Computing selected topics

- Prácticas primer parcial -

Grupo 3CM8

Vargas Romero Erick Efraín
Prof. Juárez Martínez Genaro

Instituto Politécnico Nacional
Escuela Superior de Cómputo
Juan de Dios Bátiz, nueva industrial Vallejo
07738 ciudad de México

Contents

Chapter 1

Máquina de Turing

1.1 Introducción

Sabemos que en la computación existen diversos tipos de problemas, por tal razón es que se estudian todos estos problemas para poder asignarles una categoría.

Existe además la teoría de los problemas indecidibles. Estos problemas indecidibles es un problema de decisión para el cual es imposible construir un algoritmo que siempre conduzca a una respuesta de si o no que sea completamente correcta. Podemos decir que un problema de decisión es una pregunta arbitraria de si o no en un conjunto finito de entradas. Los valores de entrada pueden ser desde números naturales, hasta datos de otro tipo, como cadenas de un lenguaje formal.

Existe una infinidad de problemas indecidibles, algunos algunas áreas en las que ocurre esto es:

- En lógica
- Máquinas abstractas
- Matrices
- Física cuántica
- Teoría de combinación de grupos

Siendo así, se requieren herramientas que nos permitan determinar ciertas cuestiones sobre los problemas que son indecidibles o bien intratables. Como resultado de esto es necesario reconstruir nuestra teoría sobre la indecibilidad, esto es sin basarnos en un lenguaje de programación sino en un modelo computacional simple el cual es la máquina de Turing.

1.2 Definición

La máquina de Turing es un dispositivo que manipula símbolos sobre una cinta de acuerdo a una tabla de reglas. A pesar de ser bastante símple, una máquina de Turing puede simular la lógica de un algoritmo de computacional.

1.2.1 De manera informal

La máquina de Turing opera mecánicamente sobre una cinta. En la cinta hay símbolos que la máquina puede leer y escribir, uno a la vez, usando un cabezal el cual puede leer o escribir en la cinta. Las operaciones estan determinadas por un conjunto de instrucciones elementales o básicas, este conjunto es finito, por ejemplo, si se encuentra el estado q_i y el símbolo que se ha visto es el X_i entonces se escribe Y_i.

1.2.2 De manera formal

El modelo matemático de una máquina de Turing está formado por un alfabeto de entrada y uno de salida, un símbolo especial el cual es llamado símbolo blanco, un conjunto de estados finitos y un conjunto de transiciones entre dichos estados. El funcionamiento de la máquina de Turing se basa en una función de transición, la cual recibe un estado inicial y una cadena de caracteres. La máquina de Turing va leyendo de celda en celda una a la vez, borrando el símbolo en el que se encuentra actualmente el cabezal y escribiendo un nuevo símbolo perteneciente al alfabeto de salida, para después desplazar el cabezal a la izquierda o bien a la derecha. Esto se repetirá según se indique en la función de transición para finalmente detenerse en un estado final o bien de aceptación. Una máquina de Turing con una sola cinta puede definirse como una séptupla es decir

$$M = (Q, \Sigma, \Gamma, s, \epsilon, F, \delta) \tag{1.1}$$

dónde:

- Q es un conjunto de estados finitos
- Σ es un conjunto finito de símbolos todos diferentes del espacio en blanco, el cual es denominado alfabeto de máquina o de entrada
- Γ es un conjunto de símbolos de la cinta denominado alfabeto de la cinta ($\Sigma \subseteq \Gamma$)
- s $\in$ Q es el estado inicial
- $\epsilon \in \Gamma$ es un símbolo denominado blanco, y es el único símbolo que se puede repetir infinitas veces.
- F $\subseteq$ Q es el conjunto de estados finales

- $\delta : Q$ x $\Gamma \rightarrow Q$ x Γ x $\{L, R\}$ es una función parcial denominada función de transición, sonde L es un movimiento a la izquierda y R es un movimiento a la derecha

1.3 Componentes

1.3.1 Cinta

La cinta está dividida en celdas, una la lado de la otra. Cada una de estas celdas contiene algún símbolo perteneciente de algún alfabeto finito. El alfabeto contiene un símbolo especial el cual es el símbolo en blanco y uno o más símbolos adicionales. La cinta también suponemos que es infinita tanto a la izquierda como a la derecha. Las celdas que no se hayan escrito con anterioridad se asume que contiene el símbolo en blanco.

1.3.2 Cabezal

El cabezal puede tanto leer como escribir símbolos en la cinta además de mover la cinta a la izquierda o a la derecha. El movimiento de la cinta solo se realiza de celda en celda una y solo una vez. En ocasiones la cabeza es la que se mueve y la cinta es fija.

1.3.3 Registro de estado

El registro de estado almacena el estado de la máquina de Turing, uno de los estados finitos. Siempre hay un estado inicial con el que el registro de la máquina de Turing inicia.

1.3.4 Tabla

La máquina de Turing debe de tener una tabla finita de instrucciones, la cual suele nombrarse también como tabla de acción o bien función de transición. Las instrucciones generalmente son cinco tuplas es decir $q_i a_j \rightarrow q_{i1} a_{j1} d_k$ que dado el estado q_i en la máquina se encuentra actualmente el símbolo a_j que está leyendo la cinta, es decir el símbolo que está abajo del cabezal le indica a la máquina que debe hacer después en secuencia.

1. Borrar o escribir un símbolo (reemplazar a_j por a_{j1})
2. Mueve el cabezal el cual es descrito por d_k y que puede tener los valores Ĺṕara moverse a la izquierda o Ŕṕara moverse a la derecha o bien Ńṕara mantenerse en su lugar
3. Asume el mismo o el nuevo estado como prescrito es decir, ve al estado q_{i1}

1.4 Práctica

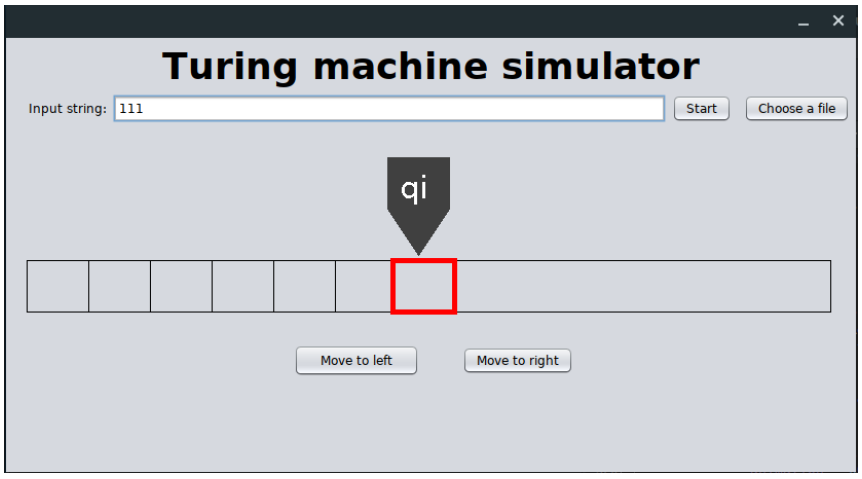
1.4.1 Descripción

Para esta práctica se ha creado una máquina de Turing la cual tiene como fin duplicar el número de unos que reciba de entrada. Siendo así podemos decir que esta máquina de Turing tiene un solo símbolo diferente del espacio en blanco en nuestro caso $\Sigma = \{1\}$. También debemos definir el alfabeto que tendrá la cinta $\Gamma = \{X\}$. Para esta máquina de Turing nuestro estado inicial lo denotaremos con $s = \{q_i\}$ nuestro estado final $F = \{q_f\}$
Para realizar esta práctica el lenguaje de programación elegido fué Java, además de hacer uso del entorno de desarrollo integrado Netbeans.

1.4.2 Pruebas

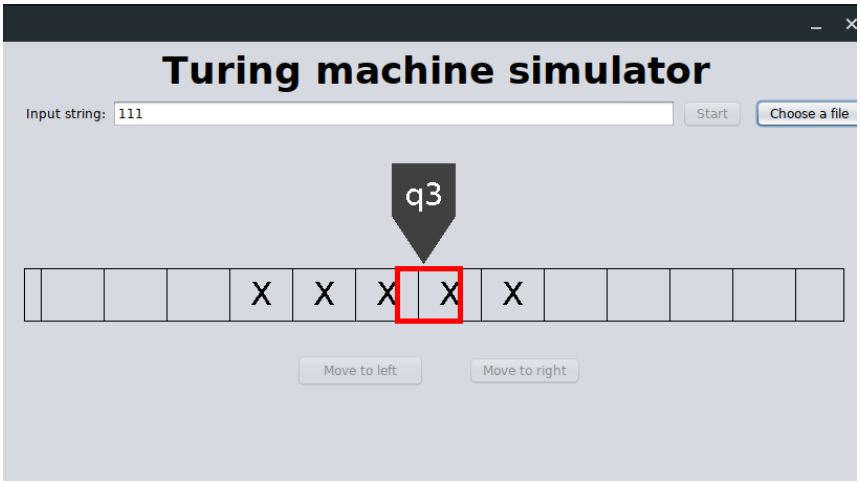
Este programa es bastante sencillo pero tiene ciertos aspectos que deben ser mostrados. Primeramente, este programa requiere de que sea ingresada una cadena únicamente de unos. Esta cadena será ingresada en el único campo de texto que tiene el programa. Tal y como se muestra a continuación

Figure 1.1: Interfaz general



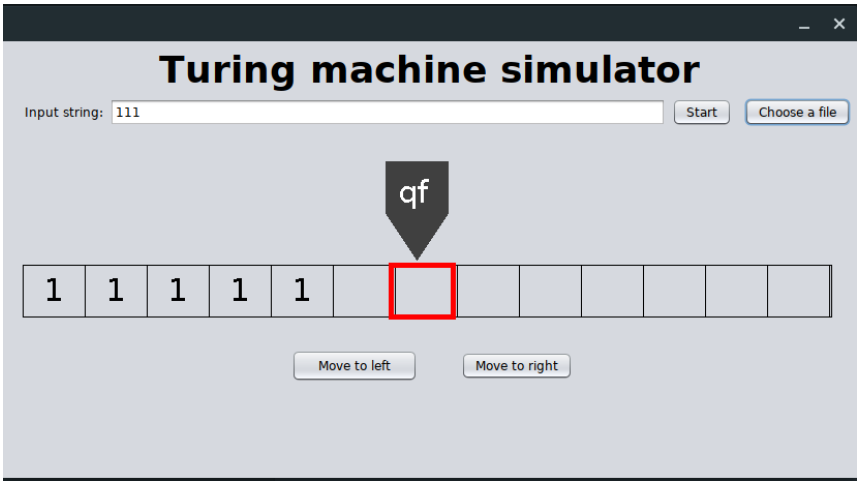
En la siguiente imagen podemos ver que el programa ya está en ejecución, la cadena que ha sido ingresada es "111" además de que ya hay algunas transiciones del autómata. En la cabeza podemos ver que se muestra el estado actual, además en la parte de abajo podemos ver a lo que apunta la cabeza de la máquina.

Figure 1.2: Ingreso de una cadena



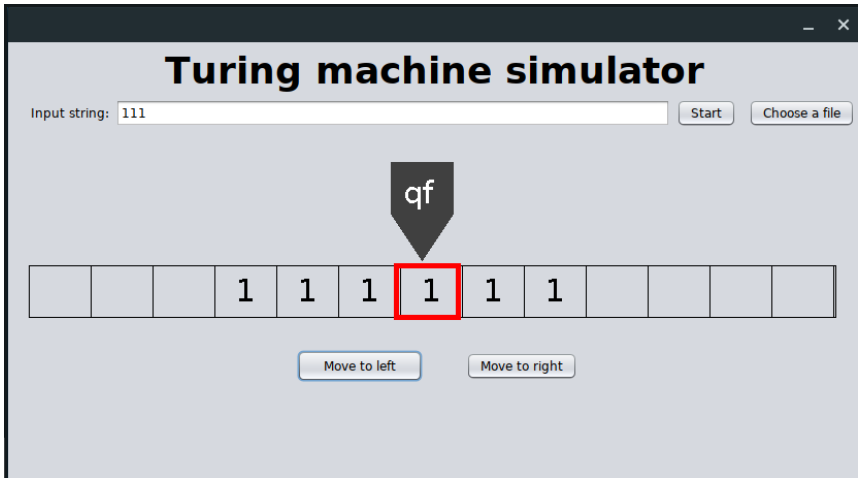
En esta figura podemos apreciar que la máquina ha terminado de ejecutar, ya ha duplicado la cadena de unos y se indica que ya está en un estado final.

Figure 1.3: Fin de ejecución



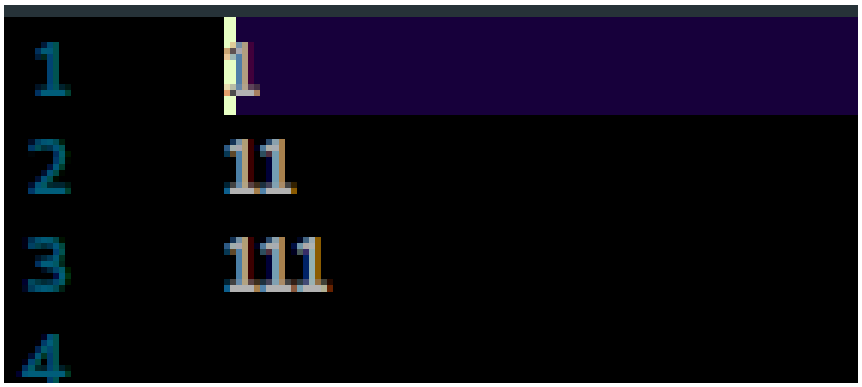
Como podemos observar abajo hay algunos botones, los cuales harán que se mueva la cinta a la derecha o a la izquierda, dependiendo de lo que se desee. Y así podemos efectivamente comprobar que la cadena ha sido duplicada.

Figure 1.4: Movimiento de la cinta



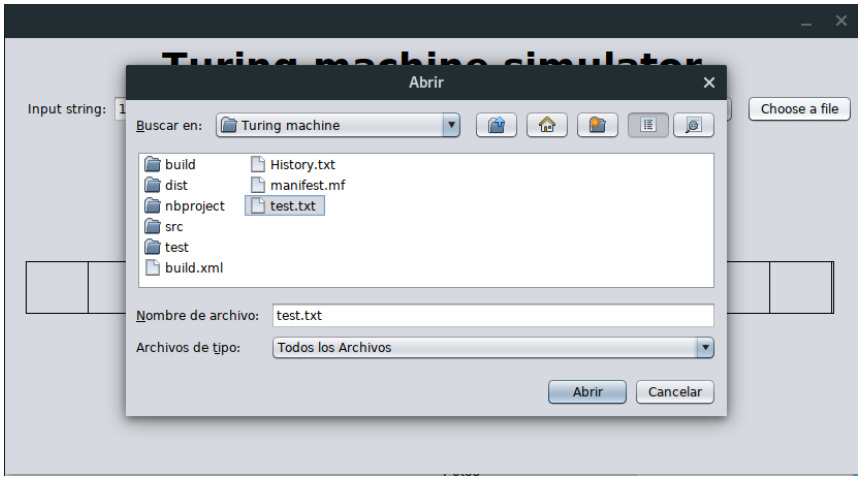
Este programa también puede leer archivos, con varias cadenas de unos. El programa seleccionará aleatoriamente una de esas cadenas, y será la que ejecutará. En la siguiente imagen podemos ver el contenido de un archivo, el cual tiene solo tres cadenas.

Figure 1.5: Archivo de prueba



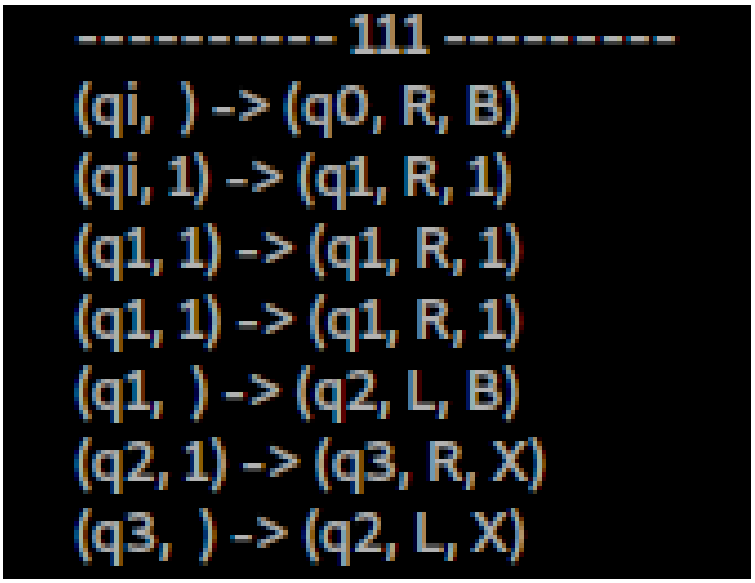
Seleccionamos el botón "Choose a file" el cual abrirá un JFileChooser y el usuario podrá elegir el archivo de texto que desee cargar. Finalmente seleccionamos el archivo y el programa elegirá la cadena al azar.

Figure 1.6: Carga del archivo



Finalmente, este programa también genera archivos, el programa generará un archivo llamado History, el cual contendrá el "historial" del programa, es decir, todas las transiciones que ocurrieron para determinar si una cadena fue o no válida.

Figure 1.7: Contenido History.txt



1.4.3 Código

A continuación se anexa el código correspondiente a la máquina de Turing. Como ya se ha mencionado el lenguaje utilizado para este simulador fué Java, siendo así tenemos código autogenerado por el entorno de desarrollo integrado Netbeans.

```
1  package turing.machine.UI;
2
3  import java.awt.BasicStroke;
4  import java.awt.Color;
5  import java.awt.Font;
6  import java.awt.Graphics;
7  import java.awt.Graphics2D;
8  import java.awt.Polygon;
9  import java.awt.Rectangle;
10 import java.awt.geom.Rectangle2D;
11 import java.io.BufferedReader;
12 import java.io.BufferedWriter;
13 import java.io.File;
14 import java.io.FileReader;
15 import java.io.FileWriter;
16 import java.util.ArrayList;
17 import java.util.Arrays;
18 import javax.swing.JFileChooser;
19 import sun.awt.windows.ThemeReader;
20
21 /**
22  * @author erick
23  */
24 public class Simulator extends javax.swing.JFrame {
25
26     /**
27      * *********** GLOBAL VARIABLES ************
28      */
29     /**
30      * ******** TURING MACHINE VARIABLES *******
31      */
32     private static final String q[] = {"qi", "q1", "q2", "q3", "q4"};
33     private static final Character t_symbol = 'X';
34     private static final Character symbol = '1';
35     private static final String qi = q[0];
36     private static final String qf = q[4];
37     private static final char B = ' ';
38     private String qc = qi;
39
40     private ArrayList<Character> tape;
41
42     /**
43      * *********** OTHER VARIABLES ************
44      */
45     private int animation = 0;
46     private static final char L = 'L';
47     private static final char R = 'R';
48     private static final char N = 'N';
49     private char move_to = N;
50
51     /**
52      * ********** END GLOBAL VARIBLES **********
53      */
```

```
54      /**
55       * Creates new form Simulator
56       */
57      public Simulator() {
58          initComponents();
59          this.setResizable(false);
60      }
61
62      private void enableButons(boolean enable){
63          BTNToLeft.setEnabled(enable);
64          BTNToRight.setEnabled(enable);
65          BTNStart.setEnabled(enable);
66      }
67      @Override
68      public void paint(Graphics g) {
69          super.paint(g);
70          System.out.println("animation to " + move_to + " contains " +
                animation);
71          Font font = new Font("Dialog", Font.PLAIN, 30);
72          g.setFont(font);
73          int Xi = 20;
74          int Xf = this.getWidth() - 45;
75
76          int Yi = 250;
77          int Yf = 50;
78
79          int middle = (Xf + Xi) / 2;
80          /**
81           * ****** BEGIN DRAWING TAPE *******
82           */
83
84          try {
85              //Drawing container
86              g.drawRect(Xi, Yi, Xf, Yf);
87
88              //Drawing cells and text
89              for (int i = -(Xf - 1) * 2, k = 0; i < Xf * 2; i += 3) {
90
91                  /**
92                   * Making each cell*
93                   */
94                  //Xi contains the size of each cell
95                  g.drawRect(Xi * i + animation, Yi, 0, Yf);
96                  g.setFont(font);
97
98                  if (i >= (Xi - 5) && k < tape.size() - 1) {
99                      //Setting text in each cell
100                     g.drawString(tape.get(k++) + "", Xi * i + 20 +
                            animation, (Yi + Yf - 15));
101                 }
102             }
103         } catch (Exception e) {
104             System.err.println(e);
```

```
105         }
106
107         /**
108          * ******* END DRAWING TAPE ********
109          */
110         /**
111          * ****** BEGIN DRAWING HEAD *******
112          */
113         int x_coordinates[] = {middle - 30, middle - 30, middle, middle
                + 30, middle + 30, middle - 30};
114         int y_coordinates[] = {Yi - 100, Yi - 50, +Yi - 5, Yi - 50, Yi -
                100, Yi - 100};
115         Polygon head = new Polygon(x_coordinates, y_coordinates,
                x_coordinates.length);
116         Graphics2D g2D = (Graphics2D) g;
117         g2D.setColor(Color.DARK_GRAY);
118         g2D.fillPolygon(head);
119         g2D.drawPolygon(head);
120         g2D.setColor(Color.WHITE);
121         g2D.drawString(qc, middle - 18, Yi - 60);
122
123         g2D.setColor(Color.RED);
124         g2D.setStroke(new BasicStroke(5.0f));
125         g2D.drawRect(middle - 25, Yi, 60, Yf);
126         /**
127          * ******* END DRAWING HEAD ********
128          */
129     }
130
131     private void move() {
132         try {
133             for (int i = 0; i < 60; i++) {
134                 animate();
135                 Thread.sleep(10);
136             }
137             Thread.sleep(300);
138         } catch (Exception e) {
139             System.err.println(e);
140         }
141     }
142
143     private void animate() {
144         if (move_to == 'L') {
145             animation++;
146         } else if (move_to == 'R') {
147             animation--;
148         }
149         repaint();
150     }
151
152     @SuppressWarnings("unchecked")
153     // <editor-fold defaultstate="collapsed" desc="Generated Code">
154     private void initComponents() {
```

```
155
156         jLabel1 = new javax.swing.JLabel();
157         jLabel2 = new javax.swing.JLabel();
158         TXTInputString = new javax.swing.JTextField();
159         BTNStart = new javax.swing.JButton();
160         BTNToRight = new javax.swing.JButton();
161         BTNToLeft = new javax.swing.JButton();
162         BTNChooseFIle = new javax.swing.JButton();
163
164         setDefaultCloseOperation(javax.swing.WindowConstants.
                EXIT_ON_CLOSE);
165         setMaximumSize(new java.awt.Dimension(1000, 1000));
166         setMinimumSize(new java.awt.Dimension(750, 452));
167
168         jLabel1.setFont(new java.awt.Font("Dialog", 1, 36)); // NOI18N
169         jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER
                );
170         jLabel1.setText("Turing machine simulator");
171         jLabel1.setToolTipText("");
172         jLabel1.setAlignmentX(0.5F);
173
174         jLabel2.setText("Input string:");
175
176         BTNStart.setText("Start");
177         BTNStart.addActionListener(new java.awt.event.ActionListener() {
178             public void actionPerformed(java.awt.event.ActionEvent evt)
                    {
179                 BTNStartActionPerformed(evt);
180             }
181         });
182
183         BTNToRight.setText("Move to right");
184         BTNToRight.addActionListener(new java.awt.event.ActionListener()
                {
185             public void actionPerformed(java.awt.event.ActionEvent evt)
                    {
186                 BTNToRightActionPerformed(evt);
187             }
188         });
189
190         BTNToLeft.setText("Move to left");
191         BTNToLeft.setMaximumSize(new java.awt.Dimension(122, 31));
192         BTNToLeft.setMinimumSize(new java.awt.Dimension(122, 31));
193         BTNToLeft.setPreferredSize(new java.awt.Dimension(122, 31));
194         BTNToLeft.addActionListener(new java.awt.event.ActionListener()
                {
195             public void actionPerformed(java.awt.event.ActionEvent evt)
                    {
196                 BTNToLeftActionPerformed(evt);
197             }
198         });
199
200         BTNChooseFIle.setText("Choose a file");
```

```
201         BTNChooseFIle.addActionListener(new java.awt.event.
                ActionListener() {
202             public void actionPerformed(java.awt.event.ActionEvent evt)
                    {
203                 BTNChooseFIleActionPerformed(evt);
204             }
205         });
206 
207         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
208         getContentPane().setLayout(layout);
209         layout.setHorizontalGroup(
210             layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                    .LEADING)
211             .addGroup(layout.createSequentialGroup()
212                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
213                     .addGroup(layout.createSequentialGroup()
214                         .addContainerGap()
215                         .addComponent(jLabel1, javax.swing.GroupLayout.
                                DEFAULT_SIZE, javax.swing.GroupLayout.
                                DEFAULT_SIZE, Short.MAX_VALUE))
216                     .addGroup(layout.createSequentialGroup()
217                         .addGap(22, 22, 22)
218                         .addComponent(jLabel2)
219                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
220                         .addComponent(TXTInputString, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 538, javax.swing
                                .GroupLayout.PREFERRED_SIZE)
221                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
222                         .addComponent(BTNStart)
223                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.UNRELATED)
224                         .addComponent(BTNChooseFIle)))
225                 .addContainerGap())
226             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout
                    .createSequentialGroup()
227                 .addGap(0, 0, Short.MAX_VALUE)
228                 .addComponent(BTNToLeft, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 122, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
229                 .addGap(42, 42, 42)
230                 .addComponent(BTNToRight)
231                 .addGap(275, 275, 275))
232         );
233         layout.setVerticalGroup(
234             layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                    .LEADING)
235             .addGroup(layout.createSequentialGroup()
236                 .addContainerGap()
237                 .addComponent(jLabel1)
```

```
238                     .addPreferredGap(javax.swing.LayoutStyle.
                            ComponentPlacement.RELATED)
239                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.BASELINE)
240                         .addComponent(jLabel2)
241                         .addComponent(TXTInputString, javax.swing.
                                GroupLayout.PREFERRED_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
242                         .addComponent(BTNStart)
243                         .addComponent(BTNChooseFIle))
244                     .addGap(216, 216, 216)
245                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.BASELINE)
246                         .addComponent(BTNToLeft, javax.swing.GroupLayout.
                                PREFERRED_SIZE, javax.swing.GroupLayout.
                                DEFAULT_SIZE, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
247                         .addComponent(BTNToRight))
248                     .addContainerGap(29, Short.MAX_VALUE))
249         );
250 
251         pack();
252     }// </editor-fold>
253 
254     private void BTNStartActionPerformed(java.awt.event.ActionEvent evt)
            {
255         animation = 60;
256         try {
257             //Getting the input string
258             tape = new ArrayList<Character>();
259             tape.add(B);
260             for (char c : TXTInputString.getText().trim().toCharArray())
                    {
261                 tape.add(c);
262             }
263             tape.add(B);
264             if (!tape.toString().isEmpty()) {
265                 transitionFunction();
266             }
267             enableButons(false);
268         } catch (Exception e) {
269             System.err.println(e.getLocalizedMessage());
270         }
271     }
272 
273     private void BTNToLeftActionPerformed(java.awt.event.ActionEvent evt
            ) {
274         new Thread(new Runnable() {
275             @Override
276             public void run() {
277 
278                 try {
```

```
279                     for (int i = 0; i < 60; i++) {
280                         move_to = R;
281                         animate();
282                         Thread.sleep(10);
283                     }
284                 } catch (Exception e) {
285                     System.err.println(e);
286                 }
287             }
288         }).start();
289     }
290 
291     private void BTNToRightActionPerformed(java.awt.event.ActionEvent
        evt) {
292         new Thread(new Runnable() {
293             @Override
294             public void run() {
295                 try {
296                     for (int i = 0; i < 60; i++) {
297                         move_to = L;
298                         animate();
299                         Thread.sleep(10);
300                     }
301                 } catch (Exception e) {
302                     System.err.println(e);
303                 }
304             }
305         }).start();
306     }
307 
308     private void BTNChooseFIleActionPerformed(java.awt.event.ActionEvent
         evt) {
309 
310         String str = "";
311         ArrayList<String> lines = new ArrayList<>();
312         try {
313             JFileChooser jFileChooser = new JFileChooser();
314             jFileChooser.showOpenDialog(this);
315             File file = jFileChooser.getSelectedFile();
316             if (file != null) {
317                 FileReader files = new FileReader(file);
318                 BufferedReader read = new BufferedReader(files);
319                 while ((str = read.readLine()) != null) {
320                     lines.add(str);
321                 }
322                 read.close();
323             }
324 
325             int rand = (int) (Math.random() * (lines.size() - 1));
326             animation = 60  ;
327             tape = new ArrayList<>();
328             tape.add(B);
329             String selected = lines.get(rand).trim();
```

```
330             TXTInputString.setText(selected);
331             for (char c : selected.toCharArray()) {
332                 tape.add(c);
333             }
334             tape.add(B);
335             if (!tape.toString().isEmpty()) {
336                 transitionFunction();
337             }
338             enableButons(false);
339         } catch (Exception e) {
340             System.err.println(e);
341         }
342 
343 
344     }
345 
346     public static void main(String args[]) {
347 
348         //<editor-fold defaultstate="collapsed" desc=" Look and feel
                setting code (optional) ">
349         /* If Nimbus (introduced in Java SE 6) is not available, stay
                with the default look and feel.
350          * For details see http://download.oracle.com/javase/tutorial/
                uiswing/lookandfeel/plaf.html
351          */
352         try {
353             for (javax.swing.UIManager.LookAndFeelInfo info : javax.
                    swing.UIManager.getInstalledLookAndFeels()) {
354                 if ("Nimbus".equals(info.getName())) {
355                     javax.swing.UIManager.setLookAndFeel(info.
                            getClassName());
356                     break;
357                 }
358             }
359         } catch (ClassNotFoundException ex) {
360             java.util.logging.Logger.getLogger(Simulator.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
361         } catch (InstantiationException ex) {
362             java.util.logging.Logger.getLogger(Simulator.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
363         } catch (IllegalAccessException ex) {
364             java.util.logging.Logger.getLogger(Simulator.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
365         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
366             java.util.logging.Logger.getLogger(Simulator.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
367         }
368         //</editor-fold>
369 
370         java.awt.EventQueue.invokeLater(new Runnable() {
371             public void run() {
372                 new Simulator().setVisible(true);
373             }
```

```
374         });
375     }
376
377     /**
378      * ************************************************************
379      * * * * * * * * * BEGIN TURING MACHINE CODE * * * * * * * * *
380      * ************************************************************
381      */
382     private void transitionFunction() throws Exception {
383         System.out.println("turing.machine.UI.Simulator.
                transitionFunction()");
384         writeHistory("---------- " + TXTInputString.getText() + "
                ----------");
385         new Thread(new Runnable() {
386             @Override
387             public void run() {
388                 //This make a undifined cicle
389                 boolean flag = true;
390                 //This variable store the position of the current symbol
391                 int i = 0;
392                 //This variable contains the actual state
393                 qc = qi;
394                 //This variable contains the current symbol
395                 Character c_symbol = B;
396
397                 while (flag) {
398                     System.out.println(flag);
399                     c_symbol = tape.get(i);
400                     move_to = N;
401                     if (qc == q[0]) {
402
403                         //If current symbol it's a '1'
404                         if (c_symbol == symbol) {
405                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q1, R, 1)");
406                             qc = q[1];
407                             tape.set(i++, symbol);
408                             move_to = R;
409                         } else if (c_symbol == B) {
410                             System.out.println("INITIAL");
411                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q0, R, B)");
412                             qc = q[0];
413                             tape.set(i++, B);
414                             move_to = R;
415                         }
416
417                     } else if (qc == q[1]) {
418
419                         //If current symbol it's a '1'
420                         if (c_symbol == symbol) {
421                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q1, R, 1)");
```

```
422                             qc = q[1];
423                             tape.set(i++, symbol);
424                             move_to = R;
425                         } //If current symbol it's a ' '
426                         else if (c_symbol == B) {
427                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q2, L, B)");
428                             qc = q[2];
429                             tape.set(i--, B);
430                             move_to = L;
431                         }
432                     } else if (qc == q[2]) {
433
434                         //If current symbol it's a '1'
435                         if (c_symbol == symbol) {
436                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q3, R, X)");
437                             qc = q[3];
438                             tape.set(i++, t_symbol);
439                             move_to = R;
440                         } //If current symbol it's a 'X'
441                         else if (c_symbol == t_symbol) {
442                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q2, L, X)");
443                             qc = q[2];
444                             tape.set(i--, t_symbol);
445                             tape.add(B);
446                             move_to = L;
447                         } else if (c_symbol == B) {
448                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q4, R, B)");
449                             qc = q[4];
450                             tape.set(i++, B);
451                             move_to = R;
452                         }
453                     } else if (qc == q[3]) {
454
455                         //If current symbol it's a 'X'
456                         if (c_symbol == t_symbol) {
457                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q3, R, X)");
458                             qc = q[3];
459                             tape.set(i++, t_symbol);
460                             move_to = R;
461                         } //If current symbol it's a ' '
462                         else if (c_symbol == B) {
463                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q2, L, X)");
464                             qc = q[2];
465                             //tape.add(B);
466                             tape.set(i--, t_symbol);
467                             //tape.add(B);
468                             move_to = L;
```

```
469                         }
470                     } else if (qc == q[4]) {
471
472                         //If current symbol it's a 'X'
473                         if (c_symbol == t_symbol) {
474                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (q4, R, 1)");
475                             qc = q[4];
476                             tape.set(i++, symbol);
477                             move_to = R;
478                         } //If current symbol it's a ' '
479                         else if (c_symbol == B) {
480                             writeHistory("(" + qc + ", " + c_symbol + ")
                                    -> (qf, R, B)");
481                             System.out.println("FINAL");
482                             flag = false;
483                             qc = "qf";
484                             move_to = R;
485                             move();
486                             enableButons(true);
487                             break;
488                         }
489                     }
490                     move();
491
492                 }
493             }
494
495         }).start();
496         ;
497
498         System.out.println("turing.machine.UI.Simulator.
                transitionFunction()");
499
500     }
501
502     /**
503      * ************************************************************
504      * * * * * * * * * END TURING MACHINE CODE * * * * * * * * * *
505      * ************************************************************
506      */
507     private void writeHistory(String state) {
508         try {
509             BufferedWriter writer = new BufferedWriter(new FileWriter("
                    History.txt", true));
510             writer.append("\n");
511             writer.append(state);
512             writer.close();
513         } catch (Exception e) {
514             System.err.println(e);
515         }
516     }
517
```

```
518     // Variables declaration - do not modify
519     private javax.swing.JButton BTNChooseFIle;
520     private javax.swing.JButton BTNStart;
521     private javax.swing.JButton BTNToLeft;
522     private javax.swing.JButton BTNToRight;
523     private javax.swing.JTextField TXTInputString;
524     private javax.swing.JLabel jLabel1;
525     private javax.swing.JLabel jLabel2;
526     // End of variables declaration
527 }
```

Chapter 2

Autómata celular

2.1 Introducción

Los autómatas celulares son modelos matemáticos que valga la redundancia, modelan sistemas dinámicos, los cuales evolucionan con el paso del tiempo. Los autómatas celulares fueron descubiertos por John von Neumann en la década de 1940, y fue descrito en su libro Theory of Self-reproducing Automata. John von Neumann tenía como objetivo modelar una máquina, que fuese capaz de auto replicarse, al intentar esto llego a un modelo matemático, el cual describe a dicha máquina con ciertas reglas sobre una red rectangular.

2.2 Definición

Como ya se ha mencionado anteriormente, un autómata celular es un modelo matemático para un sistema dinámico, este sistema evoluciona con el paso del tiempo. El autómata celular está compuesto por células o celdas las cuales adquieren ciertos valores o estados. Al ser este un sistema dinámico y al evolucionar a través del tiempo los estados o valores que tienen las células cambian de un instante a otro, esto en unidades de tiempo discreto, en otras palabras es posible hacer una cuantización. Siendo así, el cunjunto de células evolucionan según la expresión matemática, la cual evolucionará según los estados de las células vecinas, a esto se le conoce como regla de transición local.

2.3 Componentes

2.3.1 Un espacio rectangular

El autómata celular está definido ya sea en un espacio de dos dimensiones o bien en un espacio de n dimensiones, este es el espacio de evoluciones y cada una de las divisiones de este espacio es llamada célula.

2.3.2 Conjunto de estados

Los estados son finitos y cada elemento de la célula tomará un valor de este conjunto de estados. A cada vecindad diferente le corresponde un elemento del conjunto de estados.

2.3.3 Vecindades

Conjunto de contiguo de células cuya posición es relativa respecto a cada una de ellas. Como se mencionó anteriormente, a cada vecindad diferente le corresponde un estado diferente del conjunto de estados.

2.3.4 Función local

Es la regla de evolución que determina el comportamiento del autómata celular. Esta regla esta conformada por una célula central y sus vecindades. También esta define como debe cambiar de estado cada una de las células dependiendo de los estados de las vecindades anteriores. Esta función puede ser representada como una función algebraica o como un conjunto de ecuaciones.

2.4 Límites o fronteras

Podemos hacer una representación visual de los autómatas celulares, y para que podamos entenderlo de mejor manera es necesario mencionar los límites y las fronteras, del espacio en el cual existe el autómata celular.

2.4.1 Frontera abierta

Considera que todas las células fuera del espacio del autómata tienen un valor el cual es fijo.

2.4.2 Frontera reflectora

Las células fuera del espacio del autómata toman los valores que están dentro como si se tratase de un espejo.

2.4.3 Frontera periódica o circular

Las células que están en los límites o en la frontera interaccionan con sus vecinos inmediatos y con las células que están en el extremo opuesto del arreglo, como si el plano estuviese doblado a manera de cilindro.

2.4.4 Sin frontera

La representación de autómatas no tiene límite alguno, es infinito.

2.5 Práctica

2.5.1 Descripción

Para esta práctica se ha creado nuestro primer autómata celular, el cual cumple la función local del "Juego de la vida". El juego de la vida es un autómata celular que fue diseñado por el matemático John Horton Conway en 1970. Este se trata de un juego de cero jugadores, es decir, el estado de evolución está definido por el estado inicial y no requiere entrada de datos alguna posteriormente. El tablero de este juego es una matriz formada con células (espacios cuadrados) que se extienden por el infinito a toda dirección. Cada célula tiene ocho células vecinas, que son las que están más próximas a ella, incluidas las diagonales, de forma gráfica

	Célula central	

Las reglas para ese juego son las siguientes

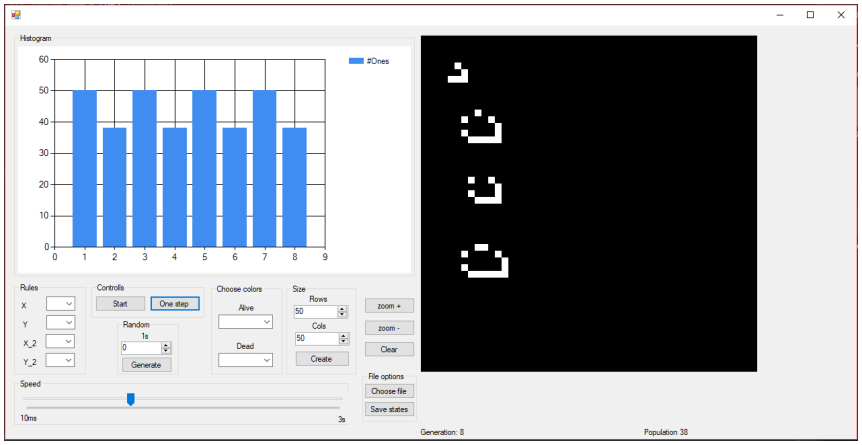
- Una célula muerta con exactamente 3 células vivas "nace" (En la siguiente generación estará viva)
- Una célula viva con 2 o 3 células vecinas vivas sigue viva, en otro caso esta muere.

Pero para esta práctica no solo es posible utilizar estas reglas, si es necesario podemos cambiar estas reglas, donde los valores pueden ir desde 1 hasta el 8.
Para esta práctica se decidió por utilizar el lenguaje de programación C# y el entorno de desarrollo integrado Visual Studio 2015.

2.5.2 Pruebas

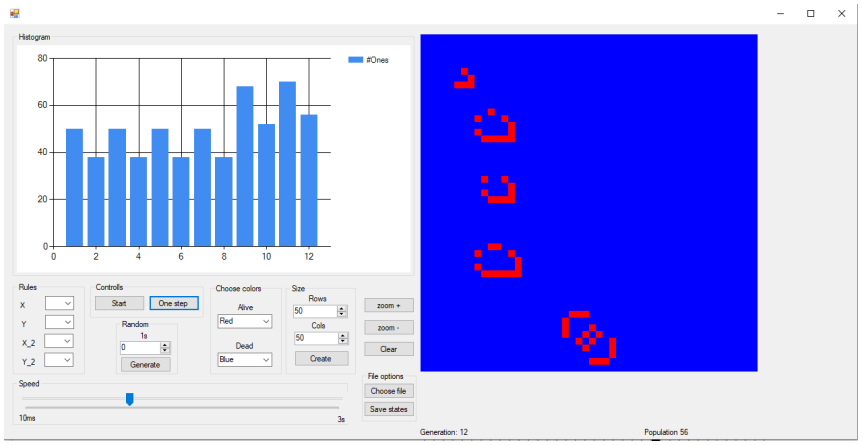
Este programa tiene varios aspectos que han sido cubiertos. Para iniciar las pruebas se han añadido algunas células vivas haciendo click en el elemento que inicialmente es color negro. Esto se realiza haciendo uso de algunos eventos, tal y como se muestra en la siguiente figura.

Figure 2.1: Prueba haciendo uso de eventos



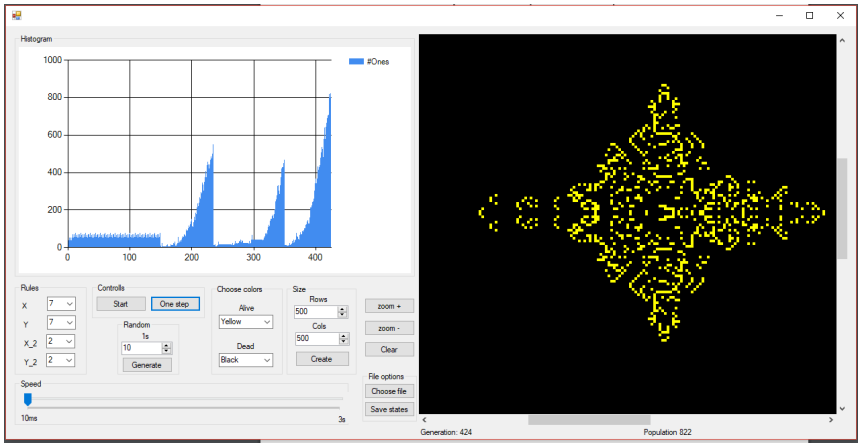
A continuación se muestra el uso del cambio de colores. Podemos seleccionar los colores del elemento que nos representa de forma gráfica la matriz, podemos hacer uso de hasta 7 colores.

Figure 2.2: Uso de colores



Las dos pruebas anteriores han sido realizadas colocando las reglas de "Game of life", estas reglas son colocadas por defecto, si no hay establecido algún parámetro. Para la siguiente prueba se han realizado cambios por las reglas de "Diffusion"

Figure 2.3: Prueba regla de difusión



Otro de los parámetros cubiertos es el guardar alguna generación que queramos, por ejemplo la última generada con las reglas de difusión.

Figure 2.4: Guardando datos

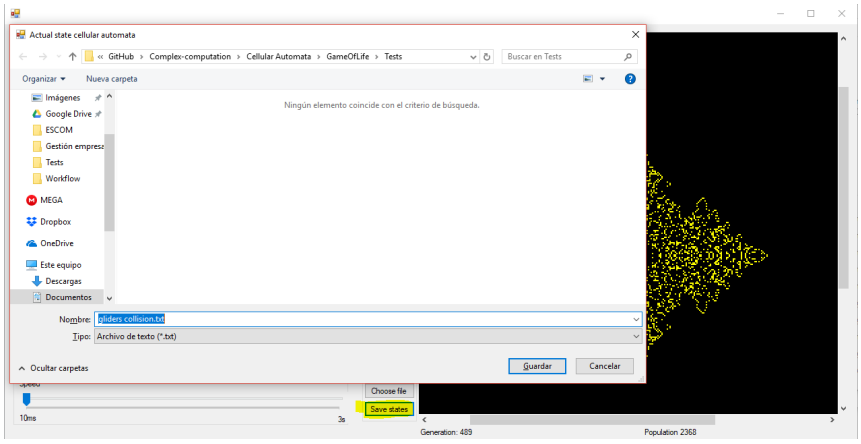
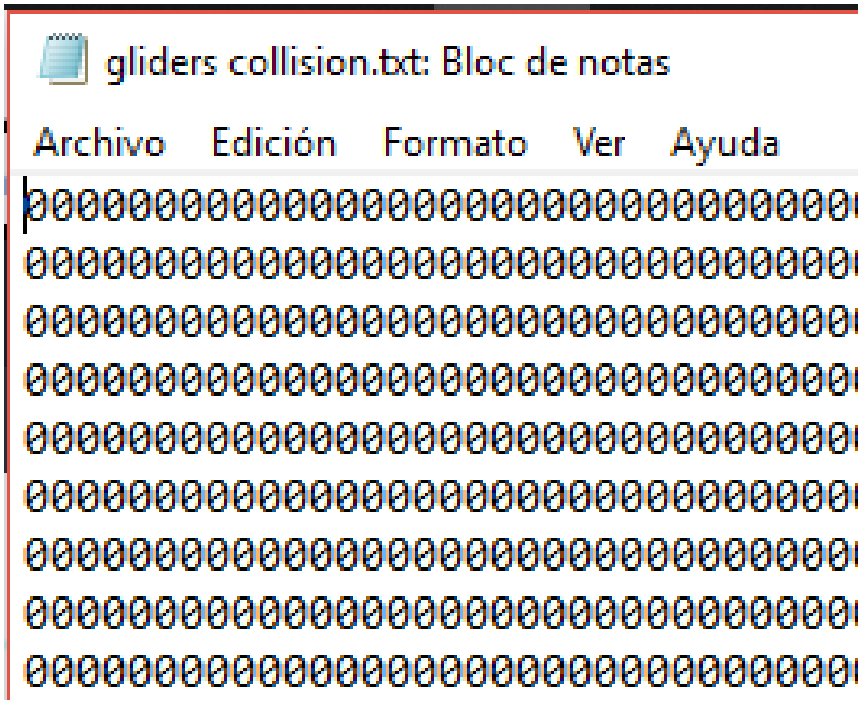
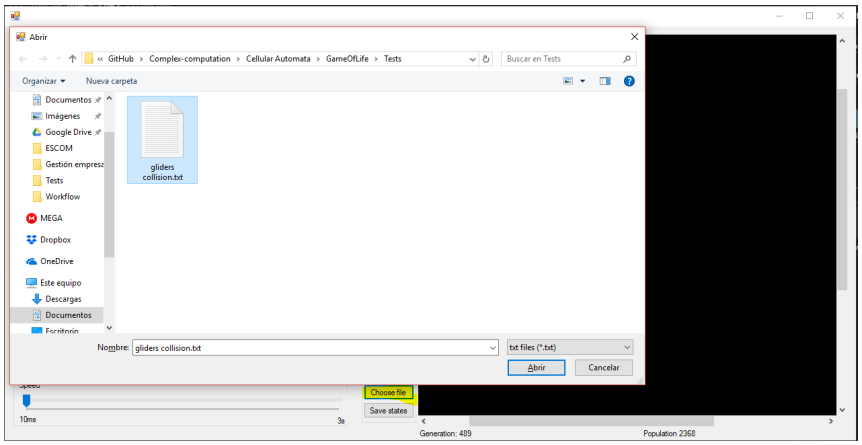


Figure 2.5: Contenido del archivo



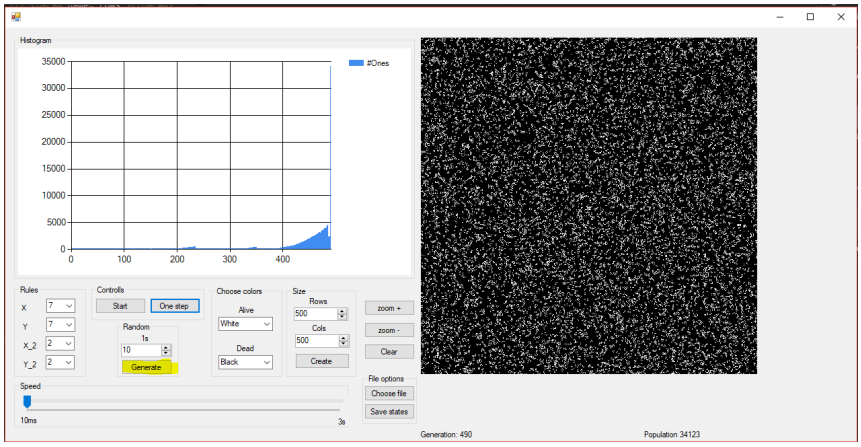
También como era de esperarse algún archivo de texto también puede ser cargado al programa.

Figure 2.6: Recuperando datos



Finalmente es posible generar un random para llenar matriz. El usuario tiene la posibilidad de elegir la probabilidad con la que desea que aparezcan los unos.

Figure 2.7: Random



2.5.3 Códigos

A continuación se anexa el código generado para la creación del simulador. Primero tenemos la siguiente clase la cual contiene código autogenerado por el IDE. Esta código es únicamente para el manejo de la interfaz.

```
1  namespace GameOfLife
2  {
3      partial class Form1
4      {
5          /// <summary>
6          /// Variable del ñdiseador necesaria.
7          /// </summary>
8          private System.ComponentModel.IContainer components = null;
9
10         /// <summary>
11         /// Limpiar los recursos que se éestn usando.
12         /// </summary>
13         /// <param name="disposing">true si los recursos administrados
               se deben desechar; false en caso contrario.</param>
14         protected override void Dispose(bool disposing)
15         {
16             if (disposing && (components != null))
17             {
18                 components.Dispose();
19             }
20             base.Dispose(disposing);
21         }
22
23         #region óCdigo generado por el ñDiseador de Windows Forms
24
25         /// <summary>
26         /// éMtodo necesario para admitir el ñDiseador. No se puede
               modificar
27         /// el contenido de este émtodo con el editor de ócdigo.
```

```
28          /// </summary>
29          private void InitializeComponent()
30          {
31              this.components = new System.ComponentModel.Container();
32              System.Windows.Forms.DataVisualization.Charting.ChartArea
                    chartArea1 = new System.Windows.Forms.DataVisualization.
                    Charting.ChartArea();
33              System.Windows.Forms.DataVisualization.Charting.Legend
                    legend1 = new System.Windows.Forms.DataVisualization.
                    Charting.Legend();
34              System.Windows.Forms.DataVisualization.Charting.Series
                    series1 = new System.Windows.Forms.DataVisualization.
                    Charting.Series();
35              this.PBAutomataSimulator = new System.Windows.Forms.
                    PictureBox();
36              this.CHHistogram = new System.Windows.Forms.
                    DataVisualization.Charting.Chart();
37              this.groupBox1 = new System.Windows.Forms.GroupBox();
38              this.BTNStart = new System.Windows.Forms.Button();
39              this.groupBox2 = new System.Windows.Forms.GroupBox();
40              this.label5 = new System.Windows.Forms.Label();
41              this.label4 = new System.Windows.Forms.Label();
42              this.label3 = new System.Windows.Forms.Label();
43              this.label2 = new System.Windows.Forms.Label();
44              this.ComboBY2i = new System.Windows.Forms.ComboBox();
45              this.ComboBX2i = new System.Windows.Forms.ComboBox();
46              this.ComboBYi = new System.Windows.Forms.ComboBox();
47              this.ComboBXi = new System.Windows.Forms.ComboBox();
48              this.TXTGeneration = new System.Windows.Forms.Label();
49              this.TXTPopulation = new System.Windows.Forms.Label();
50              this.BTNStep = new System.Windows.Forms.Button();
51              this.TBSpeed = new System.Windows.Forms.TrackBar();
52              this.TimerSimulation = new System.Windows.Forms.Timer(this.
                    components);
53              this.flowLayoutPanel1 = new System.Windows.Forms.
                    FlowLayoutPanel();
54              this.groupBox3 = new System.Windows.Forms.GroupBox();
55              this.label6 = new System.Windows.Forms.Label();
56              this.label1 = new System.Windows.Forms.Label();
57              this.BTNZoomP = new System.Windows.Forms.Button();
58              this.BTNZoomM = new System.Windows.Forms.Button();
59              this.groupBox4 = new System.Windows.Forms.GroupBox();
60              this.groupBox5 = new System.Windows.Forms.GroupBox();
61              this.button1 = new System.Windows.Forms.Button();
62              this.numericOnes = new System.Windows.Forms.NumericUpDown();
63              this.label7 = new System.Windows.Forms.Label();
64              this.groupBox6 = new System.Windows.Forms.GroupBox();
65              this.CBDead = new System.Windows.Forms.ComboBox();
66              this.CBAlive = new System.Windows.Forms.ComboBox();
67              this.label10 = new System.Windows.Forms.Label();
68              this.label9 = new System.Windows.Forms.Label();
69              this.BTNSelectFile = new System.Windows.Forms.Button();
70              this.BTNClear = new System.Windows.Forms.Button();
```

```
71              this.groupBox7 = new System.Windows.Forms.GroupBox();
72              this.label8 = new System.Windows.Forms.Label();
73              this.label11 = new System.Windows.Forms.Label();
74              this.numericRows = new System.Windows.Forms.NumericUpDown();
75              this.numericCols = new System.Windows.Forms.NumericUpDown();
76              this.BTNCreateMatrix = new System.Windows.Forms.Button();
77              this.groupBox8 = new System.Windows.Forms.GroupBox();
78              this.BTNSave = new System.Windows.Forms.Button();
79              ((System.ComponentModel.ISupportInitialize)(this.
                    PBAutomataSimulator)).BeginInit();
80              ((System.ComponentModel.ISupportInitialize)(this.CHHistogram
                    )).BeginInit();
81              this.groupBox1.SuspendLayout();
82              this.groupBox2.SuspendLayout();
83              ((System.ComponentModel.ISupportInitialize)(this.TBSpeed)).
                    BeginInit();
84              this.flowLayoutPanel1.SuspendLayout();
85              this.groupBox3.SuspendLayout();
86              this.groupBox4.SuspendLayout();
87              this.groupBox5.SuspendLayout();
88              ((System.ComponentModel.ISupportInitialize)(this.numericOnes
                    )).BeginInit();
89              this.groupBox6.SuspendLayout();
90              this.groupBox7.SuspendLayout();
91              ((System.ComponentModel.ISupportInitialize)(this.numericRows
                    )).BeginInit();
92              ((System.ComponentModel.ISupportInitialize)(this.numericCols
                    )).BeginInit();
93              this.groupBox8.SuspendLayout();
94              this.SuspendLayout();
95              //
96              // PBAutomataSimulator
97              //
98              this.PBAutomataSimulator.BackColor = System.Drawing.
                    SystemColors.ActiveCaptionText;
99              this.PBAutomataSimulator.Location = new System.Drawing.Point
                    (3, 3);
100             this.PBAutomataSimulator.Name = "PBAutomataSimulator";
101             this.PBAutomataSimulator.Size = new System.Drawing.Size(545,
                     505);
102             this.PBAutomataSimulator.SizeMode = System.Windows.Forms.
                    PictureBoxSizeMode.AutoSize;
103             this.PBAutomataSimulator.TabIndex = 1;
104             this.PBAutomataSimulator.TabStop = false;
105             this.PBAutomataSimulator.Paint += new System.Windows.Forms.
                    PaintEventHandler(this.PBAutomataSimulator_Paint);
106             this.PBAutomataSimulator.MouseDown += new System.Windows.
                    Forms.MouseEventHandler(this.
                    PBAutomataSimulator_MouseDown);
107             //
108             // CHHistogram
109             //
110             chartArea1.Name = "ChartArea1";
```

```
111             this.CHHistogram.ChartAreas.Add(chartArea1);
112             legend1.Name = "Legend1";
113             this.CHHistogram.Legends.Add(legend1);
114             this.CHHistogram.Location = new System.Drawing.Point(6, 19);
115             this.CHHistogram.Name = "CHHistogram";
116             series1.ChartArea = "ChartArea1";
117             series1.Legend = "Legend1";
118             series1.Name = "#Ones";
119             this.CHHistogram.Series.Add(series1);
120             this.CHHistogram.Size = new System.Drawing.Size(584, 337);
121             this.CHHistogram.TabIndex = 2;
122             this.CHHistogram.Text = "chart1";
123             //
124             // groupBox1
125             //
126             this.groupBox1.Controls.Add(this.CHHistogram);
127             this.groupBox1.Location = new System.Drawing.Point(13, 12);
128             this.groupBox1.Name = "groupBox1";
129             this.groupBox1.Size = new System.Drawing.Size(596, 362);
130             this.groupBox1.TabIndex = 3;
131             this.groupBox1.TabStop = false;
132             this.groupBox1.Text = "Histogram";
133             //
134             // BTNStart
135             //
136             this.BTNStart.Location = new System.Drawing.Point(6, 19);
137             this.BTNStart.Name = "BTNStart";
138             this.BTNStart.Size = new System.Drawing.Size(75, 23);
139             this.BTNStart.TabIndex = 4;
140             this.BTNStart.Text = "Start";
141             this.BTNStart.UseVisualStyleBackColor = true;
142             this.BTNStart.Click += new System.EventHandler(this.
                    BTNStart_Click);
143             //
144             // groupBox2
145             //
146             this.groupBox2.Controls.Add(this.label5);
147             this.groupBox2.Controls.Add(this.label4);
148             this.groupBox2.Controls.Add(this.label3);
149             this.groupBox2.Controls.Add(this.label2);
150             this.groupBox2.Controls.Add(this.ComboBY2i);
151             this.groupBox2.Controls.Add(this.ComboBX2i);
152             this.groupBox2.Controls.Add(this.ComboBYi);
153             this.groupBox2.Controls.Add(this.ComboBXi);
154             this.groupBox2.Location = new System.Drawing.Point(13, 383);
155             this.groupBox2.Name = "groupBox2";
156             this.groupBox2.Size = new System.Drawing.Size(107, 137);
157             this.groupBox2.TabIndex = 6;
158             this.groupBox2.TabStop = false;
159             this.groupBox2.Text = "Rules";
160             //
161             // label5
162             //
```

```
163             this.label5.AutoSize = true;
164             this.label5.Location = new System.Drawing.Point(11, 111);
165             this.label5.Name = "label5";
166             this.label5.Size = new System.Drawing.Size(26, 13);
167             this.label5.TabIndex = 7;
168             this.label5.Text = "Y_2";
169             //
170             // label4
171             //
172             this.label4.AutoSize = true;
173             this.label4.Location = new System.Drawing.Point(11, 83);
174             this.label4.Name = "label4";
175             this.label4.Size = new System.Drawing.Size(26, 13);
176             this.label4.TabIndex = 6;
177             this.label4.Text = "X_2";
178             //
179             // label3
180             //
181             this.label3.AutoSize = true;
182             this.label3.Location = new System.Drawing.Point(10, 55);
183             this.label3.Name = "label3";
184             this.label3.Size = new System.Drawing.Size(14, 13);
185             this.label3.TabIndex = 5;
186             this.label3.Text = "Y";
187             //
188             // label2
189             //
190             this.label2.AutoSize = true;
191             this.label2.Location = new System.Drawing.Point(8, 27);
192             this.label2.Name = "label2";
193             this.label2.Size = new System.Drawing.Size(14, 13);
194             this.label2.TabIndex = 4;
195             this.label2.Text = "X";
196             //
197             // ComboBY2i
198             //
199             this.ComboBY2i.FormattingEnabled = true;
200             this.ComboBY2i.Items.AddRange(new object[] {
201             "1",
202             "2",
203             "3",
204             "4",
205             "5",
206             "6",
207             "7",
208             "8"});
209             this.ComboBY2i.Location = new System.Drawing.Point(47, 104);
210             this.ComboBY2i.Name = "ComboBY2i";
211             this.ComboBY2i.Size = new System.Drawing.Size(43, 21);
212             this.ComboBY2i.TabIndex = 3;
213             //
214             // ComboBX2i
215             //
```

```
216             this.ComboBX2i.FormattingEnabled = true;
217             this.ComboBX2i.Items.AddRange(new object[] {
218             "1",
219             "2",
220             "3",
221             "4",
222             "5",
223             "6",
224             "7",
225             "8"});
226             this.ComboBX2i.Location = new System.Drawing.Point(47, 76);
227             this.ComboBX2i.Name = "ComboBX2i";
228             this.ComboBX2i.Size = new System.Drawing.Size(43, 21);
229             this.ComboBX2i.TabIndex = 2;
230             //
231             // ComboBYi
232             //
233             this.ComboBYi.FormattingEnabled = true;
234             this.ComboBYi.Items.AddRange(new object[] {
235             "1",
236             "2",
237             "3",
238             "4",
239             "5",
240             "6",
241             "7",
242             "8"});
243             this.ComboBYi.Location = new System.Drawing.Point(48, 48);
244             this.ComboBYi.Name = "ComboBYi";
245             this.ComboBYi.Size = new System.Drawing.Size(43, 21);
246             this.ComboBYi.TabIndex = 1;
247             //
248             // ComboBXi
249             //
250             this.ComboBXi.FormattingEnabled = true;
251             this.ComboBXi.Items.AddRange(new object[] {
252             "1",
253             "2",
254             "3",
255             "4",
256             "5",
257             "6",
258             "7",
259             "8"});
260             this.ComboBXi.Location = new System.Drawing.Point(48, 20);
261             this.ComboBXi.Name = "ComboBXi";
262             this.ComboBXi.Size = new System.Drawing.Size(43, 21);
263             this.ComboBXi.TabIndex = 0;
264             //
265             // TXTGeneration
266             //
267             this.TXTGeneration.AutoSize = true;
268             this.TXTGeneration.BackColor = System.Drawing.Color.
```

```
            Transparent;
269         this.TXTGeneration.ForeColor = System.Drawing.Color.Black;
270         this.TXTGeneration.Location = new System.Drawing.Point(615,
                598);
271         this.TXTGeneration.Name = "TXTGeneration";
272         this.TXTGeneration.Size = new System.Drawing.Size(62, 13);
273         this.TXTGeneration.TabIndex = 7;
274         this.TXTGeneration.Text = "Generation ";
275         //
276         // TXTPopulation
277         //
278         this.TXTPopulation.AutoSize = true;
279         this.TXTPopulation.Location = new System.Drawing.Point(947,
                598);
280         this.TXTPopulation.Name = "TXTPopulation";
281         this.TXTPopulation.Size = new System.Drawing.Size(57, 13);
282         this.TXTPopulation.TabIndex = 8;
283         this.TXTPopulation.Text = "Population";
284         //
285         // BTNStep
286         //
287         this.BTNStep.Location = new System.Drawing.Point(87, 19);
288         this.BTNStep.Name = "BTNStep";
289         this.BTNStep.Size = new System.Drawing.Size(75, 23);
290         this.BTNStep.TabIndex = 9;
291         this.BTNStep.Text = "One step";
292         this.BTNStep.UseVisualStyleBackColor = true;
293         this.BTNStep.Click += new System.EventHandler(this.
                BTNStep_Click);
294         //
295         // TBSpeed
296         //
297         this.TBSpeed.Location = new System.Drawing.Point(6, 19);
298         this.TBSpeed.Maximum = 3000;
299         this.TBSpeed.Minimum = 10;
300         this.TBSpeed.Name = "TBSpeed";
301         this.TBSpeed.Size = new System.Drawing.Size(491, 45);
302         this.TBSpeed.TabIndex = 10;
303         this.TBSpeed.Value = 1000;
304         this.TBSpeed.ValueChanged += new System.EventHandler(this.
                trackBar1_ValueChanged);
305         //
306         // TimerSimulation
307         //
308         this.TimerSimulation.Tick += new System.EventHandler(this.
                TimerSimulation_Tick);
309         //
310         // flowLayoutPanel1
311         //
312         this.flowLayoutPanel1.Controls.Add(this.PBAutomataSimulator)
                ;
313         this.flowLayoutPanel1.Location = new System.Drawing.Point
                (615, 12);
```

```
314             this.flowLayoutPanel1.Name = "flowLayoutPanel1";
315             this.flowLayoutPanel1.Size = new System.Drawing.Size(639,
                    583);
316             this.flowLayoutPanel1.TabIndex = 11;
317             //
318             // groupBox3
319             //
320             this.groupBox3.Controls.Add(this.label6);
321             this.groupBox3.Controls.Add(this.label1);
322             this.groupBox3.Controls.Add(this.TBSpeed);
323             this.groupBox3.Location = new System.Drawing.Point(13, 526);
324             this.groupBox3.Name = "groupBox3";
325             this.groupBox3.Size = new System.Drawing.Size(497, 69);
326             this.groupBox3.TabIndex = 12;
327             this.groupBox3.TabStop = false;
328             this.groupBox3.Text = "Speed";
329             //
330             // label6
331             //
332             this.label6.AutoSize = true;
333             this.label6.Location = new System.Drawing.Point(479, 53);
334             this.label6.Name = "label6";
335             this.label6.Size = new System.Drawing.Size(18, 13);
336             this.label6.TabIndex = 12;
337             this.label6.Text = "3s";
338             //
339             // label1
340             //
341             this.label1.AutoSize = true;
342             this.label1.Location = new System.Drawing.Point(6, 51);
343             this.label1.Name = "label1";
344             this.label1.Size = new System.Drawing.Size(32, 13);
345             this.label1.TabIndex = 11;
346             this.label1.Text = "10ms";
347             //
348             // BTNZoomP
349             //
350             this.BTNZoomP.Location = new System.Drawing.Point(534, 405);
351             this.BTNZoomP.Name = "BTNZoomP";
352             this.BTNZoomP.Size = new System.Drawing.Size(75, 23);
353             this.BTNZoomP.TabIndex = 13;
354             this.BTNZoomP.Text = "zoom +";
355             this.BTNZoomP.UseVisualStyleBackColor = true;
356             this.BTNZoomP.Click += new System.EventHandler(this.
                    BTNZoomP_Click);
357             //
358             // BTNZoomM
359             //
360             this.BTNZoomM.Location = new System.Drawing.Point(534, 438);
361             this.BTNZoomM.Name = "BTNZoomM";
362             this.BTNZoomM.Size = new System.Drawing.Size(75, 23);
363             this.BTNZoomM.TabIndex = 14;
364             this.BTNZoomM.Text = "zoom -";
```

```
365             this.BTNZoomM.UseVisualStyleBackColor = true;
366             this.BTNZoomM.Click += new System.EventHandler(this.
                    BTNZoomM_Click);
367             //
368             // groupBox4
369             //
370             this.groupBox4.Controls.Add(this.BTNStart);
371             this.groupBox4.Controls.Add(this.BTNStep);
372             this.groupBox4.Location = new System.Drawing.Point(128, 383)
                    ;
373             this.groupBox4.Name = "groupBox4";
374             this.groupBox4.Size = new System.Drawing.Size(168, 53);
375             this.groupBox4.TabIndex = 15;
376             this.groupBox4.TabStop = false;
377             this.groupBox4.Text = "Controlls";
378             //
379             // groupBox5
380             //
381             this.groupBox5.Controls.Add(this.button1);
382             this.groupBox5.Controls.Add(this.numericOnes);
383             this.groupBox5.Controls.Add(this.label7);
384             this.groupBox5.Location = new System.Drawing.Point(166, 438)
                    ;
385             this.groupBox5.Name = "groupBox5";
386             this.groupBox5.Size = new System.Drawing.Size(92, 82);
387             this.groupBox5.TabIndex = 16;
388             this.groupBox5.TabStop = false;
389             this.groupBox5.Text = "Random";
390             //
391             // button1
392             //
393             this.button1.Location = new System.Drawing.Point(7, 55);
394             this.button1.Name = "button1";
395             this.button1.Size = new System.Drawing.Size(75, 23);
396             this.button1.TabIndex = 4;
397             this.button1.Text = "Generate";
398             this.button1.UseVisualStyleBackColor = true;
399             this.button1.Click += new System.EventHandler(this.
                    button1_Click);
400             //
401             // numericOnes
402             //
403             this.numericOnes.Location = new System.Drawing.Point(6, 32);
404             this.numericOnes.Name = "numericOnes";
405             this.numericOnes.Size = new System.Drawing.Size(76, 20);
406             this.numericOnes.TabIndex = 2;
407             //
408             // label7
409             //
410             this.label7.AutoSize = true;
411             this.label7.Location = new System.Drawing.Point(33, 17);
412             this.label7.Name = "label7";
413             this.label7.Size = new System.Drawing.Size(18, 13);
```

```
414             this.label7.TabIndex = 0;
415             this.label7.Text = "1s";
416             //
417             // groupBox6
418             //
419             this.groupBox6.Controls.Add(this.CBDead);
420             this.groupBox6.Controls.Add(this.CBAlive);
421             this.groupBox6.Controls.Add(this.label10);
422             this.groupBox6.Controls.Add(this.label9);
423             this.groupBox6.Location = new System.Drawing.Point(306, 385)
                    ;
424             this.groupBox6.Name = "groupBox6";
425             this.groupBox6.Size = new System.Drawing.Size(106, 135);
426             this.groupBox6.TabIndex = 17;
427             this.groupBox6.TabStop = false;
428             this.groupBox6.Text = "Choose colors";
429             //
430             // CBDead
431             //
432             this.CBDead.FormattingEnabled = true;
433             this.CBDead.Items.AddRange(new object[] {
434             "White",
435             "Black",
436             "Red",
437             "Blue",
438             "Green",
439             "Yellow",
440             "Violet"});
441             this.CBDead.Location = new System.Drawing.Point(11, 102);
442             this.CBDead.Name = "CBDead";
443             this.CBDead.Size = new System.Drawing.Size(81, 21);
444             this.CBDead.TabIndex = 3;
445             this.CBDead.SelectedIndexChanged += new System.EventHandler(
                    this.CBDead_SelectedIndexChanged);
446             //
447             // CBAlive
448             //
449             this.CBAlive.AutoCompleteCustomSource.AddRange(new string[]
                    {
450             "White",
451             "Black",
452             "Red",
453             "Blue",
454             "Green",
455             "Yellow",
456             "Violet"});
457             this.CBAlive.FormattingEnabled = true;
458             this.CBAlive.Items.AddRange(new object[] {
459             "White",
460             "Black",
461             "Red",
462             "Blue",
463             "Green",
```

```
464             "Yellow",
465             "Violet"});
466             this.CBAlive.Location = new System.Drawing.Point(11, 45);
467             this.CBAlive.Name = "CBAlive";
468             this.CBAlive.Size = new System.Drawing.Size(81, 21);
469             this.CBAlive.TabIndex = 2;
470             this.CBAlive.SelectedIndexChanged += new System.EventHandler
                    (this.CBAlive_SelectedIndexChanged);
471             //
472             // label10
473             //
474             this.label10.AutoSize = true;
475             this.label10.Location = new System.Drawing.Point(35, 85);
476             this.label10.Name = "label10";
477             this.label10.Size = new System.Drawing.Size(33, 13);
478             this.label10.TabIndex = 1;
479             this.label10.Text = "Dead";
480             //
481             // label9
482             //
483             this.label9.AutoSize = true;
484             this.label9.Location = new System.Drawing.Point(38, 28);
485             this.label9.Name = "label9";
486             this.label9.Size = new System.Drawing.Size(30, 13);
487             this.label9.TabIndex = 0;
488             this.label9.Text = "Alive";
489             //
490             // BTNSelectFile
491             //
492             this.BTNSelectFile.Location = new System.Drawing.Point(3,
                    17);
493             this.BTNSelectFile.Name = "BTNSelectFile";
494             this.BTNSelectFile.Size = new System.Drawing.Size(75, 23);
495             this.BTNSelectFile.TabIndex = 18;
496             this.BTNSelectFile.Text = "Choose file";
497             this.BTNSelectFile.UseVisualStyleBackColor = true;
498             this.BTNSelectFile.Click += new System.EventHandler(this.
                    button2_Click);
499             //
500             // BTNClear
501             //
502             this.BTNClear.Location = new System.Drawing.Point(534, 470);
503             this.BTNClear.Name = "BTNClear";
504             this.BTNClear.Size = new System.Drawing.Size(75, 23);
505             this.BTNClear.TabIndex = 19;
506             this.BTNClear.Text = "Clear";
507             this.BTNClear.UseVisualStyleBackColor = true;
508             this.BTNClear.Click += new System.EventHandler(this.
                    BTNClear_Click);
509             //
510             // groupBox7
511             //
512             this.groupBox7.Controls.Add(this.BTNCreateMatrix);
```

```
513             this.groupBox7.Controls.Add(this.numericCols);
514             this.groupBox7.Controls.Add(this.numericRows);
515             this.groupBox7.Controls.Add(this.label11);
516             this.groupBox7.Controls.Add(this.label8);
517             this.groupBox7.Location = new System.Drawing.Point(418, 385)
                    ;
518             this.groupBox7.Name = "groupBox7";
519             this.groupBox7.Size = new System.Drawing.Size(104, 135);
520             this.groupBox7.TabIndex = 20;
521             this.groupBox7.TabStop = false;
522             this.groupBox7.Text = "Size";
523             //
524             // label8
525             //
526             this.label8.AutoSize = true;
527             this.label8.Location = new System.Drawing.Point(31, 15);
528             this.label8.Name = "label8";
529             this.label8.Size = new System.Drawing.Size(34, 13);
530             this.label8.TabIndex = 0;
531             this.label8.Text = "Rows";
532             //
533             // label11
534             //
535             this.label11.AutoSize = true;
536             this.label11.Location = new System.Drawing.Point(36, 55);
537             this.label11.Name = "label11";
538             this.label11.Size = new System.Drawing.Size(27, 13);
539             this.label11.TabIndex = 1;
540             this.label11.Text = "Cols";
541             //
542             // numericRows
543             //
544             this.numericRows.Location = new System.Drawing.Point(11, 31)
                    ;
545             this.numericRows.Maximum = new decimal(new int[] {
546             1000,
547             0,
548             0,
549             0});
550             this.numericRows.Minimum = new decimal(new int[] {
551             10,
552             0,
553             0,
554             0});
555             this.numericRows.Name = "numericRows";
556             this.numericRows.Size = new System.Drawing.Size(81, 20);
557             this.numericRows.TabIndex = 2;
558             this.numericRows.Value = new decimal(new int[] {
559             10,
560             0,
561             0,
562             0});
563             //
```

```
564             // numericCols
565             //
566             this.numericCols.Location = new System.Drawing.Point(13, 71)
                    ;
567             this.numericCols.Maximum = new decimal(new int[] {
568             1000,
569             0,
570             0,
571             0});
572             this.numericCols.Minimum = new decimal(new int[] {
573             10,
574             0,
575             0,
576             0});
577             this.numericCols.Name = "numericCols";
578             this.numericCols.Size = new System.Drawing.Size(81, 20);
579             this.numericCols.TabIndex = 3;
580             this.numericCols.Value = new decimal(new int[] {
581             10,
582             0,
583             0,
584             0});
585             //
586             // BTNCreateMatrix
587             //
588             this.BTNCreateMatrix.Location = new System.Drawing.Point(13,
                     99);
589             this.BTNCreateMatrix.Name = "BTNCreateMatrix";
590             this.BTNCreateMatrix.Size = new System.Drawing.Size(81, 23);
591             this.BTNCreateMatrix.TabIndex = 21;
592             this.BTNCreateMatrix.Text = "Create";
593             this.BTNCreateMatrix.UseVisualStyleBackColor = true;
594             this.BTNCreateMatrix.Click += new System.EventHandler(this.
                    BTNCreateMatrix_Click);
595             //
596             // groupBox8
597             //
598             this.groupBox8.Controls.Add(this.BTNSave);
599             this.groupBox8.Controls.Add(this.BTNSelectFile);
600             this.groupBox8.Location = new System.Drawing.Point(531, 515)
                    ;
601             this.groupBox8.Name = "groupBox8";
602             this.groupBox8.Size = new System.Drawing.Size(81, 75);
603             this.groupBox8.TabIndex = 21;
604             this.groupBox8.TabStop = false;
605             this.groupBox8.Text = "File options";
606             //
607             // BTNSave
608             //
609             this.BTNSave.Location = new System.Drawing.Point(3, 44);
610             this.BTNSave.Name = "BTNSave";
611             this.BTNSave.Size = new System.Drawing.Size(75, 23);
612             this.BTNSave.TabIndex = 0;
```

```
613             this.BTNSave.Text = "Save states";
614             this.BTNSave.UseVisualStyleBackColor = true;
615             this.BTNSave.Click += new System.EventHandler(this.
                    BTNSave_Click);
616             //
617             // Form1
618             //
619             this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F)
                    ;
620             this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
                    ;
621             this.ClientSize = new System.Drawing.Size(1266, 615);
622             this.Controls.Add(this.groupBox8);
623             this.Controls.Add(this.groupBox7);
624             this.Controls.Add(this.BTNClear);
625             this.Controls.Add(this.groupBox6);
626             this.Controls.Add(this.groupBox5);
627             this.Controls.Add(this.groupBox4);
628             this.Controls.Add(this.BTNZoomM);
629             this.Controls.Add(this.BTNZoomP);
630             this.Controls.Add(this.groupBox3);
631             this.Controls.Add(this.TXTPopulation);
632             this.Controls.Add(this.TXTGeneration);
633             this.Controls.Add(this.groupBox2);
634             this.Controls.Add(this.groupBox1);
635             this.Controls.Add(this.flowLayoutPanel1);
636             this.Name = "Form1";
637             this.Text = " ";
638             ((System.ComponentModel.ISupportInitialize)(this.
                    PBAutomataSimulator)).EndInit();
639             ((System.ComponentModel.ISupportInitialize)(this.CHHistogram
                    )).EndInit();
640             this.groupBox1.ResumeLayout(false);
641             this.groupBox2.ResumeLayout(false);
642             this.groupBox2.PerformLayout();
643             ((System.ComponentModel.ISupportInitialize)(this.TBSpeed)).
                    EndInit();
644             this.flowLayoutPanel1.ResumeLayout(false);
645             this.flowLayoutPanel1.PerformLayout();
646             this.groupBox3.ResumeLayout(false);
647             this.groupBox3.PerformLayout();
648             this.groupBox4.ResumeLayout(false);
649             this.groupBox5.ResumeLayout(false);
650             this.groupBox5.PerformLayout();
651             ((System.ComponentModel.ISupportInitialize)(this.numericOnes
                    )).EndInit();
652             this.groupBox6.ResumeLayout(false);
653             this.groupBox6.PerformLayout();
654             this.groupBox7.ResumeLayout(false);
655             this.groupBox7.PerformLayout();
656             ((System.ComponentModel.ISupportInitialize)(this.numericRows
                    )).EndInit();
657             ((System.ComponentModel.ISupportInitialize)(this.numericCols
```

```
            )).EndInit();
658             this.groupBox8.ResumeLayout(false);
659             this.ResumeLayout(false);
660             this.PerformLayout();
661
662         }
663
664         #endregion
665
666         private System.Windows.Forms.PictureBox PBAutomataSimulator;
667         private System.Windows.Forms.DataVisualization.Charting.Chart
                CHHistogram;
668         private System.Windows.Forms.GroupBox groupBox1;
669         private System.Windows.Forms.Button BTNStart;
670         private System.Windows.Forms.GroupBox groupBox2;
671         private System.Windows.Forms.Label TXTGeneration;
672         private System.Windows.Forms.Label label5;
673         private System.Windows.Forms.Label label4;
674         private System.Windows.Forms.Label label3;
675         private System.Windows.Forms.Label label2;
676         private System.Windows.Forms.ComboBox ComboBY2i;
677         private System.Windows.Forms.ComboBox ComboBX2i;
678         private System.Windows.Forms.ComboBox ComboBYi;
679         private System.Windows.Forms.ComboBox ComboBXi;
680         private System.Windows.Forms.Label TXTPopulation;
681         private System.Windows.Forms.Button BTNStep;
682         private System.Windows.Forms.TrackBar TBSpeed;
683         private System.Windows.Forms.Timer TimerSimulation;
684         private System.Windows.Forms.FlowLayoutPanel flowLayoutPanel1;
685         private System.Windows.Forms.GroupBox groupBox3;
686         private System.Windows.Forms.Label label6;
687         private System.Windows.Forms.Label label1;
688         private System.Windows.Forms.Button BTNZoomP;
689         private System.Windows.Forms.Button BTNZoomM;
690         private System.Windows.Forms.GroupBox groupBox4;
691         private System.Windows.Forms.GroupBox groupBox5;
692         private System.Windows.Forms.Button button1;
693         private System.Windows.Forms.NumericUpDown numericOnes;
694         private System.Windows.Forms.Label label7;
695         private System.Windows.Forms.GroupBox groupBox6;
696         private System.Windows.Forms.ComboBox CBDead;
697         private System.Windows.Forms.ComboBox CBAlive;
698         private System.Windows.Forms.Label label10;
699         private System.Windows.Forms.Label label9;
700         private System.Windows.Forms.Button BTNSelectFile;
701         private System.Windows.Forms.Button BTNClear;
702         private System.Windows.Forms.GroupBox groupBox7;
703         private System.Windows.Forms.NumericUpDown numericCols;
704         private System.Windows.Forms.NumericUpDown numericRows;
705         private System.Windows.Forms.Label label11;
706         private System.Windows.Forms.Label label8;
707         private System.Windows.Forms.Button BTNCreateMatrix;
708         private System.Windows.Forms.GroupBox groupBox8;
```

```
709         private System.Windows.Forms.Button BTNSave;
710     }
711 }
```

Posteriormente tenemos la clase que contiene el código que hace que el autómata se comporte como fue indicado.

```
1  using System;
2  using System.Drawing;
3  using System.Windows.Forms;
4  using System.IO;
5  using System.Collections;
6
7  namespace GameOfLife
8  {
9
10     public partial class Form1 : Form
11     {
12
13         /*************************************
14          *           GLOBAL VARIABLES          *
15          *************************************/
16
17         private bool[,] matrix;
18
19         private int cellArea = 10;
20         private int generation = 1;
21
22         private Brush alive = Brushes.White;
23         private Brush dead = Brushes.Black;
24
25         private String[] colors = { "White", "Black", "Red", "Blue", "
               Green", "Yellow", "Violet"};
26         /// <summary>
27         /// Constructor
28         /// </summary>
29         public Form1()
30         {
31             InitializeComponent();
32             createMatrix(50, 50);
33             scrollBox();
34         }
35
36         /// <summary>
37         /// This function creates a matrix of bools which size it's n x
               m
38         /// also this function adds an extra pair of cols and rows to
39         /// simulate a toroid
40         /// </summary>
41         /// <param name="rows"></param>
42         /// <param name="cols"></param>
43         private void createMatrix(int rows, int cols)
44         {
45             matrix = new bool[cols, rows];
```

```
46              scrollBox();
47          }
48
49          /// <summary>
50          /// This method paints the matrix in the Paint Box
51          /// </summary>
52          /// <param name="sender"></param>
53          /// <param name="e"></param>
54          private void PBAutomataSimulator_Paint(object sender,
                PaintEventArgs e)
55          {
56
57              Graphics graphics = e.Graphics;
58
59              for (int row = 0; row < matrix.GetLength(0); row++)
60              {
61
62                  for (int col = 0; col < matrix.GetLength(1); col++)
63                  {
64
65                      Brush b;
66
67                      if (matrix[row, col])
68                          b = alive;
69                      else
70                          b = dead;
71
72                      graphics.FillRectangle(b, row * cellArea, col *
                            cellArea, cellArea, cellArea);
73
74                  }
75              }
76          }
77
78          /// <summary>
79          /// This funcion manipulates a matrix and evaluate it
80          /// using our rules.
81          /// </summary>
82          /// <param name="p_matrix"></param>
83          /// <returns>A matrix with the new generation data</returns>
84          private bool[,] nextGeneration(bool[,] p_matrix)
85          {
86
87              /**************************************************
88               *                    CONDITIONS                  *
89               * ************************************************/
90
91              /****************** X values ********************/
92              int Xi = Int32.Parse(string.IsNullOrEmpty(ComboBXi.Text) ? "
                    2" : ComboBXi.Text);
93
94              /****************** Y values ********************/
95              int Yi = Int32.Parse(string.IsNullOrEmpty(ComboBYi.Text) ? "
```

```
            3" : ComboBYi.Text);
96
97          /****************** X2 values ******************/
98          int X2i = Int32.Parse((string.IsNullOrEmpty(ComboBX2i.Text)
                ? "3" : ComboBX2i.Text));
99
100         /****************** Y2 values ******************/
101         int Y2i = Int32.Parse((string.IsNullOrEmpty(ComboBY2i.Text)
                ? "3" : ComboBY2i.Text));
102
103         bool[,] new_matrix = new bool[p_matrix.GetLength(0),
                p_matrix.GetLength(1)];
104         //We check each cell from the original matrix and we
                substitute it
105         for (int row = 0; row < p_matrix.GetLength(0); row++)
106         {
107
108             for (int col = 0; col < p_matrix.GetLength(1); col++)
109             {
110                 /**
111                  * Here we need to evaluate using the rules given by
                        input
112                  **/
113                 int neighbors = getAliveNeighbors(p_matrix, row, col
                        );
114                 //If the cell is alive
115                 if (p_matrix[row, col])
116                 {
117                     new_matrix[row, col] = (neighbors >= Xi &&
                            neighbors <= Yi);
118                 }
119                 //If the central cell is dead
120                 else
121                 {
122                     new_matrix[row, col] = (neighbors >= X2i &&
                            neighbors <= Y2i);
123                 }
124
125             }
126
127         }
128         updateTextGeneration();
129         return new_matrix;
130     }
131
132     /// <summary>
133     /// Gets information about the cells around a central cell.
134     /// Obviouslly the cells must to be alive.
135     /// </summary>
136     /// <param name="p_matrix">The actual matrix</param>
137     /// <param name="p_row">row of the central cell</param>
138     /// <param name="p_col">col of the central cell</param>
139     /// <returns>Number of neighboors around the central cell (just
```

```
            living neighbors)</returns>
140         private int getAliveNeighbors(bool[,] p_matrix, int p_row, int
                p_col)
141         {
142
143             int neighbors = 0;
144             try
145             {
146                 int max_rows = p_matrix.GetLength(0);
147                 int max_cols = p_matrix.GetLength(1);
148
149                 for (int row = -1; row <= 1; row++)
150                 {
151
152                     for (int col = -1; col <= 1; col++)
153                     {
154
155                         int c_row = row + p_row;
156                         int c_col = col + p_col;
157
158                         //We are in the center cell
159                         if (c_row == p_row && c_col == p_col)
160                         {
161                             continue;
162                         }
163                         //Corners//
164                         if (c_row == -1 && c_col == -1 & p_matrix[
                                max_rows - 1, max_cols - 1])
165                             neighbors++;
166                         else if (c_row == max_rows && c_col == max_cols
                                && p_matrix[0, 0])
167                             neighbors++;
168                         else if (c_row == -1 && c_col == max_cols &&
                                p_matrix[max_rows - 1, 0])
169                             neighbors++;
170                         else if (c_row == max_rows && c_col == -1 &&
                                p_matrix[0, max_cols -1])
171                             neighbors++;
172                         else
173                         //Left right
174                         if (c_row == -1 && p_matrix[max_rows - 1, c_col
                                ])
175                             neighbors++;
176                         else if (c_row == max_rows && p_matrix[0, c_col
                                ])
177                             neighbors++;
178
179                         //Up down
180                         else if (c_col == -1 && p_matrix[c_row, max_cols
                                 - 1])
181                             neighbors++;
182                         else if (c_col == max_cols && p_matrix[c_row,
                                0])
```

```
183                             neighbors++;
184
185
186                         if (c_row < 0 || c_row >= max_rows)
187                         {
188                             continue;
189                         }
190
191                         if (c_col < 0 || c_col >= max_cols)
192                         {
193                             continue;
194                         }
195
196                         if (p_matrix[c_row, c_col])
197                         {
198                             neighbors++;
199                         }
200
201                     }
202                 }
203             }
204             catch (Exception e)
205             {
206             }
207
208             return neighbors;
209         }
210
211         /// <summary>
212         /// This method calls nextGeneration method and
213         /// updates the GUI and the count of our alive cells
214         /// </summary>
215         private void step()
216         {
217
218             matrix = nextGeneration(matrix);
219             countOnes();
220             PBAutomataSimulator.Invalidate();
221
222         }
223
224         /// <summary>
225         /// Here we just change the text that show us
226         /// the number of generations
227         /// </summary>
228         private void updateTextGeneration()
229         {
230             TXTGeneration.Text = "Generation: " + generation++;
231         }
232
233         /// <summary>
234         /// This method make a rezise of the Paint Box and flow layout
                panel
```

```
235         /// it makes possible make zoom and the movement into the GUI
236         /// </summary>
237         private void scrollBox()
238         {
239             PBAutomataSimulator.Size = new Size((matrix.GetLength(0)) *
                    cellArea, (matrix.GetLength(1)) * cellArea);
240             PBAutomataSimulator.SizeMode = PictureBoxSizeMode.AutoSize;
241             flowLayoutPanel1.AutoScroll = true;
242             flowLayoutPanel1.Controls.Add(PBAutomataSimulator);
243         }
244
245         /// <summary>
246         /// As you can imagine here we just get the number of ones
247         /// in our matrix (alive cells)
248         /// </summary>
249         private void countOnes()
250         {
251             int ones = 0;
252             for (int x = 0; x < matrix.GetLength(0); x++)
253             {
254
255                 for (int y = 0; y < matrix.GetLength(1); y++)
256                 {
257                     if (matrix[x, y]) ones++;
258                 }
259             }
260             CHHistogram.Series["#Ones"].Points.AddY(ones);
261             TXTPopulation.Text = "Population " + ones;
262         }
263
264         /*****************************************************
265          *                        Events                     *
266          *****************************************************/
267
268         private void BTNStep_Click(object sender, EventArgs e)
269         {
270             step();
271         }
272
273         private void PBAutomataSimulator_MouseDown(object sender,
                MouseEventArgs e)
274         {
275             int x = e.X / cellArea;
276             int y = e.Y / cellArea;
277             matrix[x, y] = !matrix[x, y];
278             PBAutomataSimulator.Invalidate();
279         }
280
281         private void BTNStart_Click(object sender, EventArgs e)
282         {
283             if (BTNStart.Text == "Start")
284             {
285                 TimerSimulation.Start();
```

```
286                 BTNStart.Text = "Stop";
287             }
288             else
289             {
290                 TimerSimulation.Stop();
291                 BTNStart.Text = "Start";
292             }
293         }
294 
295         private void trackBar1_ValueChanged(object sender, EventArgs e)
296         {
297             TimerSimulation.Interval = TBSpeed.Value;
298         }
299 
300         private void TimerSimulation_Tick(object sender, EventArgs e)
301         {
302             step();
303         }
304 
305         private void BTNZoomP_Click(object sender, EventArgs e)
306         {
307             if (cellArea < 20)
308             {
309                 cellArea++;
310                 PBAutomataSimulator.Invalidate();
311                 scrollBox();
312             }
313         }
314 
315         private void BTNZoomM_Click(object sender, EventArgs e)
316         {
317             if (cellArea > 1)
318             {
319                 cellArea--;
320                 PBAutomataSimulator.Invalidate();
321                 scrollBox();
322             }
323         }
324 
325         private void CBAlive_SelectedIndexChanged(object sender,
                EventArgs e)
326         {
327             if (CBAlive.Text == colors[0])
328                 alive = Brushes.White;
329             else if (CBAlive.Text == colors[1])
330                 alive = Brushes.Black;
331             else if (CBAlive.Text == colors[2])
332                 alive = Brushes.Red;
333             else if (CBAlive.Text == colors[3])
334                 alive = Brushes.Blue;
335             else if (CBAlive.Text == colors[4])
336                 alive = Brushes.Green;
337             else if (CBAlive.Text == colors[5])
```

```
338                 alive = Brushes.Yellow;
339             else if (CBAlive.Text == colors[6])
340                 alive = Brushes.Violet;
341             PBAutomataSimulator.Invalidate();
342 
343         }
344 
345         private void CBDead_SelectedIndexChanged(object sender,
                EventArgs e)
346         {
347             if (CBDead.Text == colors[0])
348                 dead = Brushes.White;
349             else if (CBDead.Text == colors[1])
350                 dead = Brushes.Black;
351             else if (CBDead.Text == colors[2])
352                 dead = Brushes.Red;
353             else if (CBDead.Text == colors[3])
354                 dead = Brushes.Blue;
355             else if (CBDead.Text == colors[4])
356                 dead = Brushes.Green;
357             else if (CBDead.Text == colors[5])
358                 dead = Brushes.Yellow;
359             else if (CBDead.Text == colors[6])
360                 dead = Brushes.Violet;
361             PBAutomataSimulator.Invalidate();
362         }
363 
364         private void button1_Click(object sender, EventArgs e)
365         {
366             Random r = new Random();
367             for (int x = 0; x < matrix.GetLength(0); x++) {
368 
369                 for (int y = 0; y < matrix.GetLength(1); y++) {
370 
371                     float rand = r.Next(0,100);
372                     if (rand < int.Parse(numericOnes.Text))
373                     {
374                         matrix[x, y] = true;
375                     }
376                     else matrix[x, y] = false;
377                 }
378             }
379             PBAutomataSimulator.Invalidate();
380         }
381 
382         private void BTNClear_Click(object sender, EventArgs e)
383         {
384             for (int x = 0; x < matrix.GetLength(0); x++) {
385 
386                 for (int y = 0; y < matrix.GetLength(1); y++) {
387 
388                     matrix[x, y] = false;
389                 }
```

```
390             }
391             PBAutomataSimulator.Invalidate();
392         }
393
394         private void BTNCreateMatrix_Click(object sender, EventArgs e)
395         {
396             int rows = (numericRows.Value == 0) ? 100 : (int)numericRows
                    .Value;
397             int cols = (numericCols.Value == 0) ? 100 : (int)numericCols
                    .Value;
398             createMatrix(rows, cols);
399         }
400
401         private void button2_Click(object sender, EventArgs e)
402         {
403             int min_lines = 0;
404             int min_chara = 0;
405             String fileName = null;
406
407             try
408             {
409                 using (OpenFileDialog openFileDialog = new
                        OpenFileDialog())
410                 {
411                     openFileDialog.InitialDirectory = "c\\";
412                     openFileDialog.Filter = "txt files (*.txt)|*.txt";
413                     openFileDialog.FilterIndex = 2;
414                     if (openFileDialog.ShowDialog() == DialogResult.OK)
415                     {
416                         fileName = openFileDialog.FileName;
417                     }
418                 }
419
420                 if (fileName != null)
421                 {
422                     Console.WriteLine(fileName);
423                     StreamReader objectReader = new StreamReader(
                            fileName);
424                     //Reading the file, line per line
425                     String line = "";
426                     ArrayList arrayText = new ArrayList();
427                     while (line != null)
428                     {
429                         line = objectReader.ReadLine();
430                         if (line != null)
431                             arrayText.Add(line);
432                     }
433                     objectReader.Close();
434                     //Iterate into the ArrayList and send the
                            information to the GUI
435                     min_lines = arrayText.Count;
436                     min_chara = arrayText[0].ToString().Length;
437
```

```
438                     Console.WriteLine(min_chara);
439                     Console.WriteLine(min_lines);
440                     if (min_chara > matrix.GetLength(1) && min_lines >
                            matrix.GetLength(0))
441                     {
442                         createMatrix(min_chara, min_lines);
443                     }
444                     for (int i = 0; i < min_lines; i++)
445                     {
446                         string strlne = arrayText[i].ToString().Trim();
447                         int j = 0;
448
449                         foreach (char c in strlne)
450                         {
451                             matrix[j++, i] = (c == '1');
452                         }
453                     }
454                     Console.ReadLine();
455                 }
456             }
457             catch (Exception ex) {
458                 Console.WriteLine(ex);
459             }
460             PBAutomataSimulator.Invalidate();
461         }
462
463         private void BTNSave_Click(object sender, EventArgs e)
464         {
465
466             try
467             {
468                 SaveFileDialog saveFileDialog = new SaveFileDialog();
469                 saveFileDialog.Filter = "Archivo de texto|*.txt";
470                 saveFileDialog.Title = "Actual state cellular automata";
471                 saveFileDialog.ShowDialog();
472                 if (saveFileDialog != null)
473                 {
474                     StreamWriter sw = new StreamWriter(saveFileDialog.
                            OpenFile());
475                     for (int i = 0; i < matrix.GetLength(0); i++)
476                     {
477
478                         for (int j = 0; j < matrix.GetLength(1); j++)
479                         {
480                             if (matrix[j, i])
481                                 sw.Write("1");
482                             else if (!matrix[i, j])
483                                 sw.Write("0");
484                         }
485                         sw.WriteLine();
486                     }
487                     sw.Close();
488                 }
```

```
489                 }
490                 catch (Exception ex) {
491                     Console.WriteLine(ex);
492                 }
493 
494             }
495         }
496 }
```

2.6 Conclusiones

Esta práctica es bastante interesante, ya que es increíble lo que se puede lograr con un par de condiciones, que a simple vista parecen insignificantes. Es posible el observar como es que una sola célula puede causar un gran caos en todo el sistema, claro está eso dependerá de las condiciones que sean asignadas. Esto lo podemos ver con las reglas de difusión, es posible crear figuras increíbles con tan solo un par de elementos, como los gliders, los cuales al colisionar generan una figura que prácticamente es infinita y de la cual pude observar que claramente era un fractal. También es importante recalcar que tanto en "Game of life" y "Diffusion" encontramos algunas figuras que se vuelven periódicas, como los osciladores, los ya mencionados gliders, y algunas otras figuras.