Instituto Politécnico Nacional

Escuela Superior De Computo
ESCOM

# REPORTE PROGRAMA 3

David Baltazar Real

Grupo: 3CM19

Asignatura: Computing Selected Topics

Profesor: Juarez Martinez Genaro

Fecha de entrega: 17 de Enero del 2023

# Índice general

**1 Introducción** **2**

**2 Capturas** **3**

**3 Códigos** **7**
3.1 index.html . . . . . 7
3.2 styles.css . . . . . 9
3.3 js/main.js . . . . . 9
3.4 js/Ant.js . . . . . 14

**4 Conclusiones** **22**

# Índice de figuras

2.1 (Espacio 50x50) Generacion 0 con poblacion de 87 . . . . . 3
2.2 (Espacio 50x50) Generacion 85 con poblacion 0 . . . . . 4
2.3 (Espacio 100x100) Generacion 0 con poblacion 3,678 . . . . . 4
2.4 (Espacio 100x100) Generacion 12 con poblacion 3711 . . . . . 5
2.5 (Espacio 100x100) Generacion 85 con poblacion 279 . . . . . 5
2.6 (Espacio 100x100) Generacion 128 con poblacion 117 . . . . . 6
2.7 (Espacio 100x100) Generacion 164 con poblacion 0 . . . . . 6

# Capítulo 1

# Introducción

En 1986 Christopher Langton desarrollo un algoritmo sencillo, un autómata celular llamado "La Hormiga de Langton", esté funciona en una maya cuadriculada bidimensional con celdas generalmente blancas o negras, donde la hormiga es la cabeza lectora e interpretadora que se desplaza de acuerdo al estado que va encontrando hacia cualquiera de los puntos cardinales sobre ésta red, siguiendo un conjunto de reglas diseñadas especialmente para guiar su comportamiento.

## REGLAS:

1. La hormiga da un paso adelante.
2. Si la hormiga encuentra un cuadro blanco, lo pinta de negro, da un giro de 90o hacia la derecha y avanza una unidad.
3. Si la hormiga encuentra un cuadro negro, lo pinta de blanco, girá 90o hacia la izquierda y avanza una unidad.

Lo interesante de este AC es que a pesar de seguir sólo esas pocas reglas sencillas da origen a muchas posibilidades caóticas, debido a que no existe ningún método general de análisis que nos de a conocer la posición de las hormigas después de un determinado número de movimientos.
En esta practica observaremos el comportamiento caótico que “La Hormiga de Langton” presenta definir algunas reglas extras las cuales son condiciones de nacimiento y deceso.

# Capítulo 2

# Capturas

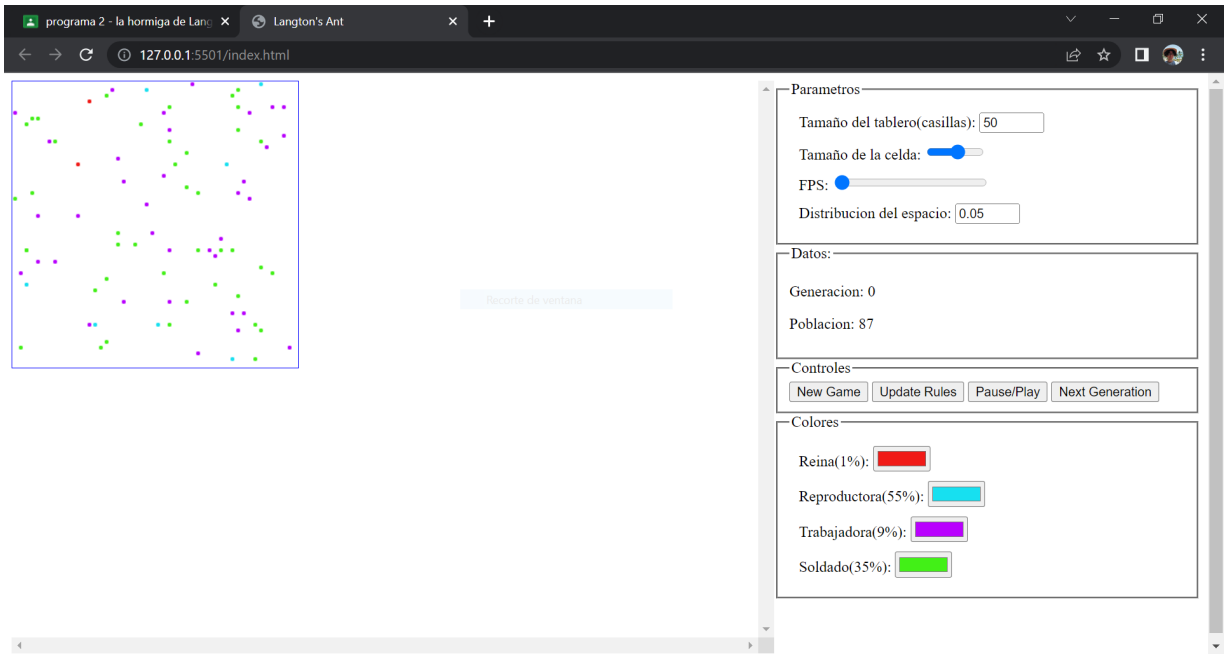


Figura 2.1: (Espacio 50x50) Generacion 0 con poblacion de 87

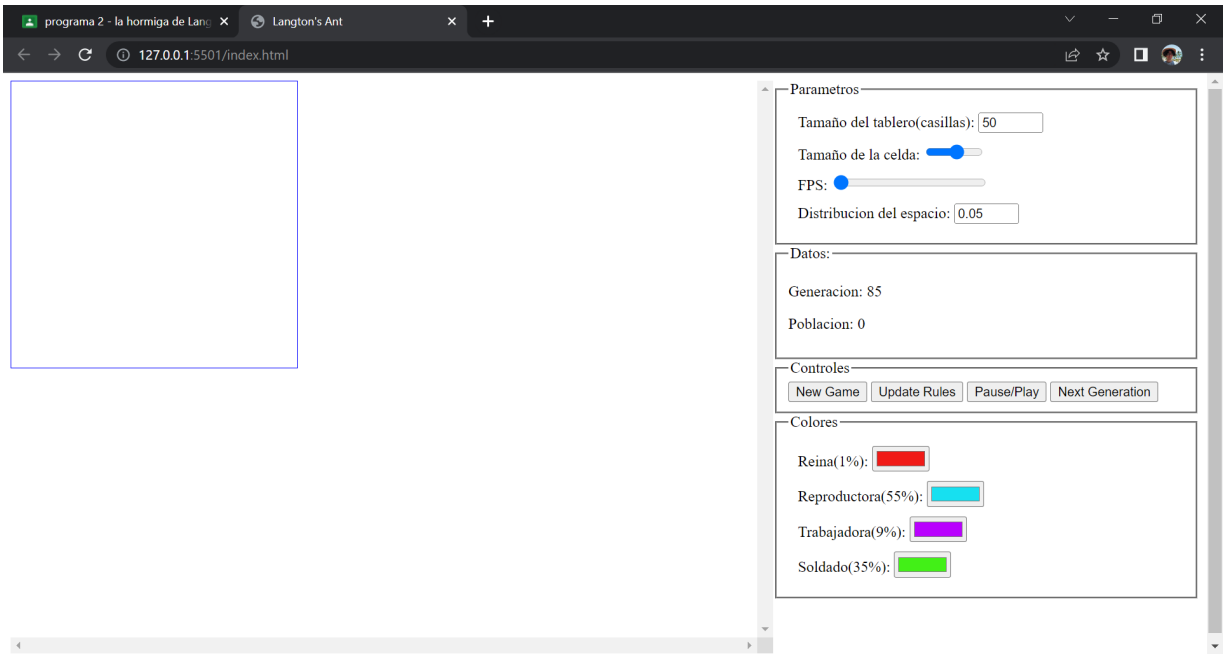


Figura 2.2: (Espacio 50x50) Generacion 85 con poblacion 0

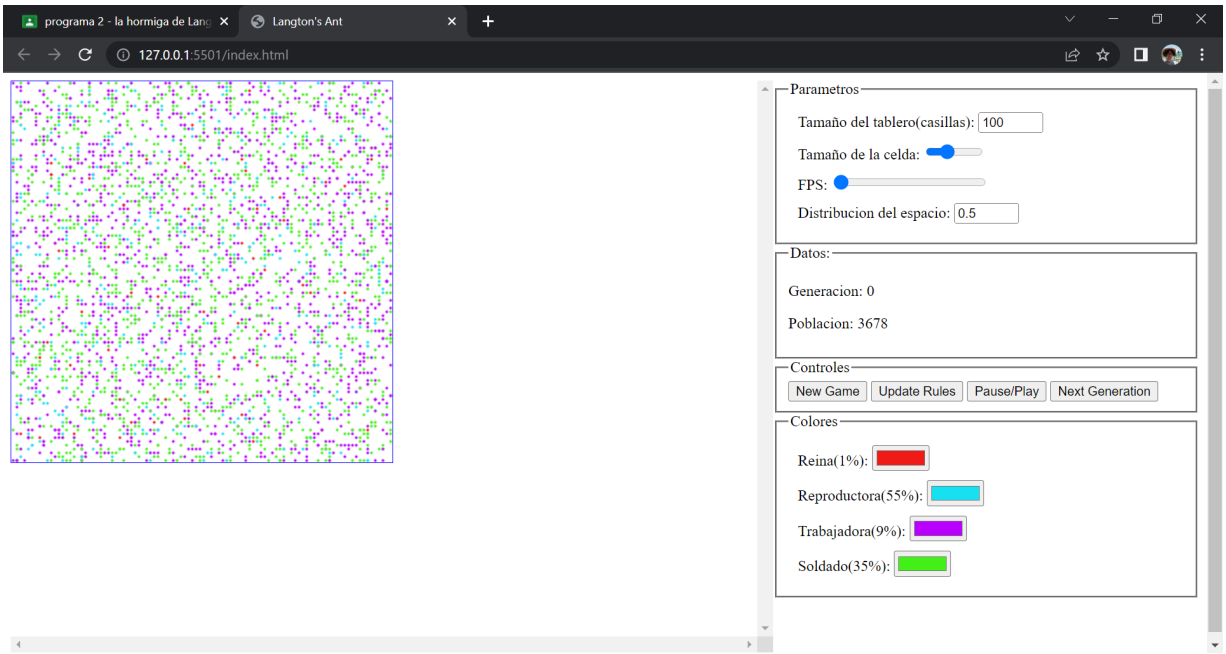


Figura 2.3: (Espacio 100x100) Generacion 0 con poblacion 3,678

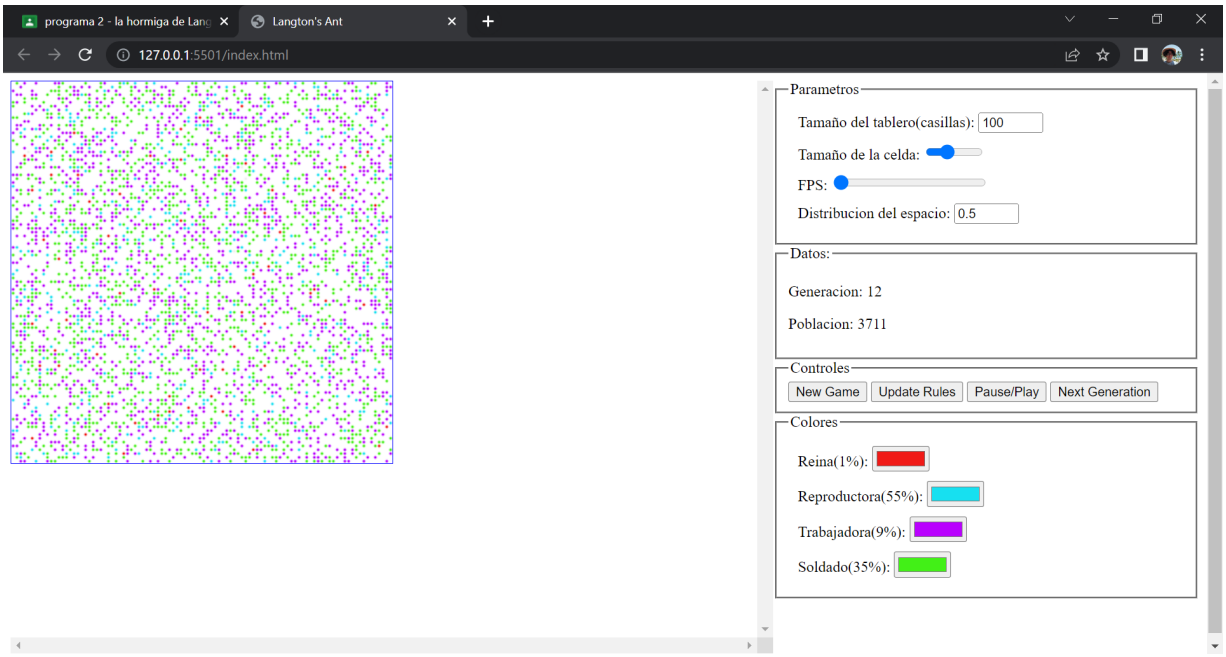


Figura 2.4: (Espacio 100x100) Generacion 12 con poblacion 3711

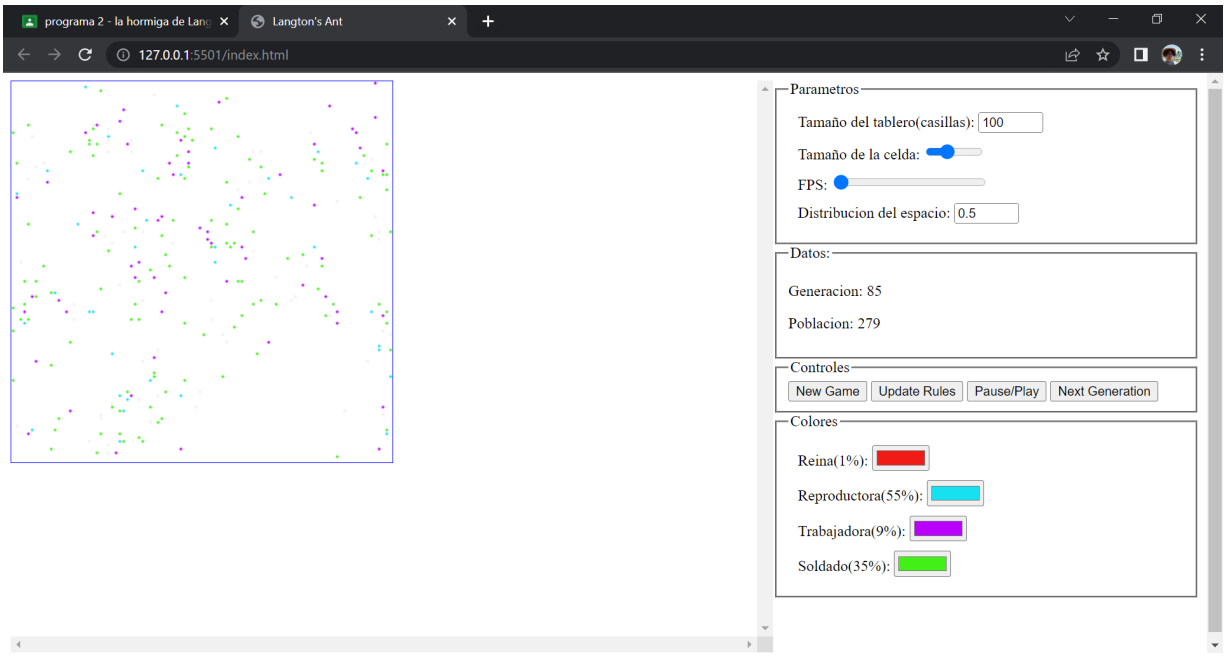


Figura 2.5: (Espacio 100x100) Generacion 85 con poblacion 279

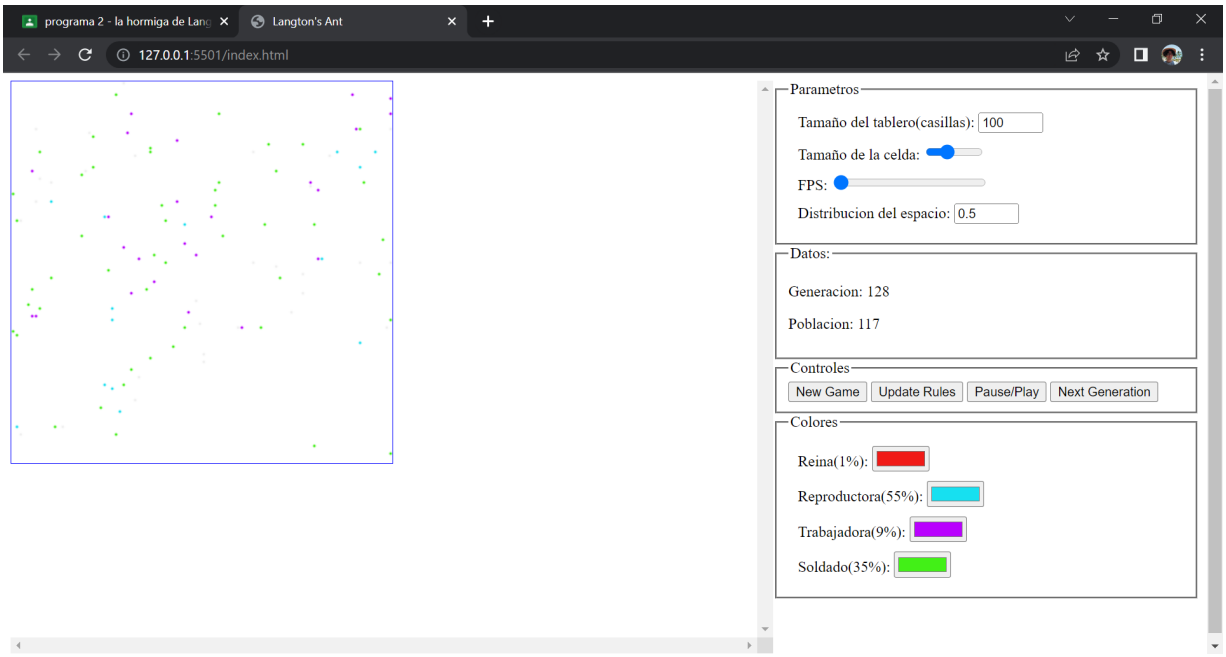


Figura 2.6: (Espacio 100x100) Generacion 128 con poblacion 117

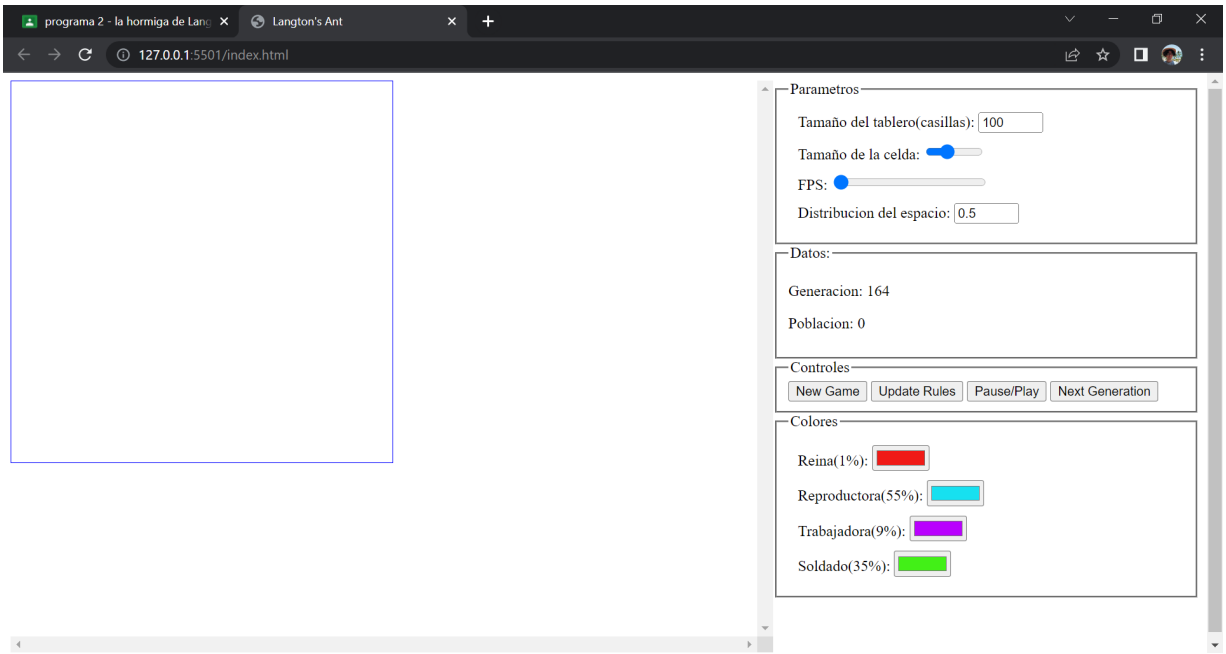


Figura 2.7: (Espacio 100x100) Generacion 164 con poblacion 0

# Capítulo 3

# Códigos

## 3.1. index.html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
7   <title>Langton's Ant</title>
8   <link rel="stylesheet" href="styles.css">
9   </head>
10  <body>
11  <div class="container" >
12  <div id="canvas-scroll" class="left">
13  <canvas id="canvas"></canvas>
14  </div>
15
16  <div class="right">
17  <fieldset class="parametros">
18  <legend>Parametros</legend>
19  <div class="parametros-section">
20  <div class="parametro">
21  <label for="grid-size">Tamaño del tablero(casillas):</label>
22  <input type="number" name="grid-size" id="grid-size">
23  </div>
24
25  <div class="parametro">
26  <label for="cell-size">Tamaño de la celda:</label>
27  <input type="range" name="cell-size" id="cell-size" value="5" min=
        "1" max="10" step="1">
28  </div>
29
30  <div class="parametro">
31  <label for="frame-rate">FPS:</label>
32  <input type="range" name="frame-rate" id="frame-rate" value="60"
        min="10" max="60" step="5">
```

```
33    </div>
34
35    <div class="parametro">
36    <label for="distribution-percent">Distribucion del espacio:</label>
37    <input type="number" name="distribution-percent" id="
          distribution-percent" value="0.5" min="0" max="1">
38    </div>
39
40    </div>
41    </fieldset>
42
43    <fieldset class="datos">
44    <legend>Datos:</legend>
45    <p>Generacion: <span id="generacion"></span></p>
46    <p>Poblacion: <span id="poblacion"></span></p>
47    </fieldset>
48
49    <fieldset class="controles">
50    <legend>Controles</legend>
51    <button id="new-game">New Game</button>
52    <button id="update-rules">Update Rules</button>
53    <button id="pause-play">Pause/Play</button>
54    <button id="next-gen">Next Generation</button>
55    </fieldset>
56
57    <fieldset class="colores">
58    <legend>Colores</legend>
59    <div class="parametro">
60    <label for="reina">Reina(1%):</label>
61    <input type="color" name="reina" id="reina" value="#F01A17"
          readonly>
62    </div>
63
64    <div class="parametro">
65    <label for="reproductora">Reproductora(55%):</label>
66    <input type="color" name="reproductora" id="reproductora" value=
          "#17E0F0" readonly>
67    </div>
68
69    <div class="parametro">
70    <label for="trabajadora">Trabajadora(9%):</label>
71    <input type="color" name="trabajadora" id="trabajadora" value="#
          B900FE" readonly>
72    </div>
73
74    <div class="parametro">
75    <label for="soldado">Soldado(35%):</label>
76    <input type="color" name="soldado" id="soldado" value="#42F017"
          readonly>
77    </div>
78    </fieldset>
79    </div>
80    </div>
81
```

```
<script type="module" src="./js/main.js"></script>
</body>
</html>
```

## 3.2. styles.css

```
#canvas-scroll {
  max-height: 600px;
  height: 600px;
  max-width: 800px;
  width: 800px;
  overflow: scroll;
}

#canvas {
  border: 1px solid blue;
}

.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
}

.parametro {
  margin: 10px;
}


.parametro input {
  width: 60px;
}

.parametro #frame-rate {
  width: 160px
}
```

## 3.3. js/main.js

```
import { Trabajadora, Soldado, Reproductora, Reina } from "./Ant.js
    ";

/***************/
/** VARIABLES **/
/***************/

// Elementos HTML //
// Parametros
const gridSizeElem = document.getElementById('grid-size');
const cellSizeElem = document.getElementById('cell-size');
```

```
11    const frameRateElem = document.getElementById('frame-rate');
12    const distributionPercentElem = document.getElementById('
          distribution-percent');
13    // Datos
14    const generacionElem = document.getElementById('generacion');
15    const poblacionElem = document.getElementById('poblacion');
16    // Botones
17    const newGameBtn = document.getElementById('new-game');
18    const updateBtn = document.getElementById('update-rules');
19    const playBtn = document.getElementById('pause-play');
20    const nextGenBtn = document.getElementById('next-gen');
21    // ----------------- //
22
23
24    // Canvas
25    const canvas = document.getElementById('canvas');
26    const ctx = canvas.getContext('2d');
27    // Datos
28    var generacion;
29    var poblacion;
30    var FPS;
31    var porcentajeDistribucion;
32    // Tablero
33    var cellSize;
34    var gridSize;
35    var tablero;
36    // Variables de control
37    var playing;
38    var gameInterval;
39    // Hormgias
40    var hormigas = [];
41    var cantidadDistribucion;
42    // Colores de las hormigas
43    const colorReina = '#F01A17'; // rojo
44    const colorReproductora = '#17E0F0'; // cyan
45    const colorTrabajadora = '#B900FE' // morado
46    const colorSoldado = '#42F017'; // verde fuerte
47    // Porcentaje de probabilidad de hormigas
48    const probabilidadReina = .01;
49    const probabilidadReproductora = .09;
50    const probabilidadSoldado = .35;
51    const probabilidadTrabajadora = .55;
52
53    /**********************/
54    /** EVENT LISTENERS **/
55    /**********************/
56
57    document.addEventListener('DOMContentLoaded', loadDefaultSettings);
58    newGameBtn.addEventListener('click', newGame);
59    updateBtn.addEventListener('click', updateRules);
60    playBtn.addEventListener('click', play);
61    nextGenBtn.addEventListener('click', nextGen);
62    canvas.addEventListener('mousedown', e => clickCanvas(e));
63
```

```
64  /***************/
65  /** FUNCIONES **/
66  /***************/
67
68  function loadDefaultSettings() {
69    // Parametros del programa
70    gridSizeElem.value = 10;
71    cellSizeElem.value = 5;
72    generacion = 0;
73    distributionPercentElem.value = 0.05;
74    frameRateElem.value = 50;
75
76    playing = false;
77
78    updateRules();
79  }
80
81  function updateRules() {
82    if (playing === true) {
83      play();
84    }
85    // Obtenemos los valores de los elementos HTML
86    gridSize = gridSizeElem.value;
87    cellSize = cellSizeElem.value;
88    porcentajeDistribucion = distributionPercentElem.value;
89    FPS = frameRateElem.value;
90    cantidadDistribucion = Math.floor((gridSize * gridSize) *
        porcentajeDistribucion);
91    // Tamaño y limpieza del canvas
92    canvas.width = gridSize * cellSize;
93    canvas.height = gridSize * cellSize;
94    // Imprimimos las hormigas
95    hormigas.forEach(function (hormiga) {
96      hormiga.dibujarHormiga(ctx, cellSize);
97    });
98    imprimirDatos();
99  }
100
101 function play() {
102   playing = !playing;
103   if (playing) {
104     gameInterval = setInterval(nextGen, 1000 / FPS);
105   } else {
106     clearInterval(gameInterval)
107   }
108 }
109
110 function newGame() {
111   if (playing === true) {
112     play(); // Pausamos el juego si se esta ejecutando
113   }
114   // Reiniciamos los datos
115   generacion = 0;
116   poblacion = 0;
```

```
117     // Creamos el tablero
118     tablero = iniciarMundo();
119     // Generamos las hormigas en el mundo
120     generarHormigas();
121     updateRules();
122     imprimirDatos();
123   }
124
125   function nextGen() {
126     let hormigasMuertas = [];
127     for (let i = 0; i < hormigas.length; i++) {
128       hormigas[i].turno(ctx, tablero, cellSize, hormigas);
129       if (hormigas[i].periodoVida <= 0) {
130         hormigas[i].morir(ctx, tablero, cellSize);
131         hormigasMuertas.push(hormigas[i].id);
132       }
133     }
134
135     hormigasMuertas.forEach(function (id) {
136       quitarHormiga(id);
137     });
138
139     if (generacion === 80) {
140       console.log(hormigasMuertas);
141       console.log(hormigas);
142     }
143
144     generacion = generacion + 1;
145     generacionElem.innerText = generacion;
146     poblacionElem.innerText = hormigas.length;
147   }
148
149   function iniciarMundo() {
150     let nuevoMundo = new Array(gridSize);
151     for (let i = 0; i < gridSize; i++) {
152       nuevoMundo[i] = new Array(gridSize);
153     }
154
155     // Llenamos la matriz de 0's
156     for (let y = 0; y < gridSize; y++) {
157       for (let x = 0; x < gridSize; x++) {
158         nuevoMundo[y][x] = 0;
159       }
160     }
161     return nuevoMundo;
162   }
163
164   function generarHormigas() {
165     hormigas = []
166     for (let i = 0; i < cantidadDistribucion; i++) {
167       crearHormiga(Math.floor(Math.random() * gridSize), Math.floor(
            Math.random() * gridSize));
168     }
169     poblacion = hormigas.length;
```

```
170   }
171
172   function crearHormiga(x, y) {
173     let nuevaHormiga = null;
174     if (Math.random() < probabilidadReina) {
175       // Creamos una hormiga reina
176       nuevaHormiga = new Reina(
177       x,
178       y,
179       generarDireccion(),
180       colorReina
181       );
182     } else if (Math.random() < probabilidadReproductora) {
183       // Creamos una hormiga reproductora
184       nuevaHormiga = new Reproductora(
185       x,
186       y,
187       generarDireccion(),
188       colorReproductora
189       );
190     } else if (Math.random() < probabilidadSoldado) {
191       // Creamos una hormiga soldado
192       nuevaHormiga = new Soldado(
193       x,
194       y,
195       generarDireccion(),
196       colorSoldado
197       );
198     } else if (Math.random() < probabilidadTrabajadora) {
199       // Creamos una hormiga trabajadora
200       nuevaHormiga = new Trabajadora(
201       x,
202       y,
203       generarDireccion(),
204       colorTrabajadora
205       );
206     }
207
208     if (nuevaHormiga != null) {
209       nuevaHormiga.dibujarHormiga(ctx, cellSize);
210       hormigas.push(nuevaHormiga);
211     }
212
213   }
214
215   function clickCanvas(e) {
216     let coordenadas = getCursorPosition(e);
217     poblacion = poblacion + 1;
218     crearHormiga(coordenadas[0], coordenadas[1]);
219     imprimirDatos();
220   }
221
222
223   function imprimirDatos() {
```

```
224     poblacionElem.innerText = hormigas.length;
225     generacionElem.innerText = generacion;
226   }
227
228   function generarDireccion() {
229     let direcciones = ['U', 'D', 'L', 'R'];
230     return direcciones[Math.floor(Math.random() * 3).toFixed(0)];
231   }
232
233   function quitarHormiga(id) {
234     let pos = hormigas.findIndex(hormiga => hormiga.id === id);
235     hormigas.splice(pos, 1);
236   }
237
238   function getCursorPosition(event) {
239     let rect = canvas.getBoundingClientRect();
240     let x = Math.floor((event.clientX - rect.left) / cellSize);
241     let y = Math.floor((event.clientY - rect.top) / cellSize);
242
243     return [x, y];
244   }
```

## 3.4. js/Ant.js

```
1   var idGlobal = 0;
2   // Colores
3   const colorReina = '#F01A17'; // rojo
4   const colorReproductora = '#17E0F0'; // cyan
5   const colorTrabajadora = '#B900FE' // morado
6   const colorSoldado = '#42F017'; // verde fuerte
7   const colorBlanco = '#ffffff';
8   // Probabilidades
9   const probabilidadReina = .01;
10  const probabilidadReproductora = .09;
11  const probabilidadSoldado = .35;
12  const probabilidadTrabajadora = .55;
13  export default class Ant {
14    constructor(x, y, dir, colorAnt) {
15      this.id = idGlobal;
16      this.x = x;
17      this.y = y;
18      this.colorAnt = colorAnt;
19      this.dir = dir;
20      this.periodoVida = 80;
21
22      idGlobal = idGlobal + 1;
23    }
24
25    girarDerecha() {
26      if (this.dir === 'U') {
27        this.dir = 'R';
```

```
28        } else if (this.dir === 'R') {
29          this.dir = 'D';
30        } else if (this.dir === 'D') {
31          this.dir = 'L';
32        } else {
33          this.dir = 'U'
34        }
35      }
36
37      girarIzquierda() {
38        if (this.dir === 'U') {
39          this.dir = 'L';
40        } else if (this.dir === 'L') {
41          this.dir = 'D';
42        } else if (this.dir === 'D') {
43          this.dir = 'R';
44        } else {
45          this.dir = 'U'
46        }
47      }
48
49      dibujarHormiga(ctx, cellSize) {
50        ctx.beginPath();
51        ctx.arc(this.x * cellSize + (cellSize / 2), this.y * cellSize +
              (cellSize / 2), cellSize / 3, 0, 2 * Math.PI, false);
52        ctx.fillStyle = this.colorAnt;
53        ctx.fill();
54      }
55
56      avanzar(mundo) {
57        if (this.dir === 'U') {
58          // Avanza hacia arriba
59          this.y -= 1;
60        } else if (this.dir === 'R') {
61          // Avanza hacia la derecha
62          this.x += 1
63        } else if (this.dir === 'D') {
64          // Avanza hacia abajo
65          this.y += 1
66        } else {
67          // Avanza hacia la izquierda
68          this.x -= 1;
69        }
70
71        // Comprobamos que no se pase del mundo
72        if (this.x < 0) {
73          this.x = mundo.length - 1;
74        } else if (this.y < 0) {
75          this.y = mundo.length - 1;
76        } else if (this.x >= mundo.length) {
77          this.x = 0;
78        } else if (this.y >= mundo.length) {
79          this.y = 0;
80        }
```

```
81
82      }
83
84      turno(ctx, mundo, cellSize, hormigas) {
85        this.periodoVida = this.periodoVida - 1;
86        // Giramos la hormiga y escogemos el colorPath
87        if (mundo[this.y][this.x] === 0) {
88          // Casilla blanca
89          this.girarIzquierda();
90        } else {
91          // Casilla Negra
92          this.girarDerecha();
93        }
94
95        // Comprobamos si hay 2 veces (Segun lo pedido)
96        let colision = this.comporbarColision(mundo, hormigas);
97        if (colision[0]) {
98          while (true) {
99            let dirAux = this.generarDireccion();
100           if (dirAux != this.dir) {
101             this.dir = dirAux;
102             break;
103           }
104         }
105       }
106       colision = this.comporbarColision(mundo, hormigas);
107       if (!colision[0]) {
108         // Avanzamos en el tablero y cambiamos el valor/colorPath de
              la casilla
109         mundo[this.y][this.x] = 1 - mundo[this.y][this.x];
110         ctx.fillStyle = colorBlanco;
111         ctx.fillRect(this.x * cellSize, this.y * cellSize, cellSize,
              cellSize)
112         this.avanzar(mundo)
113         // Dibujamos la hormiga
114         this.dibujarHormiga(ctx, cellSize);
115       }
116     }
117
118     morir(ctx, mundo, cellSize) {
119       ctx.fillStyle = colorBlanco;
120       ctx.fillRect(this.x * cellSize, this.y * cellSize, cellSize,
            cellSize)
121     }
122
123     comporbarColision(mundo, hormigas) {
124       let sigX, sigY; // Siguiente valor de x,y
125       let N = mundo.length;
126       let colision = false;
127       let idHormigaColision;
128       // Obtenemos a donde se desplazaria en este turno
129       if (this.dir == 'U') {
130         // arriba
131         sigX = this.x;
```

```
132           sigY = (this.y - 1 + N) % N;
133         } else if (this.dir == 'D') {
134           // abajo
135           sigX = this.x;
136           sigY = (this.y + 1 + N) % N;
137         } else if (this.dir == 'R') {
138           // derecha
139           sigX = (this.x + 1 + N) % N;
140           sigY = this.y;
141         } else {
142           // izquierda
143           sigX = (this.x - 1 + N) % N;
144           sigY = this.y;
145         }
146         // Comprobamos la colision
147         for (let i = 0; i < hormigas.length; i++) {
148           if (hormigas[i].x === sigX && hormigas[i].y === sigY) {
149             colision = true;
150             idHormigaColision = hormigas[i].id;
151           }
152         }
153 
154         return [colision, idHormigaColision, sigX, sigY];
155       }
156 
157       generarDireccion() {
158         let direcciones = ['U', 'D', 'L', 'R'];
159         return direcciones[Math.floor(Math.random() * 3).toFixed(0)];
160 
161       }
162     }
163 
164     // REINA //
165 
166     export class Reina extends Ant {
167       constructor(x, y, dir, colorPath, colorAnt) {
168         super(x, y, dir, colorPath, colorAnt);
169         this.tipo = 'reina';
170       }
171 
172       turno(ctx, mundo, cellSize, hormigas) {
173         this.periodoVida = this.periodoVida - 1;
174         // Giramos la hormiga y escogemos el colorPath
175         if (mundo[this.y][this.x] === 0) {
176           this.girarIzquierda();
177         } else {
178           this.girarDerecha();
179         }
180         // Comprobamos si hay alguna hormiga en la otra casilla y lo
               manejamos
181         let colision = this.comporbarColision(mundo, hormigas);
182         if (colision[0]) {
183           let iHormiga = hormigas.findIndex(hormiga => hormiga.id ===
                colision[1]);
```

```
184         let dirAux;
185         if (hormigas[iHormiga].tipo === 'reproductora') {
186           this.aparear(hormigas);
187         } else if (hormigas[iHormiga].tipo === 'reina') {
188           if (this.periodoVida >= 20) {
189             this.periodoVida = Math.random() < 0.50 ?
                  this.periodoVida : -1;
190           } else if (this.periodoVida < 20) {
191             this.periodoVida = Math.random() < 0.50 ?
                  this.periodoVida : -1;
192             return;
193           }
194         }
195         while (true) {
196           dirAux = this.generarDireccion();
197           if (dirAux != this.dir) {
198             this.dir = dirAux;
199             break;
200           }
201         }
202       }
203       colision = this.comporbarColision(mundo, hormigas);
204       if (!colision[0]) {
205         // Avanzamos en el tablero y cambiamos el valor/colorPath de
              la casilla
206         mundo[this.y][this.x] = 1 - mundo[this.y][this.x];
207         ctx.fillStyle = colorBlanco;
208         ctx.fillRect(this.x * cellSize, this.y * cellSize, cellSize,
              cellSize)
209         this.avanzar(mundo)
210         // Dibujamos la hormiga
211         this.dibujarHormiga(ctx, cellSize);
212       }
213     }
214
215     aparear(hormigas) {
216       if (Math.random() < probabilidadReina) {
217         // Creamos una hormiga reina
218         hormigas.push(new Reina(
219         this.x,
220         this.y,
221         this.generarDireccion(),
222         colorReina
223         ));
224       } else if (Math.random() < probabilidadReproductora) {
225         // Creamos una hormiga reproductora
226         hormigas.push(new Reproductora(
227         this.x,
228         this.y,
229         this.generarDireccion(),
230         colorReproductora
231         ));
232       } else if (Math.random() < probabilidadSoldado) {
233         // Creamos una hormiga soldado
```

```
234           hormigas.push(new Soldado(
235           this.x,
236           this.y,
237           this.generarDireccion(),
238           colorSoldado
239           ));
240         } else if (Math.random() < probabilidadTrabajadora) {
241           // Creamos una hormiga trabajadora
242           hormigas.push(new Trabajadora(
243           this.x,
244           this.y,
245           this.generarDireccion(),
246           '#F0F0F0',
247           colorTrabajadora
248           ));
249         }
250       }
251     }
252
253     // REPRODUCTORA //
254
255     export class Reproductora extends Ant {
256       constructor(x, y, dir, colorAnt) {
257         super(x, y, dir, colorAnt);
258         this.tipo = 'reproductora';
259       }
260       turno(ctx, mundo, cellSize, hormigas) {
261         this.periodoVida = this.periodoVida - 1;
262         // Giramos la hormiga y escogemos el colorPath
263         if (mundo[this.y][this.x] === 0) {
264           this.girarIzquierda();
265         } else {
266           this.girarDerecha();
267         }
268
269         // Comprobamos si hay alguna hormiga en la otra casilla y lo
               manejamos
270         let colision = this.comporbarColision(mundo, hormigas);
271
272         if (colision[0]) {
273           let iHormiga = hormigas.findIndex(hormiga => hormiga.id ===
                 colision[1]);
274           let dirAux;
275           if (hormigas[iHormiga].tipo === 'reina') {
276             console.log('Apareamiento');
277             this.aparear(hormigas[iHormiga], hormigas);
278           }
279           while (true) {
280             dirAux = this.generarDireccion();
281             if (dirAux != this.dir) {
282               this.dir = dirAux;
283               break;
284             }
285           }
```

```
286         }
287         colision = this.comporbarColision(mundo, hormigas);
288         if (!colision[0]) {
289           // Avanzamos en el tablero y cambiamos el valor/colorPath de
                la casilla
290           mundo[this.y][this.x] = 1 - mundo[this.y][this.x];
291           ctx.fillStyle = colorBlanco;
292           ctx.fillRect(this.x * cellSize, this.y * cellSize, cellSize,
                cellSize)
293           this.avanzar(mundo)
294           // Dibujamos la hormiga
295           this.dibujarHormiga(ctx, cellSize);
296         }
297
298       }
299
300       aparear(reina, hormigas) {
301         console.log('Aparear desde reina');
302         if (Math.random() < probabilidadReina) {
303           // Creamos una hormiga reina
304           hormigas.push(new Reina(
305           reina.x,
306           reina.y,
307           this.generarDireccion(),
308           colorReina
309           ));
310         } else if (Math.random() < probabilidadReproductora) {
311           // Creamos una hormiga reproductora
312           hormigas.push(new Reproductora(
313           reina.x,
314           reina.y,
315           this.generarDireccion(),
316           colorReproductora
317           ));
318         } else if (Math.random() < probabilidadSoldado) {
319           // Creamos una hormiga soldado
320           hormigas.push(new Soldado(
321           reina.x,
322           reina.y,
323           this.generarDireccion(),
324           colorSoldado
325           ));
326         } else if (Math.random() < probabilidadTrabajadora) {
327           // Creamos una hormiga trabajadora
328           hormigas.push(new Trabajadora(
329           reina.x,
330           reina.y,
331           this.generarDireccion(),
332           colorTrabajadora
333           ));
334         }
335       }
336     }
337
```

```
338   // SOLDADO //
339
340   export class Soldado extends Ant {
341     constructor(x, y, dir, colorAnt) {
342       super(x, y, dir, colorAnt);
343       this.tipo = 'soldado';
344     }
345   }
346
347   // TRABAJADORA //
348
349   export class Trabajadora extends Ant {
350     constructor(x, y, dir, colorAnt) {
351       super(x, y, dir, colorAnt);
352       this.tipo = 'trabajadora';
353     }
354   }
```

# Capítulo 4

# Conclusiones

Podemos observar como entra mayor sea la población hay mas probabilidad de estabilizar el numero de decesos con el de nacimientos y así poner en equilibrio el sistema, pero no hay como tal una forma formal de analizar el juego de la vida para poder saber cual es la distribución ni la configuración para hacerlo. Como se puede observar en las capturas, en la prueba de un espacio de 50 x 50 con una distribución del %5 no ocurre ningún nacimiento debido a la poca cantidad de hormigas reina que hay en el espacio por lo que al llegar a la generación 80 la población completa de hormigas se extingue mientras que en el espacio de 100 x 100 con una distribución del %50 al haber una mayor proporción de hormigas se contempla que la población empieza a crecer, sin embargo, no crece lo suficientemente rápido por lo que eventualmente mueren.

# Bibliografía


[1] Lopez, L. S. A. M. (2011). INTRODUCCIÓN A LA VIDA ARTIFICIAL Y AUTÓMATAS CELULARES.

[2] Wuensche, A., & Lesser, M. *(1992). The Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata (2nd edition).* Perseus Books.

[3] J. Martínez, G., Adamatzky, A., Chen, B., Chen, F., & Mora, J. (2010). *imple networks on complex cellular automata: From de Bruijn diagrams to jump-graphs.* (1.a ed.) [Pdf]. Springer.