);
211         }
212         ctx.stroke();
213     } else if(direccion2[i][j][k]==2){
214         if(j==0){
215             hormigas[i][tam-1].push(tipo_hormiga);
216             direccion[i][tam-1].push(3);
217             ctx.clearRect(0+((tam-1)*(escala)),0+((i)*(escala)),
218                         escala,escala);
218             ctx.fillStyle=color_oeste;
219             ctx.fillRect(0+((tam-1)*(escala)),0+((i)*(escala)),
219                         escala,escala);
220         } else{
221             hormigas[i][j-1].push(tipo_hormiga);
222             direccion[i][j-1].push(3);
223             ctx.clearRect(0+((j-1)*(escala)),0+((i)*(escala)),
224                         escala,escala);
224             ctx.fillStyle=color_oeste;
225             ctx.fillRect(0+((j-1)*(escala)),0+((i)*(escala)),
225                         escala,escala);
226         }
227         ctx.stroke();
228     } else if(direccion2[i][j][k]==3){
229         if(i==0){
230             hormigas[tam-1][j].push(tipo_hormiga);
231             direccion[tam-1][j].push(0);
232             ctx.clearRect(0+((j)*(escala)),0+((tam-1)*(escala)),
233                         escala,escala);
233             ctx.fillStyle=color_norte;

```

```

234         ctx.fillRect(0+((j)*(escala)),0+((tam-1)*(escala)),  

235         escala,escala);  

236     }  

237     }  

238     hormigas[i-1][j].push(tipo_hormiga);  

239     direccion[i-1][j].push(0);  

240     ctx.clearRect(0+((j)*(escala)),0+((i-1)*(escala)),  

241     escala,escala);  

242     ctx.fillStyle=color_norte;  

243     ctx.fillRect(0+((j)*(escala)),0+((i-1)*(escala)),  

244     escala,escala);  

245     }  

246   }  

247 }  

248  

249 function copiar(arr){  

250   var aux=Create2DArray(tam);  

251   for(var i=0; i<tam; i++){  

252     for(var j=0;j<tam; j++){  

253       aux[i][j]=arr[i][j];  

254     }  

255   }  

256   return aux;  

257 }  

258  

259 function copiar2(arr){  

260   var aux=Create2DArray(tam);  

261   for(var i=0; i<tam; i++){  

262     for(var j=0;j<tam; j++){  

263       aux[i][j]=[];  

264       for(var k=0;k<arr[i][j].length ;k++){  

265         aux[i][j][k]=arr[i][j][k];  

266       }  

267     }  

268   }  

269   return aux;  

270 }  

271  

272 function obtenerTamano(){  

273   tam = parseInt(document.getElementById("tam").value);  

274   escala=longitud/tam;  

275   array=Create2DArray(tam);

```

```

276 array2=Create2DArray(tam);
277 hormigas=Create2DArray(tam);
278 hormigas2=Create2DArray(tam);
279 coordenadas=Create2DArray(tam);
280 direccion=Create2DArray(tam);
281 direccion2=Create2DArray(tam);
282 matar=Create2DArray(tam);
283 vida=Create2DArray(tam);
284 for ( var i=0;i<tam; i++){
285     for ( var j=0;j<tam; j++){
286         hormigas[ i ][ j ]=[];
287         hormigas2[ i ][ j ]=[];
288         direccion[ i ][ j ]=[];
289         direccion2[ i ][ j ]=[];
290     }
291 }
292 }
293
294 function obtenerDistribucion (){
295     probabilidad = parseInt(document.getElementById("distribucion")
296     ).value)/100;
297     probabilidad = parseFloat(probabilidad.toFixed(2));
298 }
299
300 function obtenerProbas (){
301     probabilidades_hormiga[0] = parseInt(document.getElementById("obrera")
302     ).value)/100;
303     probabilidades_hormiga[0] = parseFloat(probabilidades_hormiga
304     [0].toFixed(2));
305     probabilidades_hormiga[1] = parseInt(document.getElementById("soldado")
306     ).value)/100;
307     probabilidades_hormiga[1] = parseFloat(probabilidades_hormiga
308     [1].toFixed(2));
309     probabilidades_hormiga[2] = parseInt(document.getElementById("reina")
310     ).value)/100;
311     probabilidades_hormiga[2] = parseFloat(probabilidades_hormiga
312     [2].toFixed(2));
313 }
```

```

314 function limpiar(){
315     ctx.clearRect(0, 0, ctx.width, ctx.height);
316     console.clear();
317     generacion=0;
318     numero_blancos=0;
319     numero_negros=0;
320     numero_hormigas=0;
321     numero_reinas=0;
322     numero_soldados=0;
323     numero_oberas=0;
324     log="";
325     document.getElementById('generacion').innerHTML ="Generación: 0";
326 }
327
328 function iniciar(){
329     limpiar();
330     obtenerConfiguracion();
331     for (var i = 0; i < tam; i++){
332         for (var j=0;j<tam;j++){
333             hormigas[i][j]=[];
334             hormigas2[i][j]=[];
335             direccion[i][j]=[];
336             direccion2[i][j]=[];
337             vida[i][j]=[];
338             ctx.fillStyle=obtenerTipoCasilla(i,j);
339             x=0+j*escala;
340             y=0+i*escala;
341             coordenadas[i][j]=x+", "+y+", "+(x+escala)+", "+(y+escala);
342             ctx.fillRect(x,y, escala, escala);
343             ctx.stroke();
344             //console.log(numero_negros+" "+generacion);
345             log=numero_negros+" "+generacion+"\n";
346         }
347     }
348     console.log("Generación: "+generacion);
349     console.log("Número negras: "+numero_negros);
350     console.log("Número hormigas: "+numero_hormigas);
351     console.log("Número obreras: "+numero_oberas);
352     console.log("Número soldados: "+numero_soldados);
353     console.log("Número reinas: "+numero_reinas);
354 }
355
356 function obtenerTipoCasilla(i,j){
357     var num=Math.random();

```

```

358 if (num <= probabilidad) {
359     random=Math.floor ((Math.random() * 4) + 0);
360     numero_hormigas++;
361     array [i][j]=1;
362     array2[i][j]=1;
363     direccion [i][j].push (random);
364     direccion2 [i][j].push (random);
365     matar [i][j]=0;
366     obtenerTipoHormiga(i,j);
367     return colores_direcciones [random];
368 }
369 else {
370     numero_negros++;
371     array [i][j]=0;
372     array2[i][j]=0;
373     return color_negro;
374 }
375 }

376
377 function obtenerTipoHormiga(i,j){
378     var num=Math.random();
379     if (num<=probabilidades_hormiga [0]){
380         numero_oberras++;
381         hormigas [i][j].push (tipos_hormiga [0]);
382         hormigas2 [i][j].push (tipos_hormiga [0]);
383     }else if (num<=probabilidades_hormiga [0]+probabilidades_hormiga [1]){
384         numero_soldados++;
385         hormigas [i][j].push (tipos_hormiga [1]);
386         hormigas2 [i][j].push (tipos_hormiga [1]);
387     }else{
388         numero_reinas++;
389         hormigas [i][j].push (tipos_hormiga [2]);
390         hormigas2 [i][j].push (tipos_hormiga [2]);
391     }
392 }

393
394 function ejecutar(){
395     generacion_grafica.push(generacion);
396     hormigas_grafica.push(numero_hormigas);
397     obreras_grafica.push(numero_oberras);
398     soldados_grafica.push(numero_soldados);
399     reinas_grafica.push(numero_reinas);
400     generacion++;
401     for (var i=0; i<tam; i++){

```

```

402     for ( var j=0;j<tam ; j++){
403         evaluar(i , j );
404     }
405 }
406 console . log ( numero _ negros + " , " + generacion );
407 document . getElementById ( ' generacion ' ) . innerHTML = " Generación :
408     " + generacion ;
409 log += numero _ negros + " , " + generacion + "\ n " ;
410 array2= copiar ( array );
411 hormigas2= copiar2 ( hormigas );
412 direccion2= copiar2 ( direccion );
413 }
414 function continuar () {
415     intervalo=setInterval ( ejecutar , 10 );
416     intervalo2=setInterval ( graficar , 1000 );
417     intervalo3=setInterval ( graficar2 , 1000 );
418 }
419
420 function pausa () {
421     clearInterval ( intervalo );
422     clearInterval ( intervalo2 );
423     clearInterval ( intervalo3 );
424 }
425 function siguiente () {
426     ejecutar ();
427     graficar ();
428     graficar2 ();
429 }
430
431 function obtenerCoordenadas ( i , j ){
432     var texto=coordenadas [ i ][ j ];
433     return texto . split ( " , " );
434 }
435
436 function clickCanvas ( event ){
437     x=event . offsetX ;
438     y=event . offsetY ;
439 // console . log ( x + " , " + y );
440 for ( var i=0;i<tam ; i++){
441     for ( var j=0;j<tam ; j++){
442         var arr=obtenerCoordenadas ( i , j );
443         if ( x >= arr [ 0 ] && x <= arr [ 2 ] && y >= arr [ 1 ] && y <= arr
444 [ 3 ] ){
445             x=0+j * escala ;

```

```

445     y=0+i*escala ;
446     if( hormigas2 [ i ][ j ].length <1){
447         console . log ( "Soy :" +i+ " , "+j );
448         random=Math. floor (( Math. random() * 4) + 0);
449         // random=3;
450         numero_hormigas++;
451         numero_negros--;
452         hormigas [ i ][ j ]. push (1);
453         hormigas2 [ i ][ j ]. push (1);
454         direccion [ i ][ j ]. push (random);
455         direccion2 [ i ][ j ]. push (random);
456         matar [ i ][ j ]=0;
457         if (random==0){
458             //Norte -> Rojo
459             ctx . fillStyle=color_norte ;
460
461         } else if (random==1){
462             //Este -> Amarillo
463             ctx . fillStyle=color_este ;
464         } else if (random==2){
465             //Sur -> Azul
466             ctx . fillStyle=color_sur ;
467         } else{
468             //Oeste -> Verde
469             ctx . fillStyle=color_oeste ;
470         }
471     } else{
472         if (matar [ i ][ j ]==4){
473             hormigas [ i ][ j ]. pop ();
474             hormigas2 [ i ][ j ]. pop ();
475             direccion [ i ][ j ]. pop ();
476             direccion2 [ i ][ j ]. pop ();
477             ctx . fillStyle=color_negro ;
478             matar [ i ][ j ]=0;
479         } else if (direccion [ i ][ j ][0]==0){
480             direccion [ i ][ j ][0]=1;
481             direccion2 [ i ][ j ][0]=1;
482             ctx . fillStyle=color_este ;
483             matar [ i ][ j ]++;
484         } else if (direccion [ i ][ j ][0]==1){
485             direccion [ i ][ j ][0]=2;
486             direccion2 [ i ][ j ][0]=2;
487             ctx . fillStyle=color_sur ;
488             matar [ i ][ j ]++;
489         } else if (direccion [ i ][ j ][0]==2){

```

```

490     direccion [ i ][ j ][ 0 ] = 3 ;
491     direccion2 [ i ][ j ][ 0 ] = 3 ;
492     ctx . fillStyle = color_oeste ;
493     matar [ i ][ j ] ++ ;
494 } else if ( direccion [ i ][ j ][ 0 ] == 3 ) {
495     direccion [ i ][ j ][ 0 ] = 0 ;
496     direccion2 [ i ][ j ][ 0 ] = 0 ;
497     ctx . fillStyle = color_norte ;
498     matar [ i ][ j ] ++ ;
499 }
500 }
501 ctx . fillRect ( x , y , escala , escala ) ;
502 ctx . stroke () ;
503 break ;
504 }
505 }
506 }
507 }
508
509 function graficar () {
510     new Chart ( chart , {
511         type: 'line' ,
512         data: {
513             labels: generacion_grafica ,
514             datasets: [ {
515                 data: hormigas_grafica ,
516                 label: "Generación: "+generacion+ " Número de hormigas
517 : "+ numero_hormigas ,
518                 borderColor: "red" ,
519                 fill: false
520             }]
521         },
522         options: {
523             title: {
524                 display: true ,
525                 text: 'Cantidad de hormigas'
526             },
527             elements: {
528                 line: {
529                     tension: 0 , // disables bezier curves
530                 }
531             },
532             animation: {
533                 duration: 0 , // general animation time
534             }
535         }
536     })
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }

```

```

534     hover: {
535         animationDuration: 0, // duration of animations when
536         hovering an item
537     },
538     responsiveAnimationDuration: 0, // animation duration
539     after a resize
540     events: [], //Para que no se pueda interactuar con la
541     gráfica
542     maintainAspectRatio:true ,
543     responsive: true ,
544 }
545 function graficar2(){
546     new Chart(chart2, {
547         type: 'line',
548         data: {
549             labels: generacion_grafica ,
550             datasets: [
551                 {
552                     data: obreras_grafica ,
553                     label: "Obreras: "+numero_oberas ,
554                     borderColor: "red",
555                     fill: false
556                 },
557                 {
558                     data: soldados_grafica ,
559                     label: "Soldados: "+numero_soldados ,
560                     borderColor: "blue",
561                     fill: false
562                 },
563                 {
564                     data: reinas_grafica ,
565                     label: "Reinas: "+numero_reinas ,
566                     borderColor: "green",
567                     fill: false
568                 }
569             ]
570         },
571         options: {
572             title: {
573                 display: true ,
574                 text: 'Cantidad de hormigas'
575             },

```

```
576     elements: {
577       line: {
578         tension: 0, // disables bezier curves
579       }
580     },
581     animation: {
582       duration: 0, // general animation time
583     },
584     hover: {
585       animationDuration: 0, // duration of animations when
586       hovering an item
587     },
588     responsiveAnimationDuration: 0, // animation duration
589     after a resize
590     events: [] , //Para que no se pueda interactuar con la
591     gráfica
592     maintainAspectRatio:true ,
593     responsive: true ,
594   }
595 });
596 }
```

3.4. Pruebas y resultados

3.4.1. Hormiga de Langton Original



Figura 60: Interfaz gráfica del programa

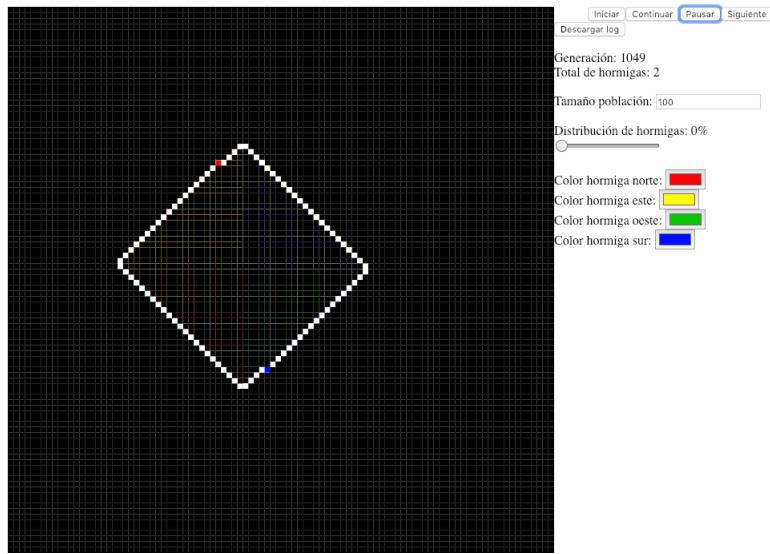


Figura 61: Dos hormigas agregadas por el usuario después de 1000 generaciones



Figura 62: Varias hormigas generadas mediante la distribución

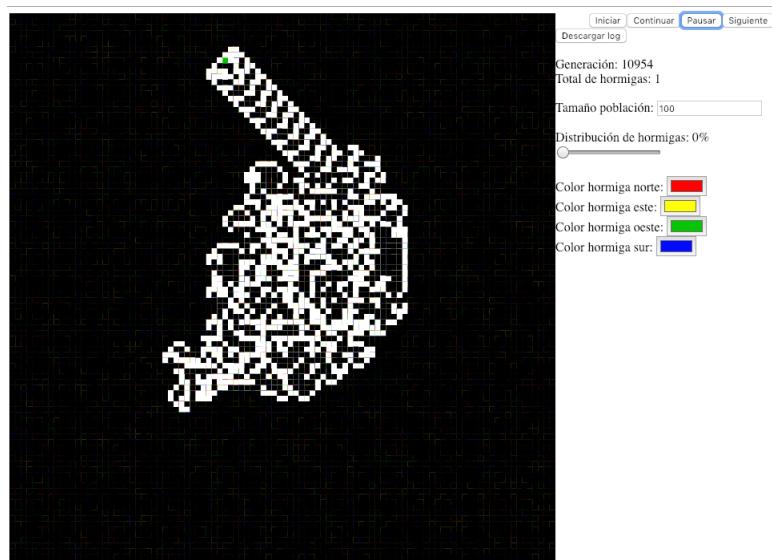


Figura 63: Hormiga repitiendo un patrón después de 10,000 generaciones

3.5. Hormiga de Langton Modificada



Figura 64: Colonia con probabilidades por defecto



Figura 65: Gráfica de la colonia anterior

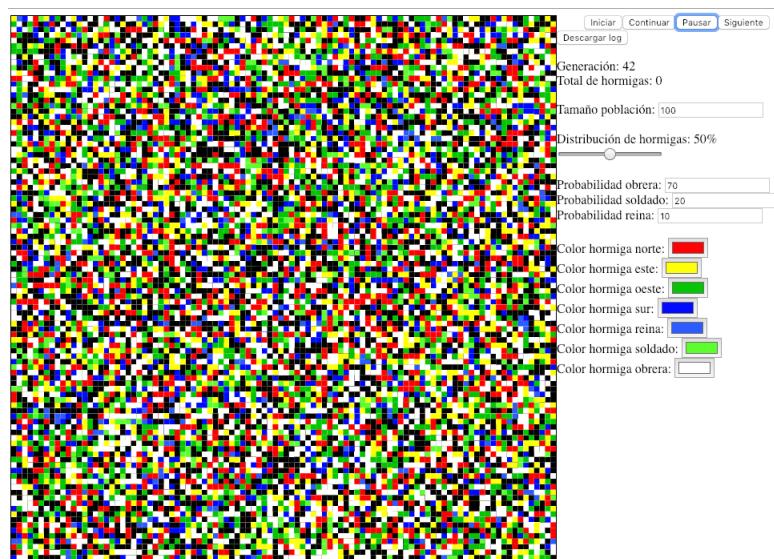


Figura 66: Probabilidades de 70,20 y 10

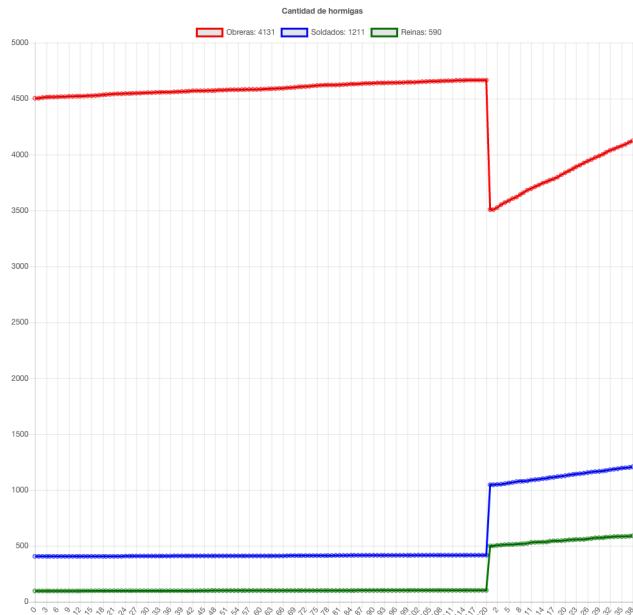


Figura 67: Gráfica de la colonia anterior

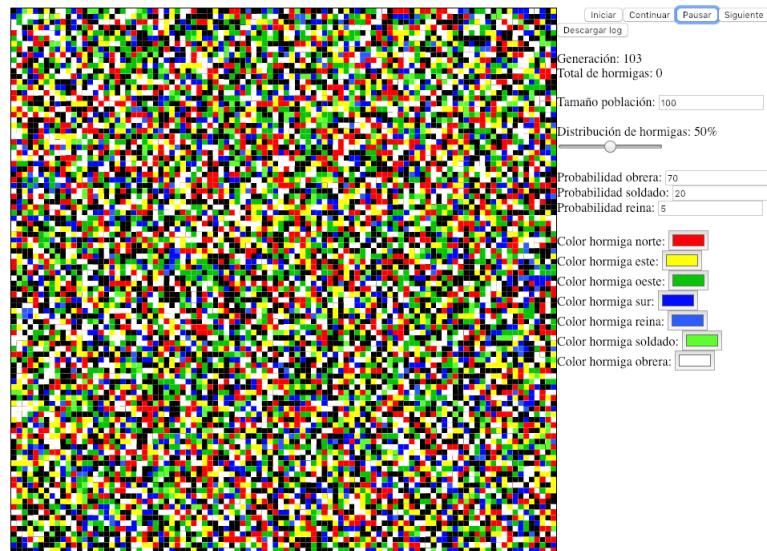


Figura 68: Probabilidades de 75, 20 y 5

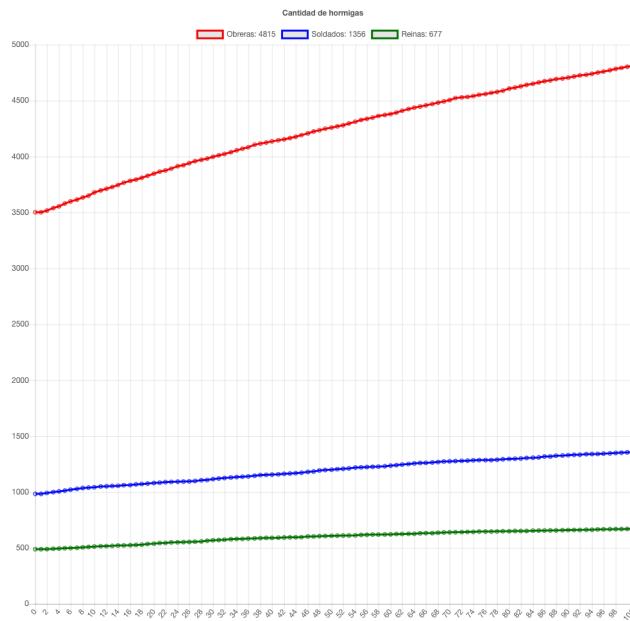


Figura 69: Gráfica de la colonia anterior

Como se puede observar, debido a las pocas reinas que existen, es muy complicado que nazcan nuevas hormigas.

3.6. Conclusiones

La hormiga de Langton es otro de ejemplo de un autómata celular bastante interesante, los comportamientos que se generan en este son bastante peculiares y complejos ya que estos tratan de describir toda una colonia. Respecto a la implementación no hubo muchos problemas, sin embargo, el rendimiento del programa no es tan bueno como se esperaba debido a que entre mayor sea el espacio a trabajar más lenta se vuelve la simulación.

El hecho de que la probabilidad de que la hormiga que se genere sea una reina es muy baja es otro problema ya que al no nacer tantas reinas la colonia no se puede mantener.

También cabe señalar que las hormigas tienden a realizar un patrón que se repite indeterminadamente, y mientras más hormigas estén cerca, más probable que se llegue a este patrón antes.

4. Grafos

4.1. Introducción

Este programa se encarga de generar los árboles que se crean con la regla de life y la regla de difusión una matriz de 2x2 hasta 7x7, sin embargo en este caso solo se alcanzó a calcular hasta 4x4, y debido a la capacidad de la computadora usada, se tiene en representación visual hasta 3x3, sin embargo, en el presente documento solo se muestra hasta 2x2, ya que las demás imágenes son demasiado grandes, y no es posible visualizar los grafos si no se les hace un acercamiento aparte en su formato común como imágenes. El programa fue desarrollado en Python y los archivos generados fueron guardados con la extensión mmd, lo cual implica que la herramienta usada para la representación visual de los grafos fue la librería de Javascript Mermaid cli.

4.2. Planteamiento de la práctica

Al ejecutar el programa se obtienen los scripts necesarios para crear las imágenes de cada matriz, cada script se debe de ejecutar para poder producir la imagen de un árbol.

4.3. Desarrollo

Este programa se desarrollo usando Python 3.7, además que se uso la herramienta para la creación de diagramas de flujo Mermaid.cli (la cual es una librería de javascript) para la creación de los grafos. Ya que es bastante eficiente y pudo generar imágenes hasta para 4x4, sin embargo, estas imágenes tienen que ser demasiado grandes, para que haciendo zoom en ellas se puedan apreciar los grafos.

Archivo: arboles.py

```
1 import numpy as np
2 import sys
3
4 dict_tipos = {
5     "life": 1,
6     "difusion": 2,
7 }
8
9
10 class Arboles:
11     def __init__(self, _tam=2, _tipo=dict_tipos["life"]):
12         self.tam = tam
13         self.tipo = tipo
14         self.vida = [2, 3, 3, 3]
15         self.difusion = [7, 7, 2, 2]
16         self.regla = self.difusion
17         self.longitud = self.tam * self.tam
18         self.max = 2 ** self.longitud
19         self.formato = "{:0{}b}".format(self.longitud)
20         if self.tipo == dict_tipos["life"]:
21             self.regla = self.vida
22
23     def obtener_siguiente(self, m):
24         nueva_matriz = m.copy()
25         for i in range(self.tam):
26             for j in range(self.tam):
27                 suma = self.revisar_vecinos(i, j, m)
28                 if m[i, j] == 1:
29                     if suma < self.regla[0] or suma > self.regla
30                         [1]:
31                         nueva_matriz[i, j] = 0
32                     else:
33                         if self.regla[2] <= suma <= self.regla[3]:
```

```

33                     nueva_matriz[i , j] = 1
34             return nueva_matriz
35
36     def revisar_vecinos(self , i , j , m):
37         vecinos = m[i - 1 , j - 1]
38         vecinos += m[i - 1 , j]
39         vecinos += m[i - 1 , (j + 1) % self.tam]
40         vecinos += m[i , (j + 1) % self.tam]
41         vecinos += m[(i + 1) % self.tam , (j + 1) % self.tam]
42         vecinos += m[(i + 1) % self.tam , j]
43         vecinos += m[(i + 1) % self.tam , j - 1]
44         vecinos += m[i , j - 1]
45     return vecinos
46
47     def numero_cadena(self , numero):
48         return self.formato.format(numero)
49
50     @staticmethod
51     def cadena_numero(cadena):
52         return int(cadena , 2)
53
54     def cadena_matriz(self , cadena):
55         m = np.zeros(shape=(self.tam , self.tam) , dtype=int)
56         k = 0
57         for i in range(self.tam):
58             for j in range(self.tam):
59                 if cadena[k] == '1':
60                     m[i , j] = 1
61                 k += 1
62         return m
63
64     def matriz_cadena(self , matriz):
65         cadena = "0" * self.longitud
66         k = 0
67         for i in range(self.tam):
68             for j in range(self.tam):
69                 if matriz[i , j] == 1:
70                     cadena[k] = "1"
71                 k += 1
72         return "".join(cadena)
73
74     def generar(self):
75         nombre_temporal = "difusion"
76         if self.tipo == dict_tipos["life"]:
77             nombre_temporal = "life"

```

```

78     nombre_archivo = "tam-{}-{}".format(self.tam,
79     nombre_temporal)
80     archivo = open("{}.mmd".format(nombre_archivo), "w")
81
82     archivo.write("graph TD;\n")
83     for i in range(self.max):
84         cadena = self.numero_cadena(i)
85         m = self.cadena_matriz(cadena)
86         m_sig = self.obtener_siguiente(m)
87         cadena_sig = self.matriz_cadena(m_sig)
88         if i == self.max-1:
89             archivo.write("\t"+ "{}-->{};\n".format(cadena,
90             cadena_sig))
91         else:
92             archivo.write("\t"+ "{}-->{};\n".format(cadena,
93             cadena_sig))
94     archivo.close()
95
96 tam = int(sys.argv[1])
97 tipo = int(sys.argv[2])
98 life = Arboles(tam, tipo)
99 life.gerarar()

```

Una vez obtenida una salida, está misma se usaba con el siguiente comando en Mermaid:

mmdc -i input.mmd -o output.png

4.4. Pruebas y resultados

A continuación se muestran los resultados obtenidos para 2x2.

Primeramente, se muestran dos archivos obtenidos como resultado de ejecutar el programa arboles.py, como se puede observar en ambos casos, el archivo ya se encuentra acomodado para ser usado con Mermaid.cli.

Archivo generado: tam-2-life.mmd

```
1 graph TD;
2   0000-->0000;
3   0001-->0000;
4   0010-->0000;
5   0011-->0011;
6   0100-->0000;
7   0101-->0101;
8   0110-->0000;
9   0111-->0000;
10  1000-->0000;
11  1001-->0000;
12  1010-->1010;
13  1011-->0000;
14  1100-->1100;
15  1101-->0000;
16  1110-->0000;
17  1111-->0000;
```

Archivo generado: tam-2-diffusion.mmd:

```
1 graph TD;
2   0000-->0000;
3   0001-->0110;
4   0010-->1001;
5   0011-->0000;
6   0100-->1001;
7   0101-->0000;
8   0110-->0000;
9   0111-->0000;
10  1000-->0110;
11  1001-->0000;
12  1010-->0000;
13  1011-->0000;
14  1100-->0000;
15  1101-->0000;
```

```
16 1110-->0000;  
17 1111-->0000;
```

Posteriormente, las figuras 70 y 71 muestran los árboles obtenidos después de usar los archivos anteriormente mostrados con Mermaid.cli.

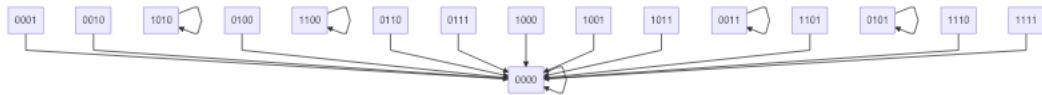


Figura 70: Árbol obtenido para 2x2 Life

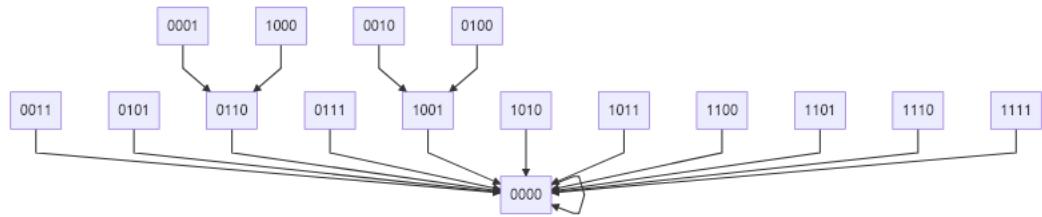


Figura 71: Árbol obtenido para 2x2 Difusión

4.5. Conclusiones

A pesar de que solo se pudo observar el comportamiento en matrices de 2x2 y 3x3 visualmente, no fue tan complicado determinar el comportamiento de life y de difusión.

En difusión los árboles se encuentran más concentrados en menos grupos mientras que en life se generan muchos árboles lo cual es interesante considerando que en la regla de difusión la población tiende a crecer. Por lo que existe una relación en estas dos características.

Se podría decir que ya que de una configuración en especifica de la regla de difusión esta crece por lo que pasa por más configuraciones distintas que si se trabajara la misma configuración en life y es por esto que la cantidad de árboles es menor en la regla de difusión que en la de life.

5. Autómata Celular con Memoria

5.1. Introducción

Los autómatas celulares(AC) surgen en la década de 1940 con John Von Neumann, que intentaba modelar una máquina que fuera capaz de auto-rePLICARSE, llegando así a un modelo matemático de dicha maquina con reglas complicadas sobre una red rectangular. Inicialmente fueron interpretados como conjunto de células que crecían, se reproducían y morían a medida que pasaba el tiempo. A esta similitud con el crecimiento de las células se le debe su nombre.[2]

Un autómata celular se caracteriza por contar con los siguientes elementos:

- Arreglo regular. Ya sea un plano de dos dimensiones o un espacio n-dimensional, este es el espacio de evoluciones, y cada división homogénea del arreglo es llamada célula.
- Conjunto de estados. Es finito y cada elemento o célula del arreglo toma un valor de este conjunto de estados. También se denomina alfabeto. Puede ser expresado en valores o colores.
- configuración inicial. Consiste en asignar un estado a cada una de las células del espacio de evolución inicial del sistema.
- Vecindades. Define el conjunto contiguo de células y posición relativa respecto a cada una de ellas. A cada vecindad diferente corresponde un elemento del conjunto de estados.
- Función local. Es la regla de evolución que determina el comportamiento del A. C. Se conforma de una célula central y sus vecindades. Define como debe cambiar de estado cada célula dependiendo de los estados anteriores de sus vecindades. Puede ser una expresión algebraica o un grupo de ecuaciones.

5.2. Planteamiento de la práctica

Este programa implementa la simulación de un autómata celular. Como fue expuesto en la sección 2 de este mismo documento, con la diferencia

que este autómata celular implementa una nueva característica, la de tener “memoria”.

Se cuenta con la opción para utilizar una matriz auxiliar junto con una función Φ para realizar el cálculo de las matrices a través de las generaciones futuras. Esta opción consiste en lo siguiente.

1. Se elige una función Φ la cual puede ser de paridad, máximo o mínimo. También, se asigna un valor de τ el cual debe de ir de 3 a 8.
2. Despues de τ iteraciones se utiliza la función Φ para obtener una matriz auxiliar, la función Φ utiliza como parámetros los valores de las τ matrices anteriores.
3. A la matriz auxiliar se le debe de aplicar la regla del autómata celular que se ha estado utilizado para calcular las iteraciones por lo que ahora tendremos una nueva matriz.
4. Esto se repite las veces que se deseé.

5.3. Desarrollo

El desarrollo de este programa, se llevo acabo usando Javascript, debido a que el entorno web y del navegador permiten que se puedan desarrollar interfaces para usuario de una forma bastante eficiente. Además, los programas que requieren de usar animaciones o que hacen un uso extensivo de dibujar en un canvas (como es el caso), se ven beneficiados del entorno web que permite todo se ejecute más rápido.

La parte de presentación para el usuario, se encuentra en el siguiente archivo html.

Archivo: gol-memoria.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Game of life Memory</title>
6     <!-- <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous"> -->
7     <link rel="stylesheet" type="text/css" href="estilos.css">
8   </head>
9   <body>
10    <div id="contenedor">
11      <!-- <canvas id="myCanvas" width="1260" height="680" style="border:1px solid #000000;"> -->
12      <div>
13        <canvas onclick="clickCanvas(event)" id="myCanvas" width="680" height="680" style="border:1px solid #000000;">
14        </canvas>
15      </div>
16      <div id="controles">
17        <div id="botones">
18          <!-- <div> -->
19          <button onclick="iniciar()">Iniciar</button>
20          <button onclick="continuar()">Continuar</button>
21          <button onclick="pausa()">Pausar</button>
22          <button onclick="siguiente()">Siguiente</button>
23          <!-- </div> -->
24        </div>
25      <div>
```

```

26      <button id="descargar" onclick="descargar()">Descargar
27  log</button>
28  </div><br>
29  <div>
30      <label id="generacion">Generación: 0</label>
31  </div><br>
32  <div>
33      <label for="tam">Tamaño población:</label>
34      <input type="number" id="tam" value="100"><br>
35  </div><br>
36  <div>
37      <label for="b1">B1:</label>
38      <input type="number" id="b1" value="2"><br>
39      <label for="b2">B2:</label>
40      <input type="number" id="b2" value="3"><br>
41      <label for="s1">S1:</label>
42      <input type="number" id="s1" value="3"><br>
43      <label for="s2">S2:</label>
44      <input type="number" id="s2" value="3"><br>
45  </div><br>
46  <div>
47      <label for="distribucion">
48          Distribución de vivos:
49          <span id="distribucion_valor">50</span>%
50      </label>
51      <input type="range" min="0" max="100" value="50" id="distribucion">
52  </div><br>
53  <div>
54      <label for="funcion">Función a utilizar:</label>
55      <select name="funcion" id="funcion">
56          <option value="0">Máximo</option>
57          <option value="1">Mínimo</option>
58          <option value="2">Paridad</option>
59      </select>
60  </div><br>
61  <div>
62      <label for="tau">Tau:</label>
63      <input type="number" placeholder="tau" id="tau" value="3"/>
64  </div><br>
65  <div>
66      <label for="color_vivo">Color de vivos:</label>
      <input type="color" value="#66ff66" id="color_vivo"><br>

```

```

67         <label for="color_muerto">Color de muertos:</label>
68         <input type="color" value="#FF0000" id="color_muerto"
69     ><br>
70         </div>
71     </div>
72     </div>
73     <canvas id="myChart" width="500" height="500"></canvas><br>
74     <div>Matriz auxiliar:</div>
75     <div>
76         <canvas id="myCanvas2" width="680" height="680" style="border:1px solid #000000;">
77             </canvas>
78         </div>
79     <script src="Chart.min.js"></script>
80     <script type="text/javascript" src="gol-memoria.js"></script>
81 </body>
82 </html>

```

La lógica con la que funciona el programa, se encuentra en el siguiente archivo de Javascript.

Archivo: gol-memoria.js

```

1 var c=document.getElementById("myCanvas");
2 var c2=document.getElementById("myCanvas2");
3 var chart = document.getElementById("myChart");
4 var longitud=680;
5 var tam=100;
6 var escala=longitud/tam;
7 var ctx=c.getContext("2d");
8 var ctx2=c2.getContext("2d");
9 ctx.lineWidth = "0.1";
10 ctx2.lineWidth = "0.1";
11 var random;
12 var x,y;
13 var regla1=2;
14 var regla2=3;
15 var regla3=3;
16 var regla4=3;
17 var generacion=0;
18 var contador=0;
19 var tau=3;
20 var numero_vivos=0;
21 var numero_muertos=0;
22 var color_vivo="#66ff66";

```

```

23 var color_muerto="#FF0000";
24 var vecindad=new Array(8);
25 var intervalo;
26 var log;
27 var generacion_grafica=[];
28 var vivos_grafica=[];
29 var total_vivos=0;
30 var array;
31 var array2;
32 var coordenadas;
33 //0-> Máximo
34 //1-> Mínimo
35 //2-> Paridad
36 var tipo_regla=[0,1,2];
37 var historico;
38 var distribucion=50;
39
40 //Función para descargar
41 function descargar(){
42     var fileContents = log;
43     var fileName = "log.txt";
44     var pp = document.createElement('a');
45     pp.setAttribute('href', 'data:text/plain;charset=utf-8,' +
        encodeURIComponent(fileContents));
46     pp.setAttribute('download', fileName);
47     pp.click();
48 }
49
50 //Modificar valor del slider en pantalla
51 document.getElementById("distribucion_valor").innerHTML =
    document.getElementById("distribucion").value;
52 document.getElementById("distribucion").oninput = function(e) {
53     e.preventDefault();
54     document.getElementById("distribucion_valor").innerHTML = this
        .value;
55 }
56
57 function Create2DArray(rows) {
58     var arr = new Array(rows);
59     for (var i=0;i<rows;i++) {
60         arr[i] = new Array(rows);
61     }
62     return arr;
63 }
64

```

```

65 function evaluar(i,j){
66     var estado=array2[i][j];
67     var cantidad_vivos=0;
68     var cantidad_muertos=0;
69     if (i==0){
70         vecindad[0]=array2[tam-1][j]; //superior
71         vecindad[1]=array2[i+1][j]; //inferior
72         if (j==0){
73             vecindad[2]=array2[tam-1][tam-1]; //superior izquierdo
74             vecindad[3]=array2[tam-1][j+1]; //superior derecho
75             vecindad[4]=array2[i][tam-1]; //izquierdo
76             vecindad[5]=array2[i][j+1]; //derecho
77             vecindad[6]=array2[i+1][tam-1]; //inferior izquierdo
78             vecindad[7]=array2[i+1][j+1]; //inferior derecho
79         }
80         else if (j==tam-1){
81             vecindad[2]=array2[tam-1][j-1]; //superior izquierdo
82             vecindad[3]=array2[tam-1][0]; //superior derecho
83             vecindad[4]=array2[i][j-1]; //izquierdo
84             vecindad[5]=array2[i][0]; //derecho
85             vecindad[6]=array2[i+1][j-1]; //inferior izquierdo
86             vecindad[7]=array2[i+1][0]; //inferior derecho
87         }
88         else{
89             vecindad[2]=array2[tam-1][j-1]; //superior izquierdo
90             vecindad[3]=array2[tam-1][j+1]; //superior derecho
91             vecindad[4]=array2[i][j-1]; //izquierdo
92             vecindad[5]=array2[i][j+1]; //derecho
93             vecindad[6]=array2[i+1][j-1]; //inferior izquierdo
94             vecindad[7]=array2[i+1][j+1]; //inferior derecho
95         }
96     }
97     else if (i==tam-1){
98         vecindad[0]=array2[i-1][j]; //superior
99         vecindad[1]=array2[0][j]; //inferior
100        if (j==0){
101            vecindad[2]=array2[i-1][tam-1]; //superior izquierdo
102            vecindad[3]=array2[i-1][j+1]; //superior derecho
103            vecindad[4]=array2[i][tam-1]; //izquierdo
104            vecindad[5]=array2[i][j+1]; //derecho
105            vecindad[6]=array2[0][tam-1]; //inferior izquierdo
106            vecindad[7]=array2[0][j+1]; //inferior derecho
107        }
108        else if (j==tam-1){
109            vecindad[2]=array2[i-1][j-1]; //superior izquierdo

```

```

110     vecindad[3]=array2[i-1][0];//superior derecho
111     vecindad[4]=array2[i][j-1];//izquierdo
112     vecindad[5]=array2[i][0];//derecho
113     vecindad[6]=array2[0][j-1];//inferior izquierdo
114     vecindad[7]=array2[0][0];//inferior derecho
115 }
116 else{
117     vecindad[2]=array2[i-1][j-1];//superior izquierdo
118     vecindad[3]=array2[i-1][j+1];//superior derecho
119     vecindad[4]=array2[i][j-1];//izquierdo
120     vecindad[5]=array2[i][j+1];//derecho
121     vecindad[6]=array2[0][j-1];//inferior izquierdo
122     vecindad[7]=array2[0][j+1];//inferior derecho
123 }
124 }
125 else{
126     vecindad[0]=array2[i-1][j];//superior
127     vecindad[1]=array2[i+1][j];//inferior
128     if (j==0){
129         vecindad[2]=array2[i-1][tam-1];//superior izquierdo
130         vecindad[3]=array2[i-1][j+1];//superior derecho
131         vecindad[4]=array2[i][tam-1];//izquierdo
132         vecindad[5]=array2[i][j+1];//derecho
133         vecindad[6]=array2[i+1][tam-1];//inferior izquierdo
134         vecindad[7]=array2[i+1][j+1];//inferior derecho
135     }
136     else if (j==tam-1){
137         vecindad[2]=array2[i-1][j-1];//superior izquierdo
138         vecindad[3]=array2[i-1][0];//superior derecho
139         vecindad[4]=array2[i][j-1];//izquierdo
140         vecindad[5]=array2[i][0];//derecho
141         vecindad[6]=array2[i+1][j-1];//inferior izquierdo
142         vecindad[7]=array2[i+1][0];//inferior derecho
143     }
144     else{
145         vecindad[2]=array2[i-1][j-1];//superior izquierdo
146         vecindad[3]=array2[i-1][j+1];//superior derecho
147         vecindad[4]=array2[i][j-1];//izquierdo
148         vecindad[5]=array2[i][j+1];//derecho
149         vecindad[6]=array2[i+1][j-1];//inferior izquierdo
150         vecindad[7]=array2[i+1][j+1];//inferior derecho
151     }
152 }
153 for ( var k=0;k<8;k++){
154     if (vecindad[k]==1){

```

```

155     cantidad_vivos++;
156 } else{
157     cantidad_muertos++;
158 }
159 }
160 if (estado==1){
161     if (!((cantidad_vivos >= regla1) && (cantidad_vivos <=
162     regla2))){
163         array [ i ][ j ]=0;
164         numero_vivos--;
165         numero_muertos++;
166         ctx . fillStyle=color_muerto ;
167         ctx . fillRect (0+(j*(escala)),0+(i*(escala)),escala , escala );
168     }
169 } else{
170     if ((cantidad_vivos >= regla3) && (cantidad_vivos <= regla4)
171 ){
172         array [ i ][ j ]=1;
173         numero_vivos++;
174         numero_muertos--;
175         ctx . fillStyle=color_vivo ;
176         ctx . fillRect (0+(j*(escala)),0+(i*(escala)),escala , escala );
177     }
178 }
179 ctx . stroke ();
180 }
181 function copiar (array){
182     var aux=Create2DArray (tam);
183     for (var i=0; i<tam; i++){
184         for (var j=0;j<tam; j++){
185             aux [ i ][ j ]=array [ i ][ j ];
186         }
187     return aux;
188 }
189 }
190 function obtenerTamano (){
191     tam = parseInt (document . getElementById ("tam") . value );
192 //console . log (tam);
193     array=Create2DArray (tam);
194     array2=Create2DArray (tam);
195     coordenadas=Create2DArray (tam);
196     escala=longitud/tam;
197 }

```

```

198
199 function obtenerRegla(){
200     regla1 = parseInt(document.getElementById("b1").value);
201     regla2 = parseInt(document.getElementById("b2").value);
202     regla3 = parseInt(document.getElementById("s1").value);
203     regla4 = parseInt(document.getElementById("s2").value);
204 }
205
206 function obtenerDistribucion(){
207     distribucion = parseInt(document.getElementById("distribucion")
208         ).value)/100;
209     distribucion = parseFloat(distribucion.toFixed(2));
210 }
211
212 function obtenerFuncion(){
213     return parseInt(document.getElementById("funcion").value);
214 }
215
216 function obtenerTau(){
217     tau=parseInt(document.getElementById("tau").value);
218     historico=new Array(tau);
219     for(var i=0;i<tau;i++){
220         historico[i]=Create2DArray(tam);
221     }
222 }
223
224 function obtenerColores(){
225     color_vivo=document.getElementById("color_vivo").value;
226     color_muerto=document.getElementById("color_muerto").value;
227 }
228
229 function obtenerConfiguracion(){
230     obtenerTamano();
231     obtenerRegla();
232     obtenerDistribucion();
233     obtenerFuncion();
234     obtenerTau();
235     obtenerColores();
236 }
237
238 function limpiar(){
239     ctx.clearRect(0, 0, ctx.width, ctx.height);
240     ctx2.clearRect(0, 0, ctx2.width, ctx2.height);
241     console.clear();
242     new Chart(chart,{});

```

```

242 numero_vivos=0;
243 numero_muertos=0;
244 generacion=0;
245 document.getElementById( 'generacion' ).innerHTML ="Generación :
246 0";
247 contador=0;
248 generacion_grafica=[];
249 vivos_grafica=[];
250 log="";
251 }
252 function iniciar (){
253 limpiar();
254 obtenerConfiguracion();
255 for (var i = 0; i < tam; i++) {
256 for(var j=0;j<tam;j++){
257 //Checamos la distribución
258 if (Math.random() < distribucion){
259 array [ i ][ j ]=1;
260 array2 [ i ][ j ]=1;
261 ctx.fillStyle=color_vivo ;
262 numero_vivos++;
263 }
264 else{
265 array [ i ][ j ]=0;
266 array2 [ i ][ j ]=0;
267 ctx.fillStyle=color_muerto ;
268 numero_muertos++;
269 }
270 x=0+j*escala ;
271 y=0+i*escala ;
272 coordenadas [ i ][ j ]=x+", "+y+", "+(x+escala )+", "+(y+escala );
273 ctx.fillRect(0+(j*( escala )),0+(i*( escala )),escala ,escala );
274 ctx.stroke();
275 }
276 }
277 generacion_grafica.push(generacion);
278 vivos_grafica.push(numero_vivos);
279 console.log( numero_vivos+", "+generacion );
280 log=numero_vivos+", "+generacion+"\n";
281 historico [ contador]=copiar (array2 );
282 contador++;
283 }
284
285 function dibujarMatrizAuxiliar(i ,j ,valor ){

```

```

286 if (valor==1) {
287     ctx2.fillStyle=color_vivo;
288     numero_vivos++;
289 } else{
290     ctx2.fillStyle=color_muerto;
291     numero_muertos++;
292 }
293 ctx2.fillRect(0+(j*(escala)),0+(i*(escala)),escala,escala);
294 ctx2.stroke();
295 }
296
297 function generarMatrizAuxiliar(){
298     var funcion=obtenerFuncion();
299     var valor;
300     for(var i=0;i<tam;i++){
301         for(var j=0;j<tam;j++){
302             if(funcion==0){//Máximo
303                 var max=0;
304                 for(var k=0;k<tau;k++){
305                     max+=historico[k][i][j];
306                 }
307                 if(max>0){
308                     valor=1;
309                 } else{
310                     valor=0;
311                 }
312             } else if(funcion==1){//Mínimo
313                 var min=0;
314                 for(var k=0;k<tau;k++){
315                     min+=historico[k][i][j];
316                 }
317                 if(min==0){
318                     valor=1;
319                 } else{
320                     valor=0;
321                 }
322             } else if(funcion==2){//Paridad
323                 var aux=0;
324                 for(var k=0;k<tau;k++){
325                     //console.log(historico[k][i][j]);
326                     //console.log("k: "+k);
327                     aux+=historico[k][i][j];
328                 }
329                 //console.log("aux: "+aux);
330                 if(aux % 2 == 0){

```

```

331         valor=1;
332     } else{
333         valor=0;
334     }
335 }
336 dibujarMatrizAuxiliar(i ,j ,valor );
337 array [ i ][ j]=valor ;
338 }
339 }
340 }
341
342 function ejecutar (){
343     obtenerRegla();
344     obtenerColores();
345     if (contador==tau){
346         contador=0;
347         numero_vivos=0;
348         numero_muertos=0;
349         console.log("Voy a generar la matriz auxiliar");
350         generarMatrizAuxiliar();
351         array2=copiar(array);
352         console.log("Vivos en auxiliar "+numero_vivos);
353     }
354     for (var i=0; i<tam; i++){
355         for (var j=0;j<tam; j++){
356             evaluar(i ,j );
357         }
358     }
359     array2=copiar (array);
360     historico [ contador]=copiar (array2);
361     contador++;
362     generacion++;
363     generacion_grafica .push(generacion);
364     vivos_grafica .push(numero_vivos);
365     total_vivos+=numero_vivos;
366     console.log (numero_vivos+", "+generacion);
367     document.getElementById( 'generacion' ).innerHTML ="Generación:
368         "+generacion;
369     log+=numero_vivos+", "+generacion+"\n";
370 }
371 function continuar(){
372     intervalo=setInterval(ejecutar , 100);
373     intervalo2=setInterval(graficar , 1000);
374 }

```

```

375
376 function pausa(){
377     clearInterval(intervalo);
378     clearInterval(intervalo2);
379 }
380 function siguiente(){
381     ejecutar();
382     graficar();
383 }
384
385 function graficar(){
386     new Chart(chart, {
387         type: 'line',
388         data: {
389             labels: generacion_grafica,
390             datasets: [{}]
391             data: vivos_grafica,
392             label: "Generación: "+generacion+" Número de vivos: "
393             + numero_vivos+" Promedio: "+(total_vivos/generacion)+""
394             Densidad: "+(total_vivos/generacion)/(tam*tam),
395             borderColor: "red",
396             fill: false
397         }]
398     },
399     options: {
400         title: {
401             display: true,
402             text: 'Gráfica de vivos'
403         },
404         elements: {
405             line: {
406                 tension: 0, // disables bezier curves
407             }
408         },
409         animation: {
410             duration: 0, // general animation time
411         },
412         hover: {
413             animationDuration: 0, // duration of animations when
414             hovering an item
415         },
416         responsiveAnimationDuration: 0, // animation duration
417             after a resize
418         events: [] , //Para que no se pueda interactuar con la
419             gráfica

```

```

415     maintainAspectRatio:true ,
416     responsive: true ,
417   }
418 );
419 }
420
421 function obtenerCoordenadas(i ,j ){
422   var texto=coordenadas[i ][j ];
423   return texto .split(",");
424 }
425
426 function clickCanvas( event){
427   x=event .offsetX;
428   y=event .offsetY;
429   // console.log(x + "," +y);
430   for( var i=0;i<tam ; i++){
431     for( var j=0;j<tam ; j++){
432       var arr=obtenerCoordenadas(i ,j );
433       if(x >= arr [0] && x <= arr [2] && y >= arr [1] && y <= arr [3]){
434         x=0+j *escala ;
435         y=0+i *escala ;
436         if( array [i ][j ]==0){
437           array [i ][j ]=1;
438           array2 [i ][j ]=1;
439           historico [contador -1][i ][j ]=1;
440           numero_vivos++;
441           numero_muertos--;
442           ctx .fillStyle=color_vivo ;
443         }else{
444           array [i ][j ]=0;
445           array2 [i ][j ]=0;
446           historico [contador -1][i ][j ]=0;
447           numero_vivos--;
448           numero_muertos++;
449           ctx .fillStyle=color_muerto ;
450         }
451         ctx .fillRect (x,y, escala , escala );
452         ctx .stroke();
453         //console.log("Soy: i:" +i+ " j:" +j);
454         console .log("Número vivos: "+numero_vivos);
455         break ;
456       }
457     }
458 }

```

```
459 }  
460  
461 // iniciar();
```

5.4. Pruebas y resultados

Para probar el funcionamiento del programa se utilizaron diferentes reglas con diferentes densidades de población, para observar el comportamiento de los mismos. Es importante señalar que todas las pruebas se hicieron con un $\tau = 4$ y con diferentes funciones Φ además de solo iterar cerca de 100 veces.

5.4.1. Regla: 2 7 4 6. Densidad: 20 %

Como se puede observar en la figura 72 esta regla termina por llenar la mayor parte espacio con una malla de unos y ceros.

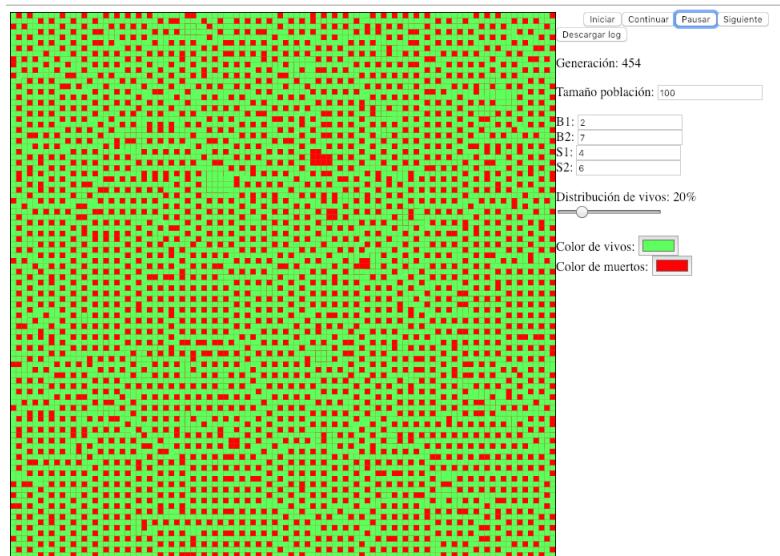


Figura 72: Resultado de iterar 100 veces sin memoria

Como se puede observar en la figura 73, el comportamiento de este autómata usando la función de máximo es muy similar a su comportamiento normal.

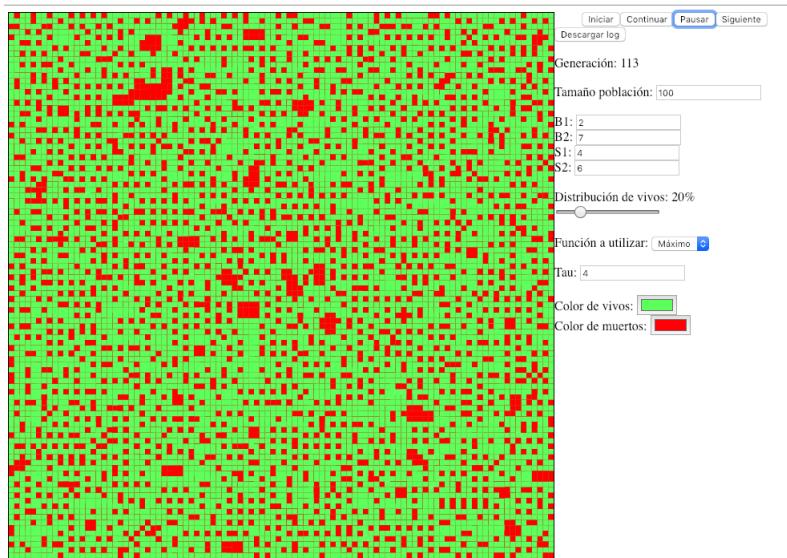


Figura 73: Resultado de iterar usando función de máximo

De igual manera, la matriz auxiliar se comporta similar, sin embargo, se distingue que está tiene menos células muertas.

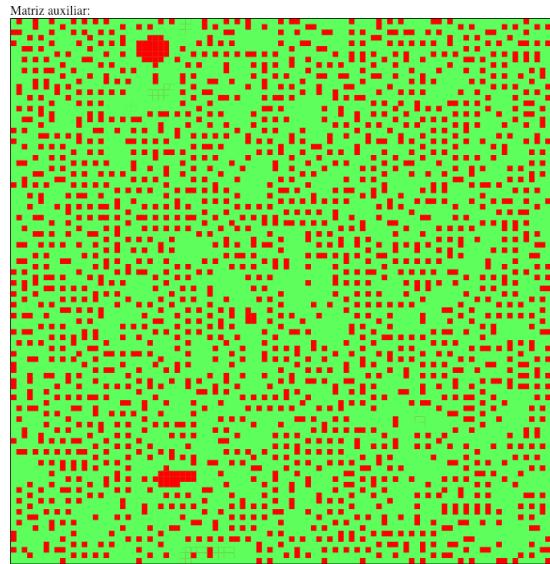


Figura 74: Matriz auxiliar

Sin embargo, en la figura 75 podemos observar como es que la función de mínimo se comporta distinto, ya que pareciera que esta generando algunos patrones dentro de la malla que parecieran repetirse.

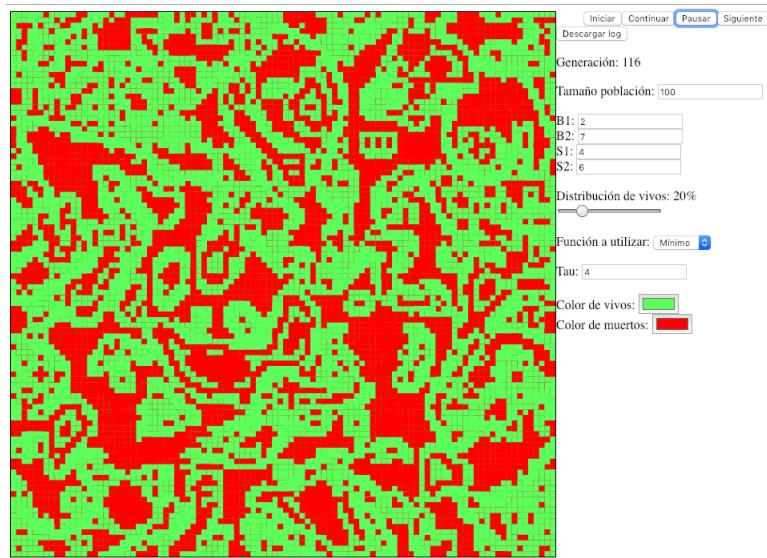


Figura 75: Resultado de iterar usando función de mínimo

La matriz auxiliar de esta, solo aparenta estar cambiando de colores.

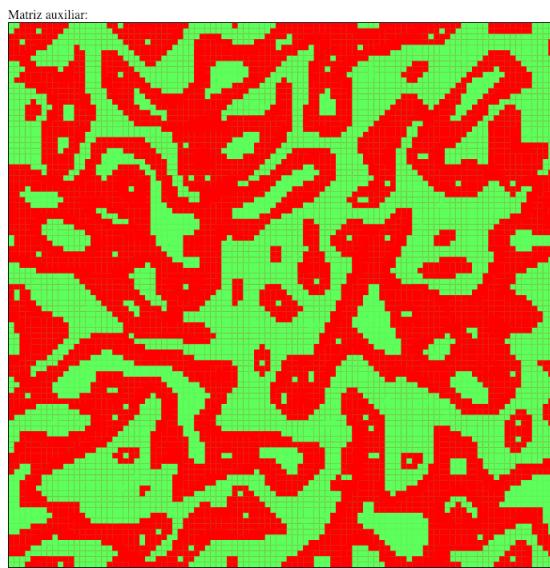


Figura 76: Matriz auxiliar

Tiene un comportamiento similar a la de máximo, sin embargo, tiene más “huecos” que la otra.

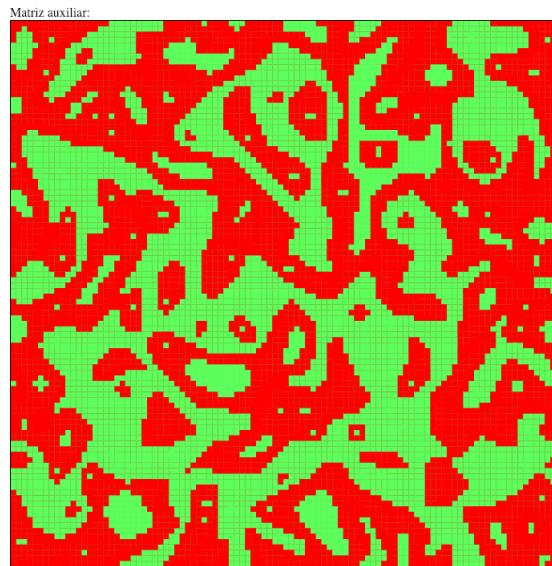


Figura 77: Resultado de iterar usando función de paridad

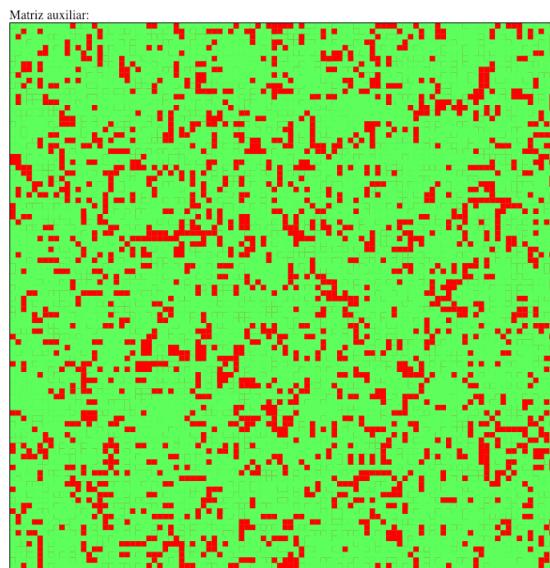


Figura 78: Matriz auxiliar

5.4.2. Regla: 3 6 3 4 (Cruz en el centro)

Como se puede observar en la figura 79 esta regla permite formar mosaicos los cuales van creciendo de tamaño, para lograr esto, la distribución se coloco en cero y se agregaron 7 cuadros en forma de cruz al centro (esto sin usar memoria).

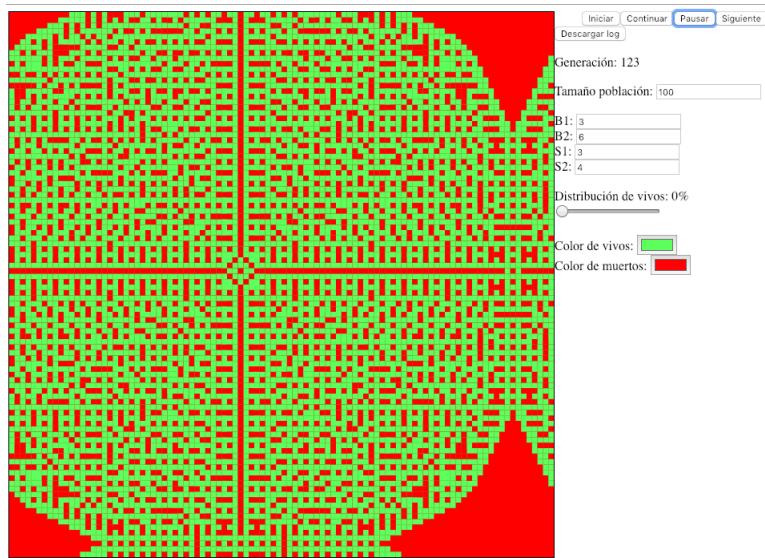


Figura 79: Resultado de poner una cruz al centro

En la figura 80, el comportamiento de este autómata usando la función de mínimo es muy similar a su comportamiento sin memoria, sin embargo, el mosaico formado es distinto.



Figura 80: Resultado de iterar usando función de mínimo

De igual manera, la matriz auxiliar forma mosaicos, pero con al diferencia, que las células que están vivas en máximo están muertas en auxiliar y viceversa.

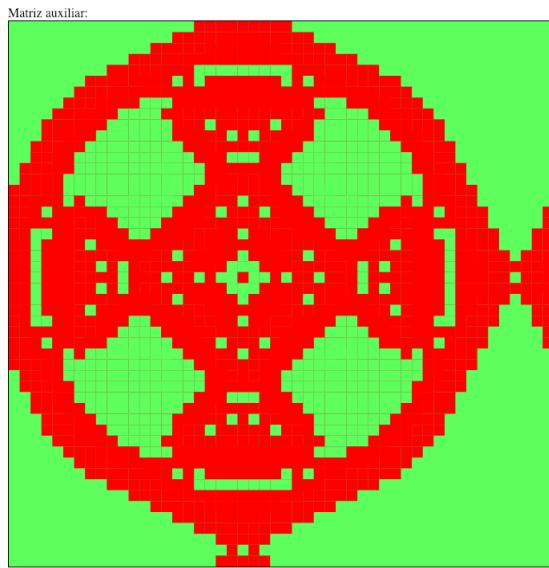


Figura 81: Matriz auxiliar

Al aplica la función de máximo el resultado es el mismo que al usar mínimo.



Figura 82: Resultado de iterar usando función de máximo

Sin embargo, la matriz auxiliar cambia y no es igual a la de mínimo, es otro mosaico distinto.

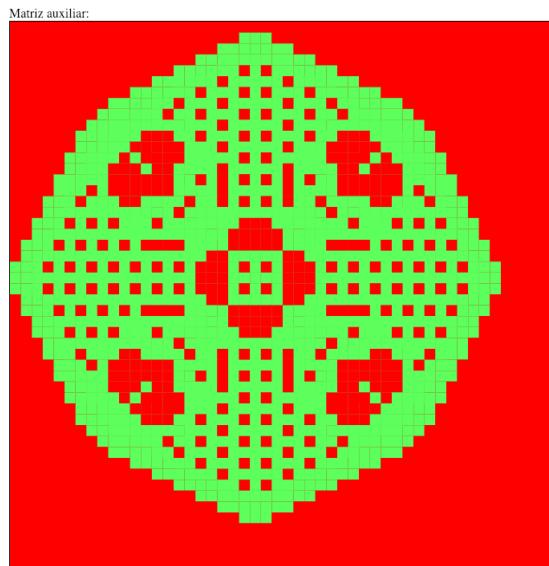


Figura 83: Matriz auxiliar

Finalmente, al aplica la función paridad, podemos observar otro mosaico muy distinto a los anteriores, ya que está tarda más generaciones que los demás en expandirse.

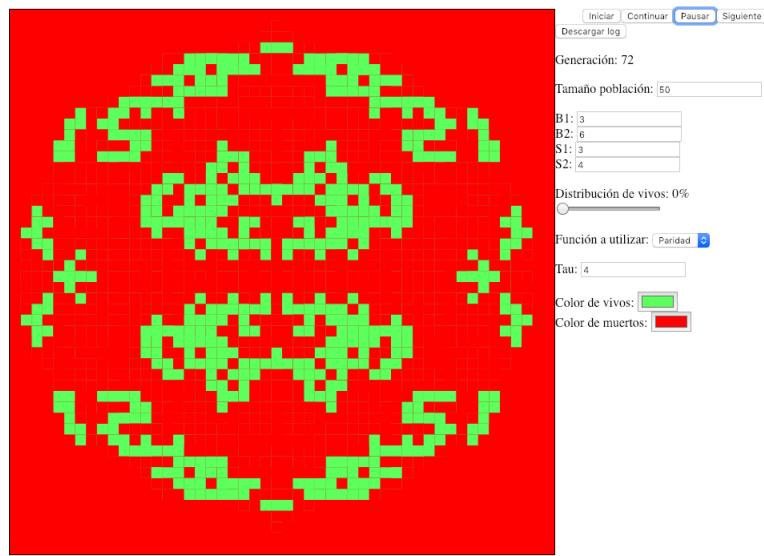


Figura 84: Resultado de iterar usando función de paridad

Al igual que en mínimo, la matriz auxiliar de la función paridad intercambia los colores con esta.

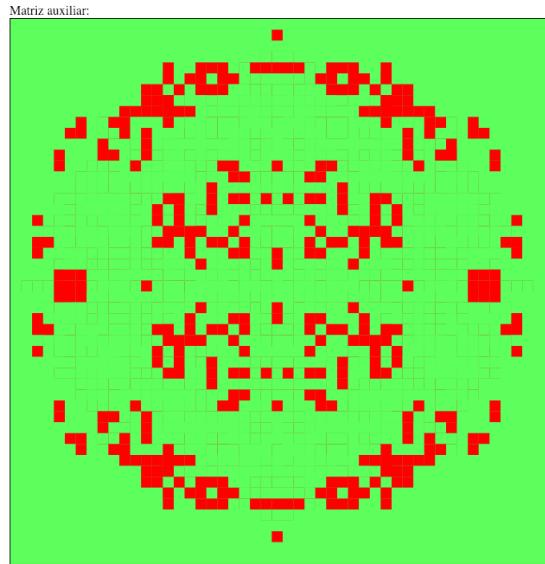


Figura 85: Matriz auxiliar

5.4.3. Regla: 1 6 1 6. Densidad: 10 %

Como se puede observar en la figura 86 esta regla forma una especie de laberinto que va cambiando pero en determinada generación, este patrón empieza a repetirse y oscila cada dos generaciones. (Sin memoria)



Figura 86: Resultado de iterar 100 veces sin memoria

En la figura 87, el comportamiento de este autómata usando la función de máximo es un tanto similar a su comportamiento normal dado que llega el momento que empiezan a oscilar sus formas y se repiten los mismo patrones, sin embargo, la forma que se consigue es distinta, ya que “el laberinto” formado se compone de paredes más delgadas.

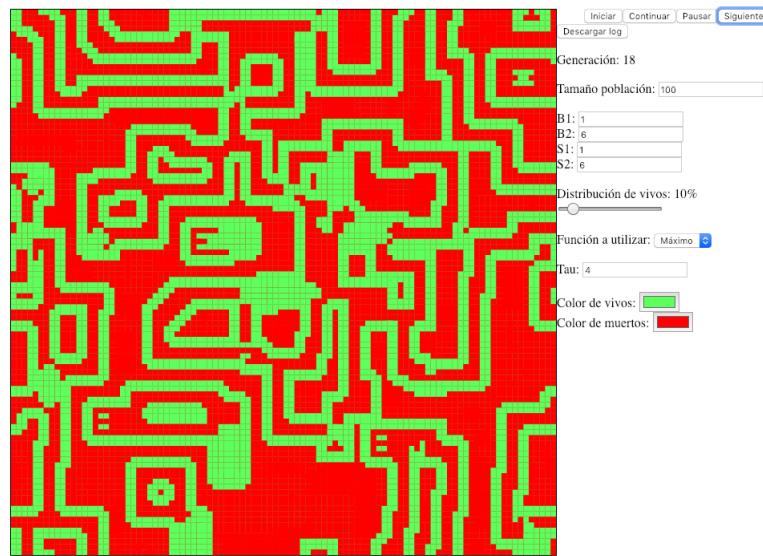


Figura 87: Resultado de iterar usando función de máximo

La matriz auxiliar del autómata anterior, forma igualmente un “laberinto” sin embargo sus paredes son más gruesas y además, también tiene un patrón que se repite eternamente.

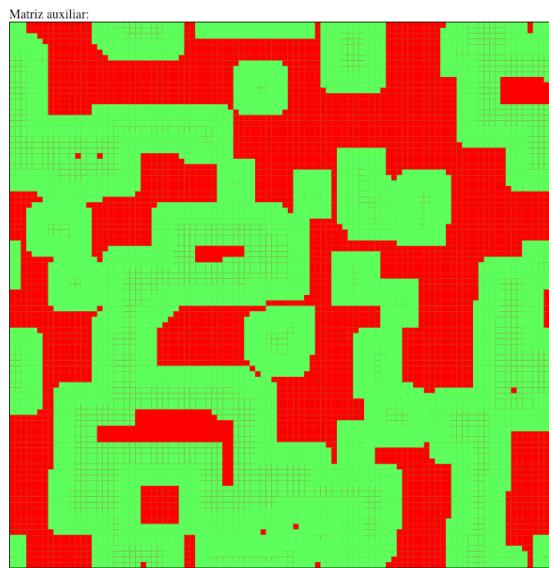


Figura 88: Matriz auxiliar

Sin embargo, en la figura 89 podemos observar el autómata con la función mínimo, de igual manera oscila repitiendo ciertos patrones solo que su periodo es mayor, ya que le toma más generaciones regresar a una forma que con máximo.

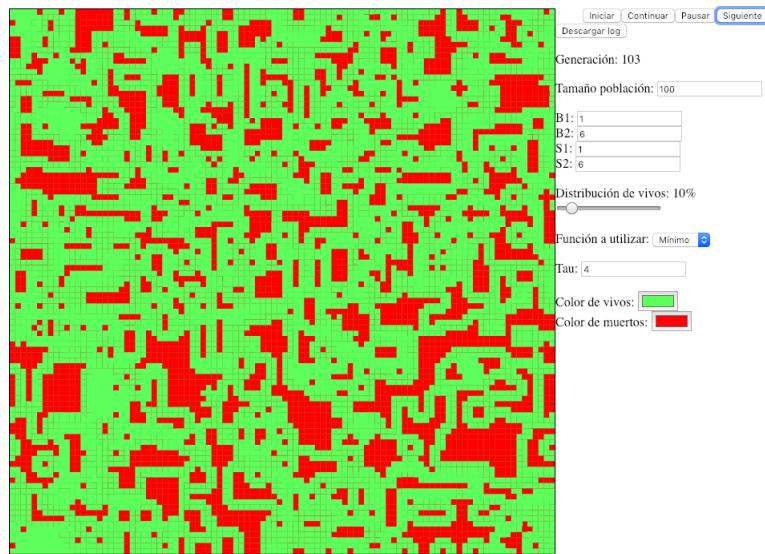


Figura 89: Resultado de iterar usando función de mínimo

La matriz auxiliar de igual forma oscila con la poca población que tiene.

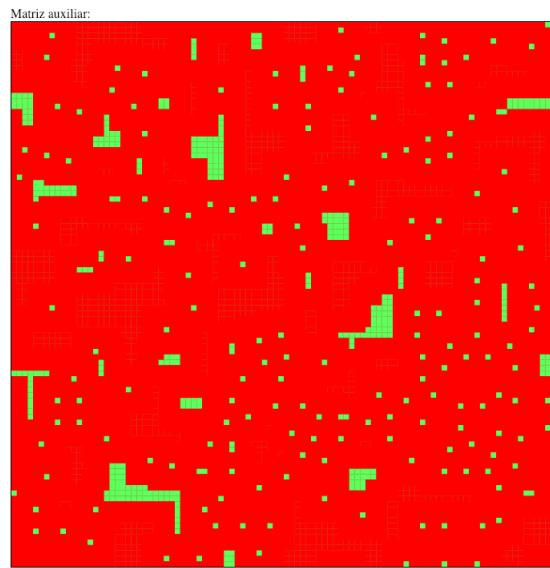


Figura 90: Matriz auxiliar

En la siguiente figura, se observa el resultado de usar la función de paridad, al igual que las anteriores funciones, se repiten los patrones en la malla de forma cíclica.

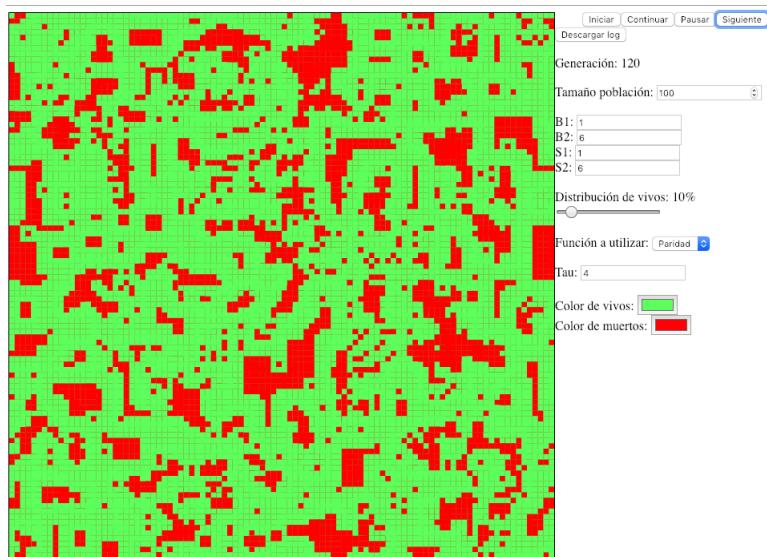


Figura 91: Resultado de iterar usando función de paridad

Cabe señalar, que paridad tiene algo interesante, ya que hay que observar su matriz auxiliar para notar que tiene una forma bastante particular.

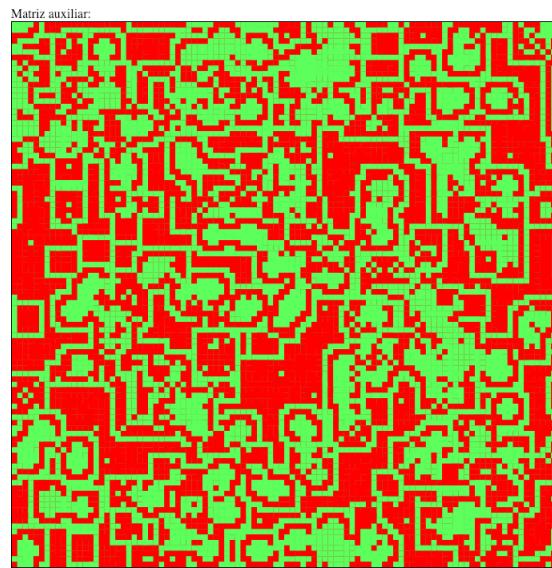


Figura 92: Matriz auxiliar

5.4.4. Regla: 3 3 1 8. Densidad: 10 %

Como se puede observar en la figura 93 esta regla después del paso de algunas generaciones queda estancada y no sufre cambio alguno (salvo alguno pequeño que no llega a deformarlo), la población se mantiene oscilando entre un límite mayor y un límite menor.

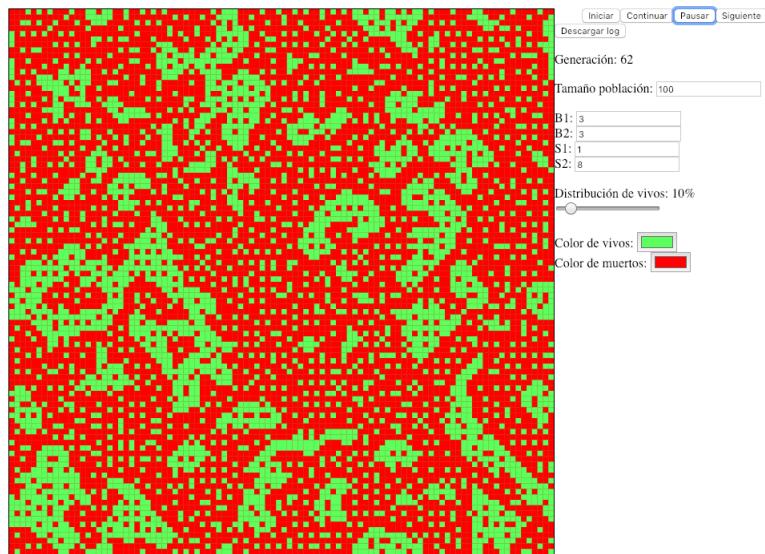


Figura 93: Resultado de iterar 100 veces sin memoria

En la figura 94, el comportamiento de este autómata usando la función de máximo es distinta a su comportamiento normal, ya que esta función provoca que el autómata oscile repitiendo formas cada cierto número de generaciones, y dentro de este se forman cuadrados y rectángulos los cuales a su vez dentro tienen otras formas como mosaicos.

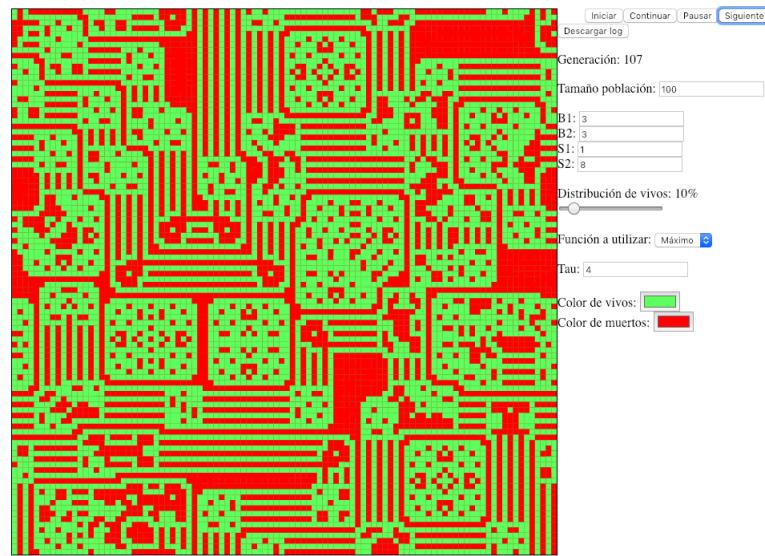


Figura 94: Resultado de iterar usando función de máximo

La matriz auxiliar del autómata anterior, tiene una forma bastante peculiar, ya que recuerda a un rompecabezas, debido a que tiene secciones muy definidas dentro de él las cuales no se tocan ni superponen.

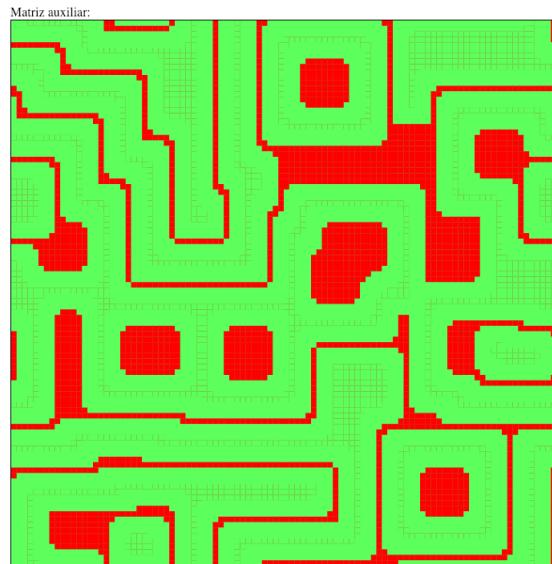


Figura 95: Matriz auxiliar

Al aplicar la función de mínimo, el autómata de igual forma pareciera que se encuentra oscilando y de cierta forma recuerda a un circuito eléctrico (a una placa mejor dicho).

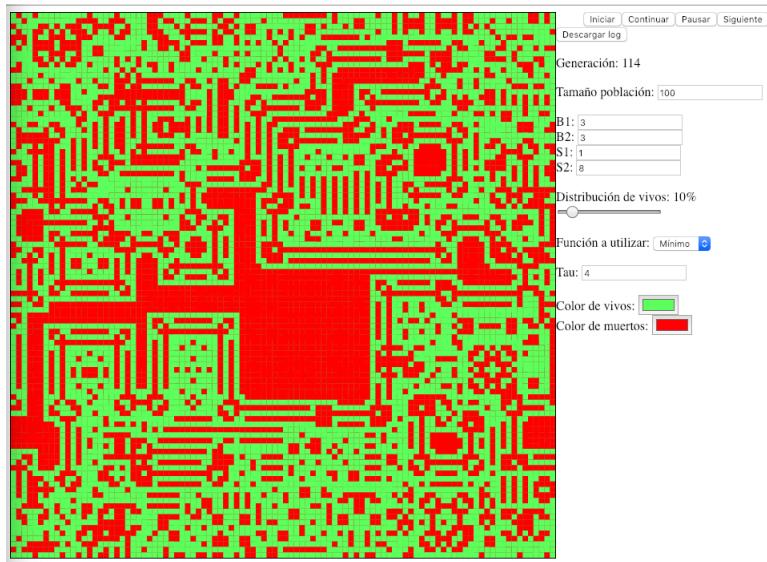


Figura 96: Resultado de iterar usando función de mínimo

La matriz auxiliar de igual manera oscila, pero con una población mucho menor.

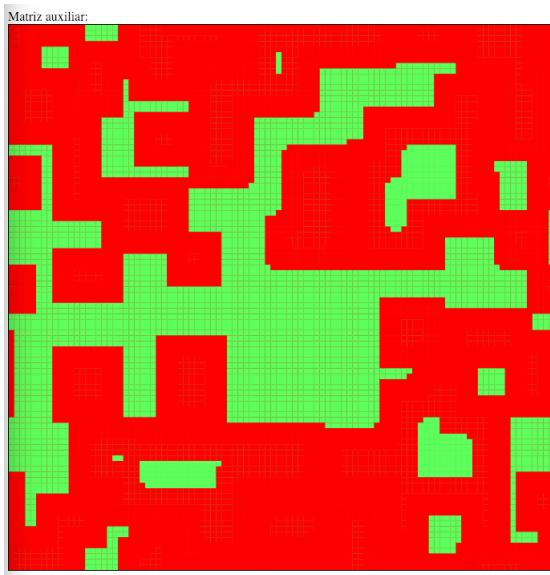


Figura 97: Matriz auxiliar

En la siguiente figura, se observa el resultado de usar la función de paridad, la población no desciende mucho en este caso y pareciera se mantiene constante.

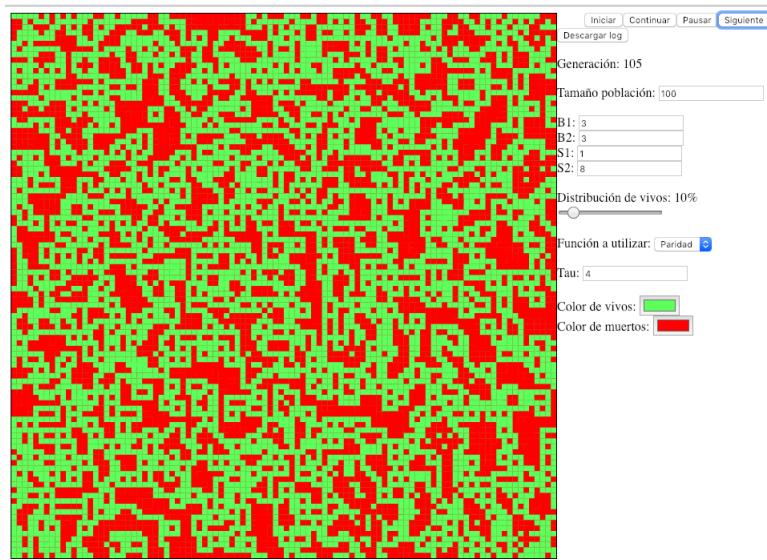


Figura 98: Resultado de iterar usando función de paridad

la matriz auxiliar tiene una gran cantidad de población la cual también varia muy poco. Parece ser que paridad es la función más estable para este autómata.

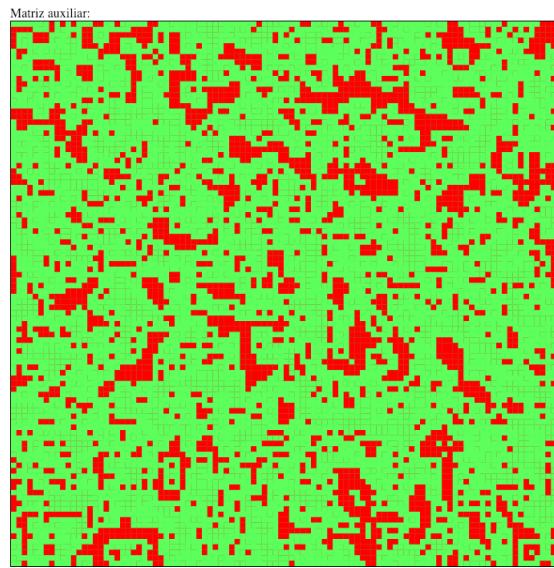


Figura 99: Matriz auxiliar

5.4.5. Regla: 3 3 1 7. Línea en el centro de 7 cuadros

Como se puede observar en la figura 100, esta regla provoca mosaicos que se repiten indeterminadamente (sin memoria), para esto, se colocaron 7 cuadros en el centro en línea horizontal.

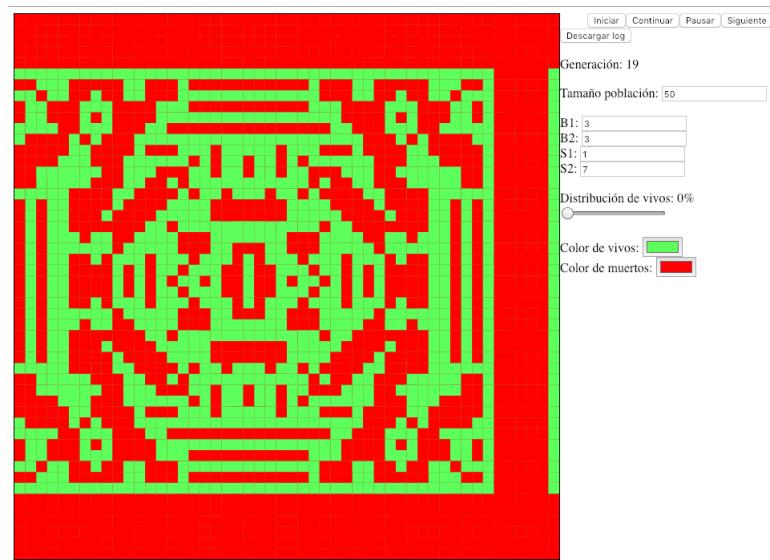


Figura 100: Resultado de iterar 20 veces sin memoria

En la figura 101, se muestra el comportamiento de este autómata al aplicar la función de máximo, el cual forma mosaicos, pero tienen una forma diferente a los que produce “normal”, ya que ahora estos en el centro tienen un rectángulo.

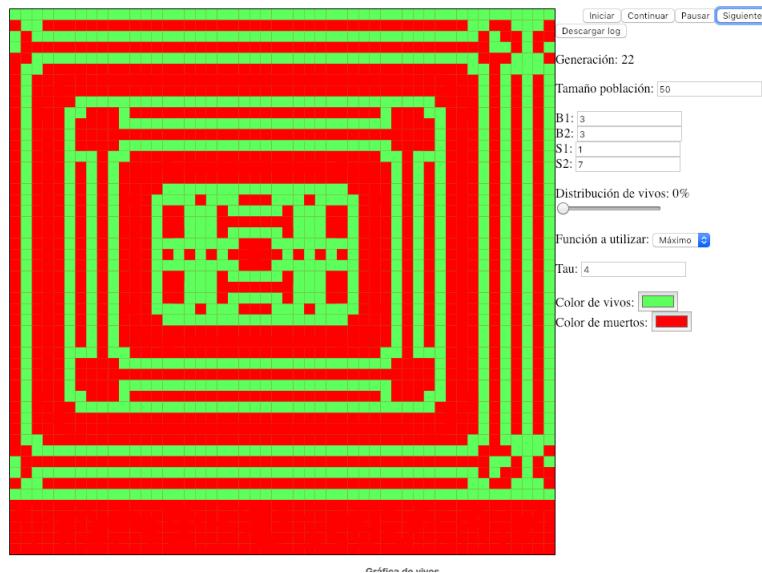


Figura 101: Resultado de iterar usando función de máximo

De igual forma, la matriz auxiliar, forma un mosaico, pero este es mucho menos vistoso ya que tiene líneas muy gruesas.

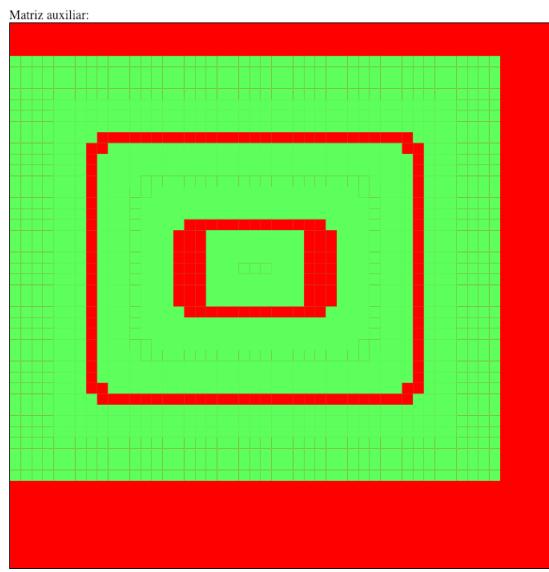


Figura 102: Matriz auxiliar

Al aplicar la función de mínimo, el autómata después de casi 80 generaciones, empezó a formar líneas las cuales son paralelas y abarcan el espacio de todo el canvas. Este es el primer autómata que veo hace eso.

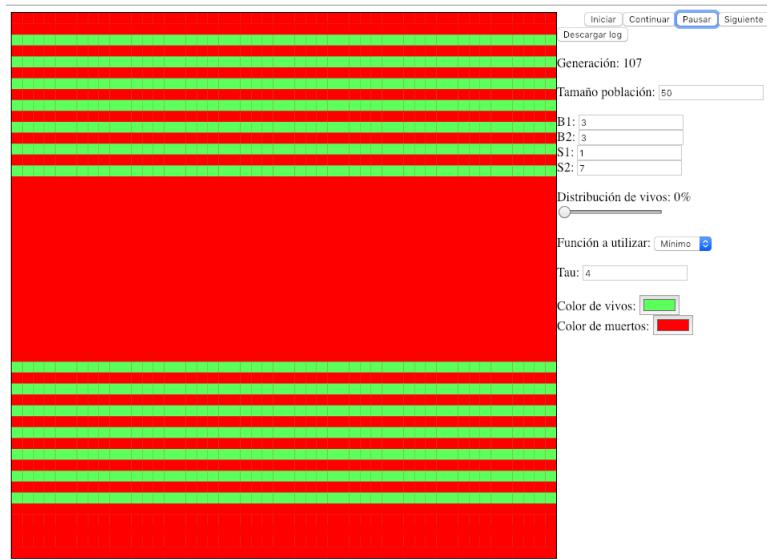


Figura 103: Resultado de iterar usando función de mínimo

La matriz auxiliar del autómata anterior, de igual forma cuenta con líneas, solo que estás son mucho mas gruesas, y diden el espacio en tres.

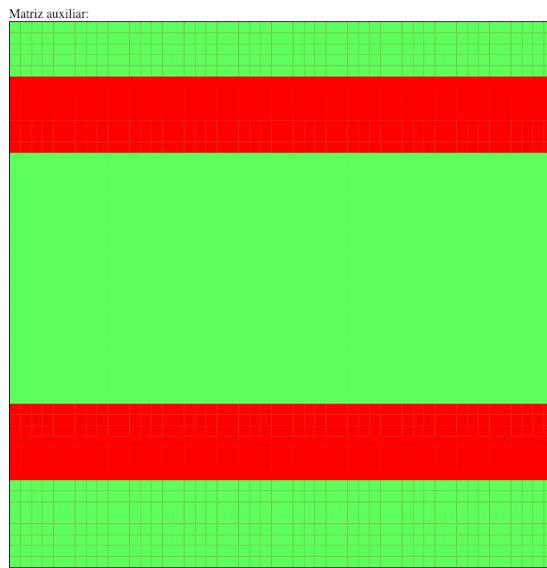


Figura 104: Matriz auxiliar

Finalmente, al usar la función de paridad, se obtiene algo muy interesante, ya que de igual forma se empieza a formar un mosaico, pero las formas dentro de este son muy curiosas, ya que son figuras geométricas complejas que hasta el momento no había visto en otro autómata.

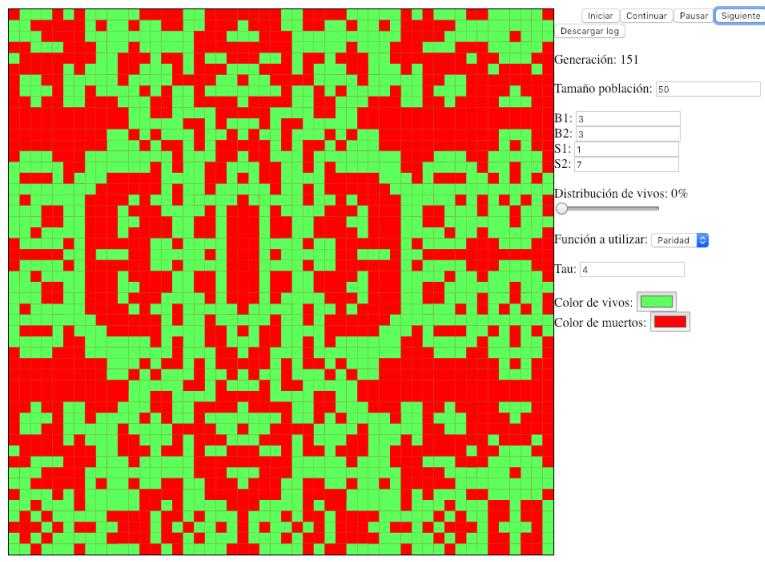


Figura 105: Resultado de iterar usando función de paridad

la matriz auxiliar de este autómata, igualmente forma un mosaico con colores invertidos.

Matriz auxiliar:

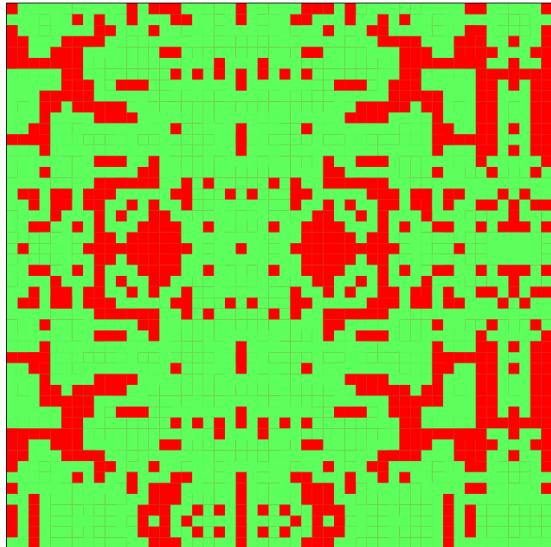


Figura 106: Matriz auxiliar

5.5. Conclusiones

El uso de una función y una matriz auxiliar para el calculo de las iteraciones en un autómata celular cambian el comportamiento de la función original, es decir, si se realiza la prueba y se compara la gráfica de unos de la regla del juego de la vida con y sin función auxiliar se puede apreciar que la que tiene la función auxiliar oscila con más frecuencia a diferencia de la que no utiliza una función extra, en esta la gráfica se estabiliza de una forma más rápida.

El uso de esta técnica al trabajar con autómatas celulares es bastante útil ya que permite el observar nuevos comportamientos con base a reglas ya conocidas, lo cual puede agilizar el estudio de estos modelos. Además de que nos acerca a modelar de forma un poco más realista los procesos del universo.

Referencias

- [1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introducción a La Teoría De Autómatas, Lenguajes Y Computación*. Addison-Wesley, 2007.
- [2] Reyes Gómez, D., *Descripción y Aplicaciones de los Autómatas Celulares*. cinvestav, 2011.
- [3] M. Gardner, “Mathematical games the fantastic combinations of john conway’s new solitaire game life.” https://web.archive.org/web/20090603015231/http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm, 1970. [Consultado: 2018-09-19].
- [4] De Felipe Vargas, D. and Sanchez Salazar C., *Comportamiento colectivo no trivial implementado en robots de bajo costo: el caso de “La hormiga de Langton”*. Instituto Politécnico Nacional, 2016.