Instituto Politécnico Nacional

Escuela Superior De Computo
ESCOM

REPORTE PROGRAMA 1

David Baltazar Real

Grupo: 3CM19

Asignatura: Computing Selected Topics

Profesor: Juarez Martinez Genaro

Fecha de entrega: 6 de Noviembre del 2022

Índice general

Índice de figuras

Capítulo 1

Introduccion

John Von Neumann investigó la cuestión del origen de la vida y trató de diseñar una máquina capaz de reproducirse. Esta idea llevo a a Von Neumann a inventar un sistema denominado Automatas Celulares, estos son capaces de construir cualquier automata a partir de un conjunto apropiado de instrucciones.
El Juego de la vida es un autómata celular diseñado por el matemático británico John Horton Conway en 1970. Es un juego de cero jugadores, en el que su evolución es determinada por un estado inicial, sin requerir intervención adicional. Se considera un sistema Turing completo que puede simular cualquier otra Máquina de Turing.
Desde su publicación, ha atraído mucho interés debido a la gran variabilidad de la evolución de los patrones. Se considera que el Juego de la vida es un buen ejemplo de emergencia y autoorganización. Es interesante para científicos, matemáticos, economistas y otros observar cómo patrones complejos pueden provenir de la implementación de reglas muy sencillas.

Capítulo 2

Codigos

2.1. index.html

```
1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
8   <title>Conway's Game Of Life</title>
9   <link rel="stylesheet" href="./styles.css">
10  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
11  </head>
12
13  <body>
14
15  <div class="container">
16  <div id="canvas-scroll" class="left">
17  <canvas id="canvas" style="border: 1px solid blue;"></canvas>
18  </div>
19
20  <div class="right">
21  <fieldset class="parametros">
22
23  <legend>Parametros</legend>
24  <div class="parametros-section">
25  <div class="parametro">
26  <label for="canvas-size">Tamaño del tablero(celulas):</label>
27  <input type="number" name="canvas-size" id="canvas-size">
28  </div>
29
30  <div class="parametro">
31  <label for="cell-size">Tamaño celula(px):</label>
32  <input type="range" name="cell-size" id="cell-size" value="5" min=
        "1" max="10" step="1">
33  </div>
```

```
34
35    <div class="parametro">
36    <label for="frame-rate">FPS:</label>
37    <input type="range" name="frame-rate" id="frame-rate" value="30"
          min="10" max="60" step="5">
38    </div>
39
40    <div class="parametro">
41    <label for="life-percent">Probabilidad de vida:</label>
42    <input type="number" name="life-percent" id="life-percent" value=
          "0.05" min="0" max="1"
43    step=".01">
44    </div>
45
46    <div class="parametro">
47    <label for="color-vivo">Color Estado Vivo:</label>
48    <input type="color" name="color-vivo" id="color-vivo">
49    </div>
50
51    <div class="parametro">
52    <label for="color-muerto">Color Estado Vivo:</label>
53    <input type="color" name="color-muerto" id="color-muerto">
54    </div>
55
56    <div class="parametro">
57    <label for="file-selector">Cargar Preset:</label>
58    <input type="file" name="file-selector" id="file-selector">
59    </div>
60    </div>
61
62    <div id="parametros-section">
63    <div class="parametro">
64    <label for="s_min">S min:</label>
65    <input type="number" name="s_min" id="s_min">
66    </div>
67
68    <div class="parametro">
69    <label for="s_max">S max:</label>
70    <input type="number" name="s_max" id="s_max">
71    </div>
72
73    <div class="parametro">
74    <label for="b_min">B min:</label>
75    <input type="number" name="b_min" id="b_min">
76    </div>
77
78    <div class="parametro">
79    <label for="b_max">B max:</label>
80    <input type="number" name="b_max" id="b_max">
81    </div>
82
83
84    </div>
85
```

```
86    </fieldset>
87
88    <fieldset class="controles">
89    <legend>Controles</legend>
90    <button id="new-game">New Game</button>
91    <button id="update-rules">Update Rules</button>
92    <button id="pause-play">Pause/Play</button>
93    <button id="next-generation">Next Generation</button>
94    <button id="load-file">Load File</button>
95    <button id="save-file">Save File</button>
96    </fieldset>
97
98    <fieldset class="datos">
99    <legend>Datos</legend>
100   <p>Generacion: <span id="generacion"></span></p>
101   <p>Poblacion: <span id="poblacion"></span></p>
102   </fieldset>
103
104   <fieldset class="graficas">
105   <legend>Grafica</legend>
106   <div id="chart"></div>
107   </fieldset>
108   </div>
109   </div>
110   <script type="module" src="./js/main.js"></script>
111   </body>
112
113   </html>
```

2.2. styles.css

```
1  #canvas-scroll {
2    max-height: 550px;
3    height: 550px;
4    max-width: 600px;
5    width: 600px;
6    overflow: scroll;
7  }
8
9  .container {
10   display: grid;
11   grid-template-columns: 1fr 1fr;
12 }
13
14 .parametros {
15   display: grid;
16   grid-template-columns: 1fr 1fr;
17 }
18
19 .parametro {
20   margin: 10px;
```

```
21 }
22
23
24 .parametro input {
25   width: 60px;
26 }
27
28 .parametro #frame-rate {
29   width: 160px
30 }
31
32 .parametro #file-selector {
33   width: 100%;
34 }
```

2.3. main.js

```
1   import Cell from "./Cell.js";
2   /****************/
3   /** VARIABLES **/
4   /****************/
5
6   // Canvas
7   const canvas = document.getElementById('canvas');
8   const ctx = canvas.getContext('2d');
9   var colorVivo;
10  var colorMuerto;
11  // Parametros
12  const canvasSizeElem = document.getElementById('canvas-size');
13  const cellSizeElem = document.getElementById('cell-size');
14  const frameRateElem = document.getElementById('frame-rate');
15  const lifePercentElem = document.getElementById('life-percent');
16  const sMinElem = document.getElementById('s_min');
17  const sMaxElem = document.getElementById('s_max');
18  const bMinElem = document.getElementById('b_min');
19  const bMaxElem = document.getElementById('b_max');
20  const colorVivoElem = document.getElementById('color-vivo');
21  const colorMuertoElem = document.getElementById('color-muerto');
22  const fileSelector = document.getElementById('file-selector');
23  // Botones
24  const newGameBtn = document.getElementById('new-game');
25  const updateBtn = document.getElementById('update-rules');
26  const playBtn = document.getElementById('pause-play');
27  const nextGenBtn = document.getElementById('next-generation');
28  const saveFileBtn = document.getElementById('save-file');
29  const loadFileBtn = document.getElementById('load-file');
30  // Datos
31  const poblacionElem = document.getElementById('poblacion');
32  const generacionElem = document.getElementById('generacion');
33  // Valores
34  var canvasSize;
```

```
35    var cellSize;
36    var frameRate;
37    var lifePercent;
38    var sMin, sMax;
39    var bMin, bMax;
40    var poblacion, generacion;
41    // Variables logicas del tablero
42    var tablero;
43    var playing;
44    var gameInterval;
45    // Grafica
46    const lineDiv = document.getElementById('chart');
47    var layout = {
48      title: {
49        text: 'Densidad de poblacion',
50        font: {
51          family: 'Courier New, monospace',
52          size: 16
53        },
54        xref: 'paper',
55        x: 0.05
56      },
57      xaxis: {
58        title: {
59          text: 'Generacion',
60          font: {
61            family: 'Courier New, monospace',
62            size: 18
63          }
64        },
65      },
66      yaxis: {
67        title: {
68          text: 'Poblacion',
69          font: {
70            family: 'Courier New, monospace',
71            size: 18
72          }
73        }
74      }
75
76    }
77    // Lector de archivos
78    var reader = new FileReader;
79
80    /**********************/
81    /** EVENT LISTENERS **/
82    /**********************/
83    document.addEventListener('DOMContentLoaded',
          loadGameOfLifeDefaultValues);
84    newGameBtn.addEventListener('click', newGame);
85    updateBtn.addEventListener('click', updateRules);
86    playBtn.addEventListener('click', play);
87    nextGenBtn.addEventListener('click', nextGen);
```

```
88    canvas.addEventListener('mousedown', e => cambiarEstadoCelula(e));
89    cellSizeElem.addEventListener('change', updateRules);
90    saveFileBtn.addEventListener('click', saveFile);
91    loadFileBtn.addEventListener('click', loadFile);
92    /***************/
93    /** FUNCIONES **/
94    /***************/
95    // -------------------------------
96    // ---- Funciones principales ----
97    // -------------------------------
98    function loadGameOfLifeDefaultValues() {
99      canvasSizeElem.value = 100;
100     cellSizeElem.value = 5;
101     frameRateElem.value = 30;
102     lifePercentElem.value = 0.05;
103     sMinElem.value = 2;
104     sMaxElem.value = 3;
105     bMinElem.value = 3;
106     bMaxElem.value = 3;
107     colorVivoElem.value = '#ffffff';
108     colorMuertoElem.value = '#000000';
109
110     playing = false;
111
112     updateRules();
113   }
114
115   function newGame() {
116
117     // Paramos el juego si ya hay uno corriendo y reiniciamos los
          datos
118     if (playing === true) {
119       play();
120     }
121     // Reiniciamos los datos
122     generacion = 0;
123     poblacion = 0;
124     Plotly.purge(lineDiv);
125     // Creamos el tablero
126     tablero = new Array(canvasSize);
127     for (let i = 0; i < canvasSize; i++) {
128       tablero[i] = new Array(canvasSize);
129     }
130     // Llenamos el tablero de celulas
131     let estado;
132     for (let y = 0; y < canvasSize; y++) {
133       for (let x = 0; x < canvasSize; x++) {
134         estado = Math.random() < lifePercent ? 1 : 0;
135         poblacion += estado;
136         tablero[y][x] = new Cell(x, y, estado, sMin, sMax, bMin, bMax
              );
137       }
138     }
139     // Agregamos a los vecinos
```

```
140       for (let y = 0; y < canvasSize; y++) {
141         for (let x = 0; x < canvasSize; x++) {
142           tablero[y][x].agregarVecinos(tablero);
143         }
144       }
145       // Imprimimos los resultados
146       imprimirTablero();
147       imprimirDatos();
148       // Graficamos
149       Plotly.plot(lineDiv, [{
150         y: [poblacion],
151         x: [generacion],
152         name: 'Densidad de poblacion',
153         type: 'line'
154       }],layout);
155     }
156
157     function updateRules() {
158       var needNewGame = false; // Variable para comprobar la necesidad
              de hacer un nuevo juego
159       // Detenemos el juego anterior
160       if (playing === true) {
161         play();
162       }
163       // Comprobamos el cambio del tamaño del canvas y de las reglas
              del juego
164       if (canvasSizeElem.value != canvasSize || sMinElem.value != sMin
              || sMaxElem.value != sMax || bMinElem.value != bMin ||
              bMaxElem.value != bMax) {
165         needNewGame = true;
166       }
167
168
169       // Actualizamos los valores
170       canvasSize = canvasSizeElem.value;
171       cellSize = cellSizeElem.value;
172       frameRate = frameRateElem.value;
173       lifePercent = lifePercentElem.value;
174       sMin = sMinElem.value;
175       sMax = sMaxElem.value;
176       bMin = bMinElem.value;
177       bMax = bMaxElem.value;
178       colorVivo = colorVivoElem.value;
179       colorMuerto = colorMuertoElem.value;
180       // Limpiamos el canvas
181       canvas.width = canvasSize * cellSize;
182       canvas.height = canvasSize * cellSize;
183
184       // Si se cambio el tamaño del canvas o las reglas se crea un
              nuevo juego
185       if (needNewGame === true)
186       newGame();
187
188       imprimirTablero();
```

```
189   }
190
191   function play() {
192     playing = !playing;
193     if (playing) {
194       gameInterval = setInterval(nextGen, 1000 / frameRate);
195     } else {
196       clearInterval(gameInterval);
197     }
198   }
199
200   function nextGen() {
201     generacion += 1;
202     poblacion = 0;
203
204     for (let y = 0; y < canvasSize; y++) {
205       for (let x = 0; x < canvasSize; x++) {
206         tablero[y][x].siguienteCiclo();
207         poblacion += tablero[y][x].estado;
208       }
209     }
210
211     for (let y = 0; y < canvasSize; y++) {
212       for (let x = 0; x < canvasSize; x++) {
213         tablero[y][x].mutar();
214       }
215     }
216
217     imprimirTablero();
218     imprimirDatos();
219     Plotly.extendTraces(lineDiv, {
220       y: [[poblacion]],
221       x: [[generacion]]
222     }, [0]);
223   }
224
225   function cambiarEstadoCelula(e) {
226     let coordenadas = getCursorPosition(e);
227     // Actualiza el contador de poblacion
228     if (tablero[coordenadas[1]][coordenadas[0]].estado === 1) {
229       poblacion -= 1;
230     } else {
231       poblacion += 1;
232     }
233     // Invierte el estado de la celula
234     tablero[coordenadas[1]][coordenadas[0]].invertirEstado();
235     // Imprime los resultados
236     imprimirTablero();
237     imprimirDatos();
238   }
239
240   function saveFile() {
241     guardarArchivo(guardarJuego(), introducirNombreTxt());
242   }
```

```
243
244   function loadFile() {
245     let file = fileSelector.files[0];
246     reader.readAsText(file);
247
248     reader.onload = cargarPreset;
249   }
250
251   // -------------------------------
252   // ---- Funciones Secundarias ----
253   // -------------------------------
254
255   function cargarPreset() {
256     let result = reader.result;
257     let lineas = result.split('\n');
258
259     for (let i = 0; i < lineas.length - 1; i++) {
260       if (i === 0) {
261         let cabecera = lineas[i].split(' ');
262
263         canvasSize = parseInt(cabecera[0]);
264         canvasSizeElem.value = canvasSize;
265
266         sMin = parseInt(cabecera[1]);
267         sMinElem.value = sMin;
268
269         sMax = parseInt(cabecera[2]);
270         sMaxElem.value = sMax;
271
272         bMin = parseInt(cabecera[3]);
273         bMinElem.value = bMin;
274
275         bMax = parseInt(cabecera[4]);
276         bMaxElem.value = bMax;
277
278         newGamePreset();
279       } else {
280         let linea = lineas[i].split('');
281         for (let x = 0; x < canvasSize; x++) {
282           poblacion += parseInt(linea[x]);
283           tablero[i - 1][x] = new Cell(x, i - 1, parseInt(linea[x]),
                sMin, sMax, bMin, bMax);
284         }
285       }
286     }
287
288     // Agregamos a los vecinos
289     for (let y = 0; y < canvasSize; y++) {
290       for (let x = 0; x < canvasSize; x++) {
291         tablero[y][x].agregarVecinos(tablero);
292       }
293     }
294     // Imprimimos los resultados
295     imprimirTablero();
```

```
296       imprimirDatos();
297       // Graficamos
298       Plotly.plot(lineDiv, [{
299         y: [poblacion],
300         x: [generacion],
301         type: 'line'
302       }]);
303     }
304
305     function guardarArchivo(contenido, nombre) {
306       if (nombre === './CANCEL')
307       return
308       const a = document.createElement('a');
309       const archivo = new Blob([contenido], { type: 'text/plain' });
310       const url = URL.createObjectURL(archivo);
311       a.href = url;
312       a.download = nombre;
313       a.click();
314       URL.revokeObjectURL(url);
315     }
316
317     function guardarJuego() {
318       let contenido;
319       // Agregamos las cabeceras del jueg(canvasSize, s min, s max, b
              min, b max)
320       contenido = canvasSize.toString() + " " + sMin.toString() + " " +
               sMax.toString() + " " + bMin.toString() + " " + bMax.toString
              () + "\n";
321       // Agregamos el contenido del tablero
322       for (let y = 0; y < canvasSize; y++) {
323         for (let x = 0; x < canvasSize; x++) {
324           contenido += tablero[y][x].estado.toString();
325         }
326         contenido += "\n";
327       }
328       return contenido
329     }
330
331     function introducirNombreTxt() {
332       let nombre = prompt('Introduce el nombre de la configuracion:');
333       if (nombre == null || nombre == '') {
334         nombre = './CANCEL';
335       } else {
336         nombre += ".txt";
337       }
338
339       return nombre;
340     }
341
342     function imprimirDatos() {
343       poblacionElem.innerText = poblacion;
344       generacionElem.innerText = generacion;
345     }
346
```

```
347   function imprimirTablero() {
348     canvas.width = canvas.width;
349     canvas.height = canvas.height;
350     for (let y = 0; y < canvasSize; y++) {
351       for (let x = 0; x < canvasSize; x++) {
352         ctx.fillStyle = tablero[y][x].estado == 1 ? colorVivo :
              colorMuerto;
353         ctx.fillRect(x * cellSize, y * cellSize, cellSize, cellSize);
354       }
355     }
356   }
357
358   function getCursorPosition(event) {
359     let rect = canvas.getBoundingClientRect();
360     let x = Math.floor((event.clientX - rect.left) / cellSize);
361     let y = Math.floor((event.clientY - rect.top) / cellSize);
362
363     return [x, y];
364   }
365
366   function newGamePreset() {
367     // Paramos el juego si ya hay uno corriendo y reiniciamos los
          datos
368     if (playing === true) {
369       play();
370     }
371     // Reiniciamos los datos
372     generacion = 0;
373     poblacion = 0;
374     Plotly.purge(lineDiv);
375     // Creamos el tablero
376     tablero = new Array(canvasSize);
377     for (let i = 0; i < canvasSize; i++) {
378       tablero[i] = new Array(canvasSize);
379     }
380   }
```

2.4. Cell.js

```
1  export default class Cell {
2    constructor(x, y, estado, sMin, sMax, bMin, bMax) {
3      this.x = x;
4      this.y = y;
5      this.estado = estado;
6      this.estadoSig = this.estado;
7
8      this.sMin = sMin;
9      this.sMax = sMax;
10     this.bMin = bMin;
11     this.bMax = bMax;
12
```

```
13      this.vecinos = [];
14    }
15
16    agregarVecinos(tablero) {
17      let xVecino;
18      let yVecino;
19      let N = tablero.length;
20
21      for (let i = -1; i < 2; i++) {
22        for (let j = -1; j < 2; j++) {
23          xVecino = (this.x + j + N) % N;
24          yVecino = (this.y + i + N) % N;
25
26          if (i != 0 || j != 0) {
27            this.vecinos.push(tablero[yVecino][xVecino]);
28          }
29        }
30      }
31    }
32
33    siguienteCiclo() {
34      let suma = 0;
35      for (let i = 0; i < this.vecinos.length; i++) {
36        suma += this.vecinos[i].estado;
37      }
38
39      this.estadoSig = this.estado;
40
41      if (suma < this.sMin || suma > this.sMax) {
42        this.estadoSig = 0;
43      }
44      if (suma >= this.bMin && suma <= this.bMax) {
45        this.estadoSig = 1;
46      }
47    }
48
49    mutar() {
50      this.estado = this.estadoSig;
51    }
52
53    invertirEstado() {
54      if (this.estado === 1) {
55        this.estado = 0;
56      } else {
57        this.estado = 1;
58      }
59    }
60  }
```

Capítulo 3

Capturas

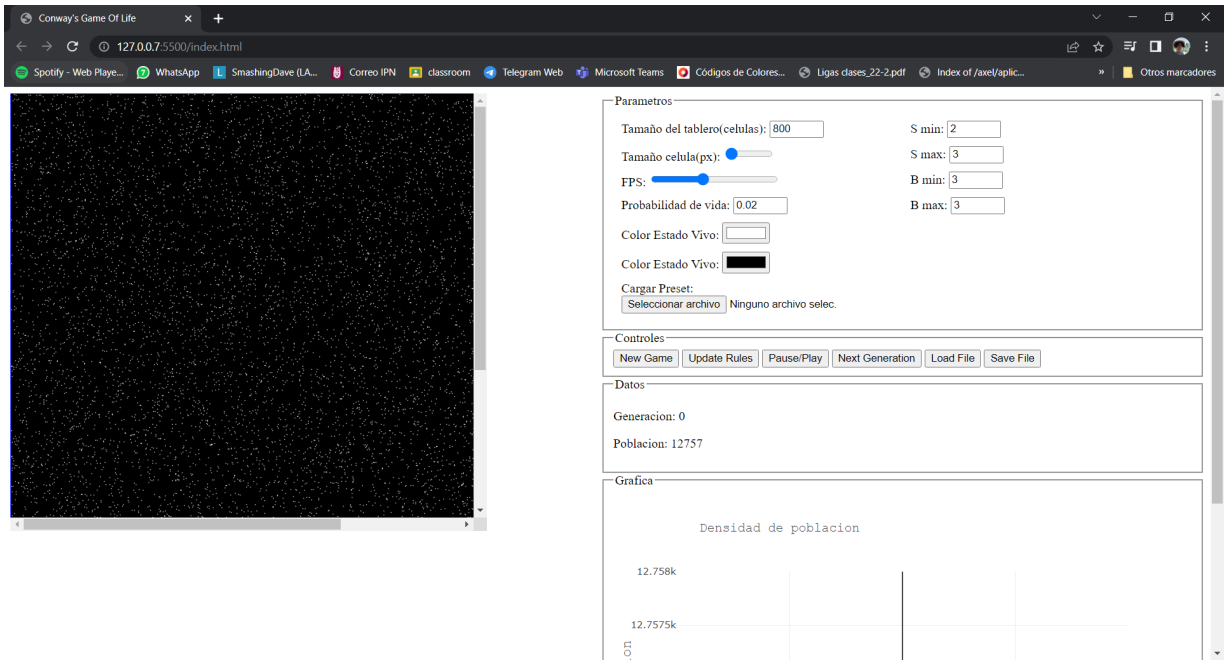

Figura 3.1: Prueba de un espacio 800x800 con probabilidad de vida de 0.02

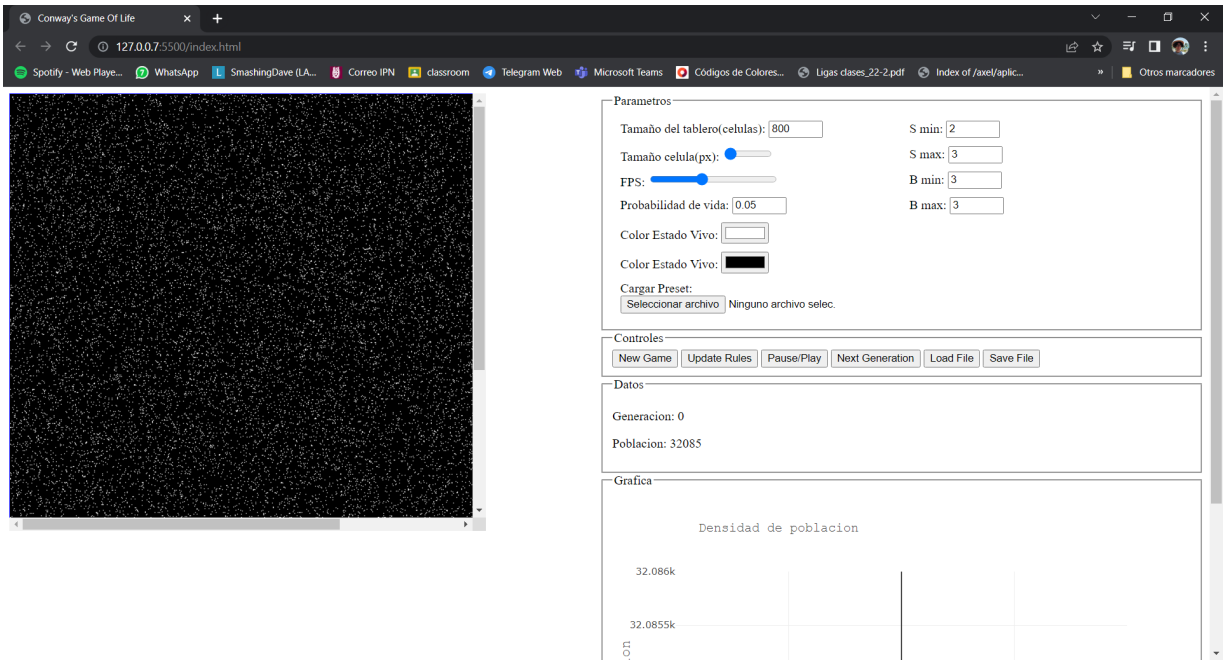

Figura 3.2: Prueba de un espacio 800x800 con probabilidad de vida de 0.05

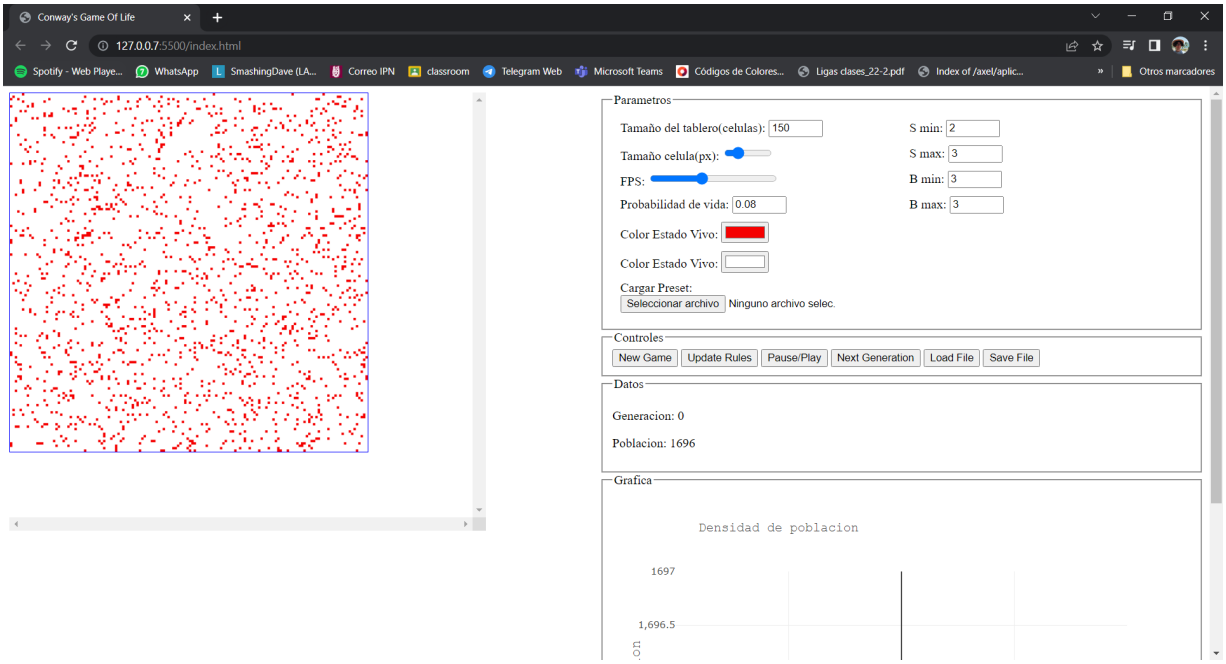

Figura 3.3: Prueba de colores de un espacio 150x150

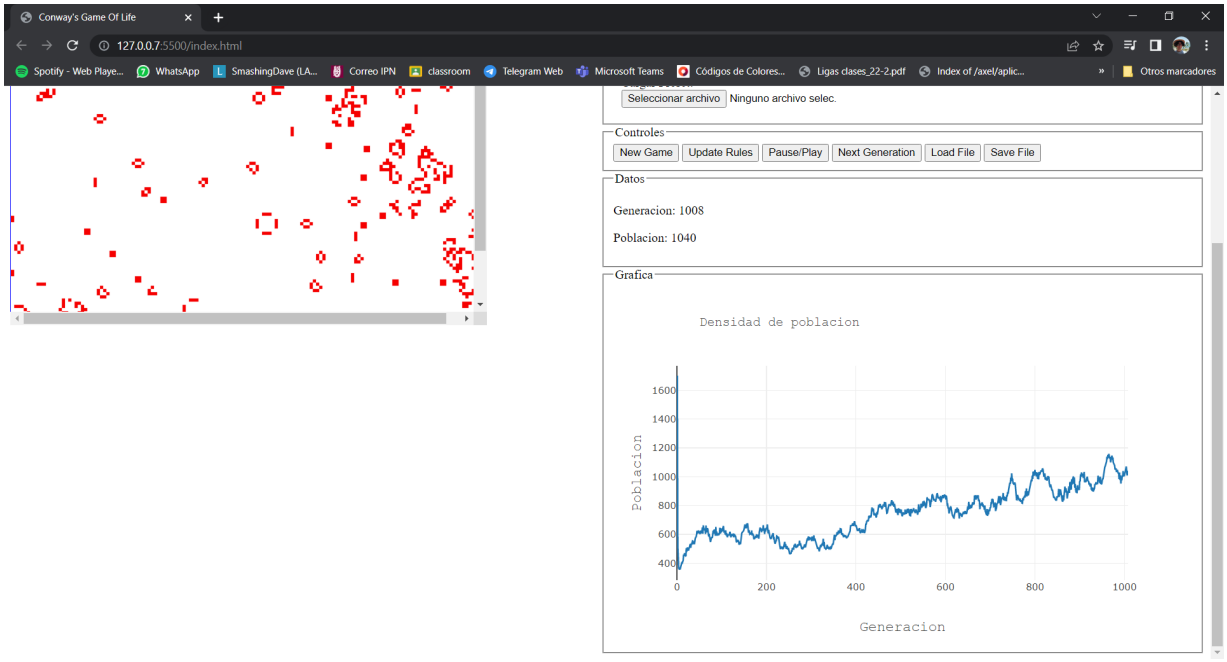

Figura 3.4: Prueba de la grafica de densidad para un espacio de 150x150 en la generacion 1008

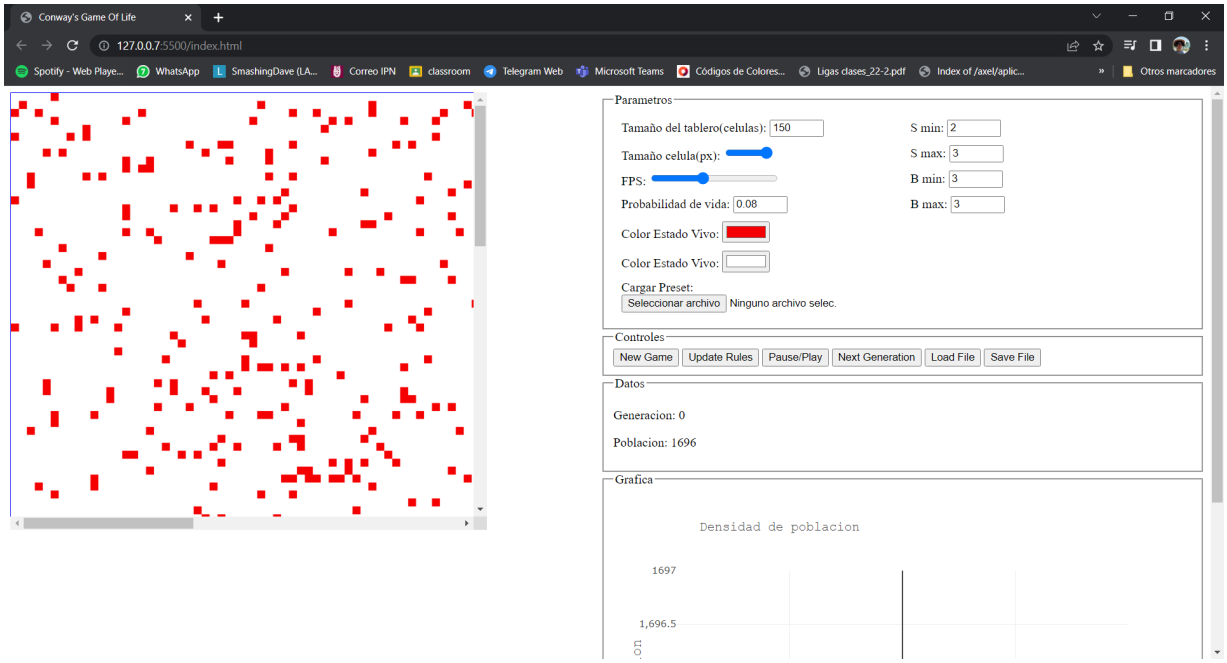

Figura 3.5: Prueba del aumento de tamaño(px) de la celula en un espacio de 150x150

Capítulo 4

Conclusiones

A lo largo de esta practica se hizo uso de los conocimientos aprendidos en clase para poder programar una version del Juego de la vida de Conway y poder observar el comportamiento de uno de los automatas celulares mas famosos que existen, asi como las pruebas de las diferentes figuras que se pueden formar y su comportamiento que va desde lo basico que es quedarse de manera estatica(Beehive) pasando por el movimiento en el espacio celular(Glider) hasta llegar un comportamiento complejo el cual podria ser como una figura que es capas de generar otras figuras constantemente(Gosper's glider gun) y el como a partir de esto podemos crear formas que simulen otro automata.

Bibliografía

[1] Gardner, M. (1971). Mathematical games: on cellular automata, self-reproduction, the Garden of Eden and the game"life". Sci. Am., 224, 112-117.

[2] Adamatzky, A. (Ed.). (2010). Game of life cellular automata. London: Springer.

[3] Martínez, G. J., Adamatzky, A., McIntosh, H. V. (2010). Localization dynamics in a binary two-dimensional cellular automaton: the Diffusion Rule. In Game of Life Cellular Automata (pp. 291-315). Springer.

[4] Martínez, G. J., Adamatzky, A., Seck-Tuoh-Mora, J. C. (2022). Some Notes About the Game of Life Cellular Automaton. In The Mathematical Artist (pp. 93-104). Springer.