ALGORITMOS

Autor: Abel Gallart Bonome

BUSCAMINAS

En JavaFX Swing

Buscaminas (nombre original en inglés: *Minesweeper*) es un videojuego para un jugador creado por Curt Jhonson y Robert Donner en 1989. El objetivo del juego es despejar un campo de minas sin detonar ninguna. El juego consiste en descubrir **todas las ubicaciones de un tablero cuadriculado en las que hay minas**. El juego empieza con todo el tablero cubierto y para empezar a jugar tendremos que pulsar en cualquier casilla, lo cual nos mostrará su contenido, el cual puede ser: una mina, en cuyo caso perderemos la partida; un número, que indicará la cantidad de minas en las 8 casillas adyacentes; o puede aparecer como una casilla en blanco, lo cual indica que no tiene ninguna mina en las casillas que están a su alrededor. Cuando pulsamos en una casilla sin minas en ocasiones se nos abrirán las casillas de alrededor que no tengan minas en sus respectivas casillas adyacentes hasta que el espacio abierto esté rodeado o bien de los límites del mapa o de

casillas con minas en sus alrededores, aunque esto no quiere decir que haya casillas más allá de las que se han mostrado sin minas ya que solo se mostrarán las casillas con minas adyacentes que tengan una casilla sin número en sus casillas cercanas. **Conociendo la cantidad de minas en las casillas adyacentes de varias posiciones al mismo tiempo se puede ir deduciendo el emplazamiento de las minas**, las cuales deberemos ir marcando hasta haber señalado el número de minas escondidas inicialmente, una vez hecho esto si hemos acertado con las casillas marcadas **habremos ganado la partida**.

El juego se puede configurar para jugar en **diferentes dificultades** ya prediseñadas o se puede personalizar manualmente el tamaño del tablero y la cantidad de minas que queremos que se escondan en él, ajustando así la dificultad al gusto de cada jugador. Independientemente de los ajustes, **cada partida será única** ya que las minas se ponen en el tablero de forma aleatoria en cada ocasión.

¿Para qué sirve el Buscaminas?

El buscaminas fue **uno de los primeros juegos que se implementaron en ordenadores personales modernos**, junto con otros juegos como el Solitario o los Corazones, todos estos **no se crearon buscando entretener a los usuarios**, por el contrario, perseguían **que la gente se habituara a usar el ratón**, haciendo que en estos juegos necesitaras **pulsar en cuadrados pequeños con precisión** como en el caso del buscaminas o **pulsar y arrastrar** las cartas como en el caso del solitario. Estos juegos con los años han seguido formando parte de los sistemas operativos de Microsoft de forma nativa, sufriendo actualizaciones y cambios en la interfaz hasta **Windows 8, que es la última versión de Windows que los incluía**.

El buscaminas **clásico usa una cuadricula** de distintas dimensiones, pero con los años han ido apareciendo otras versiones que llevan el concepto más allá introduciendo **tableros triangulares o con otras formas geométricas** además de nuevas mecánicas como la posibilidad de jugar **en tres dimensiones o que puedan aparecer varias minas en la misma casilla**.

Neurólogos británicos realizaron un estudio entre los que juegan y los que no juegan. Los resultados sorprendieron al público. Resultó que las personas que juegan al buscaminas aprenden habilidades motoras mucho más rápido y su **cerebro absorbe** nueva información más rápido. Bonificación: mejora la percepción de los jugadores de los pequeños componentes del todo.

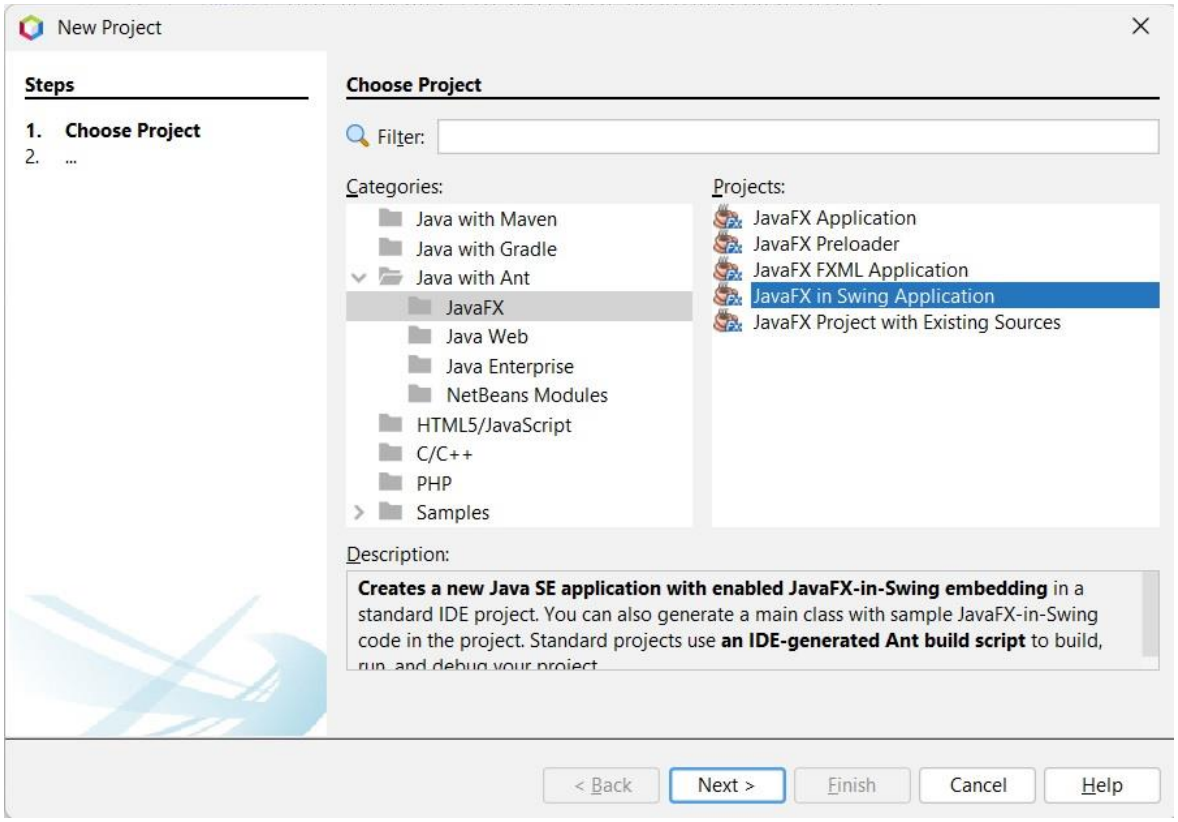
La serie de juegos Buscaminas en línea merece atención. A pesar del **nombre militante**, el juego en línea Buscaminas es **bastante pacífico**. El juego está incluido en el conjunto de juguetes gratuitos para todas las ventanas. ¡Una trama simple no requiere un estudio profundo de las reglas! Pero el juego requiere atención e ingenio por parte de los jugadores. Un movimiento falso resultará en una explosión. Un paso en falso hacia la explosión y la pérdida. Tendremos que empezar de nuevo. El juego Buscaminas es tan atractivo que puede cautivarte fácilmente durante varias horas.

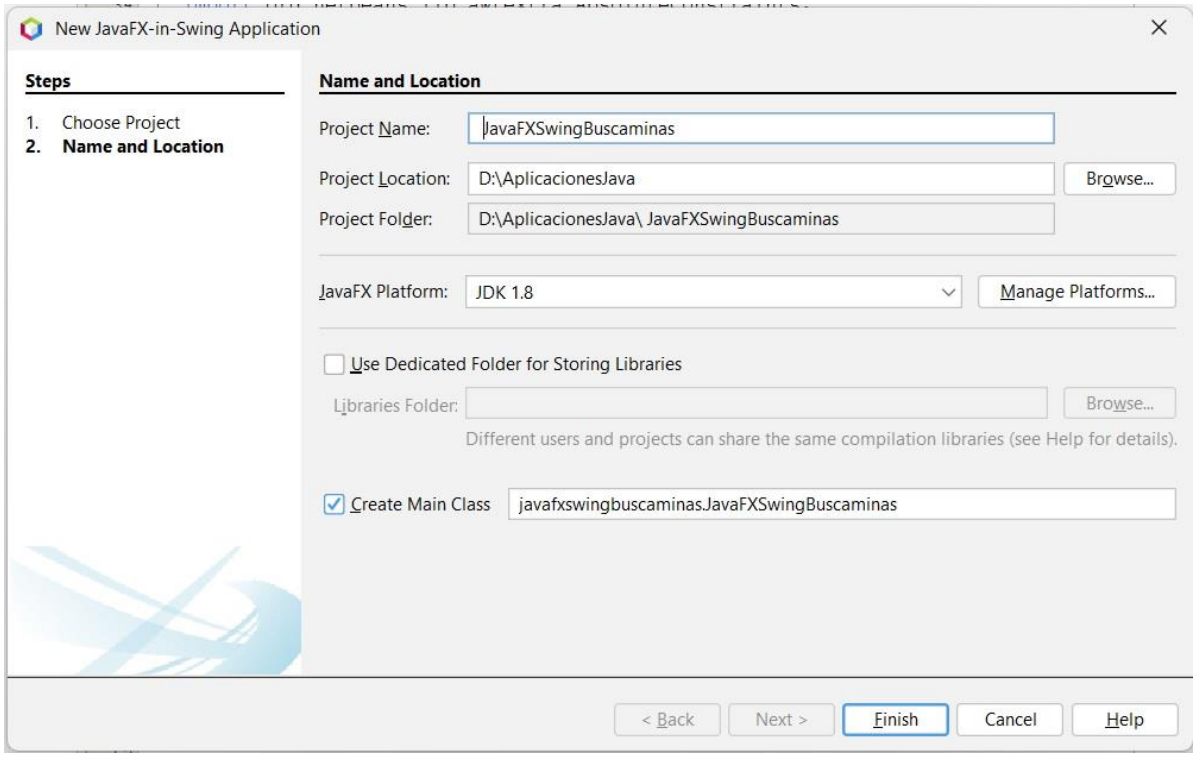
Hay que tener en cuenta que puedes perder desde el primer movimiento. Si aterrizas accidentalmente en una mina detrás de cuadrados ocultos, tendrás que empezar de nuevo. No se moleste, pero la mano misma dirige el mouse de la computadora para reiniciar el proceso.

Buscaminas es adecuado para **niños y adultos**. No cabe duda que este minijuego tiene beneficios, los cuales son los siguientes:

- Mejora el cerebro;
- Desarrolla el pensamiento;
- Desarrolla la capacidad de responder con eficacia;
- Desarrolla habilidades motoras en un niño de 3 a 9 años.
- Ayuda al cerebro a mantenerse en forma.

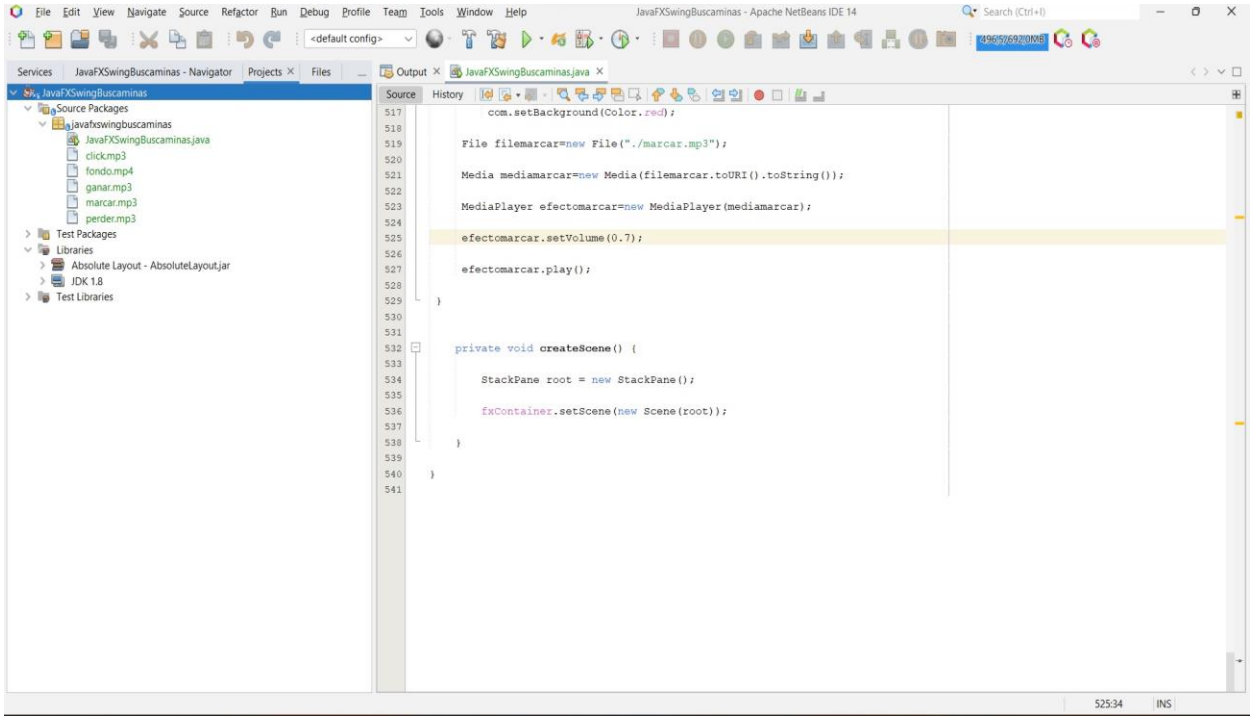
Para comenzar abrimos NetBeans y creamos nuestro proyecto JavaFX Swing.





Y agregamos a nuestro paquete los archivos multimedia que usaremos como efectos de sonido, en concreto los archivos de sonido ganar.mp3, perder.mp3, click.mp3, marcar.mp3, y fondo.mp4 un video que reproduce al fondo de nuestro juego, para hacer más visual el juego.

Código fuente:



Y ahora si pasamos a codificar nuestro algoritmo, tenemos que expresar en lenguaje Java la dinámica del juego.

```
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/javafx/FXSwingMain.java to edit this template
*/
package javafxswingbuscaminas;
```

Importamos las clases que vamos a necesitar

```
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.net.URISyntaxException;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
```

```
import javax.swing.BorderFactory;
import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;

/**
*
* @author Abel Gallart Bonome
*/
public class JavaFXSwingBuscaminas extends JApplet {

    Declaramos las variables de la clase

    int JFXPANEL_WIDTH_INT = 1000;

    int JFXPANEL_HEIGHT_INT = 1000;

    JFXPanel fxContainer;

    static int ancho=30;

    static int largo=30;

    static int bombas=30;
```

```
static int puntos=0;

static int nivel=0;

Random ran;

String [][]matris;

JButton b[][];

JLabel l[][];

static JFrame frame;

static JApplet applet;

private JLabel jLabel1;

private JLabel jLabel2;

private JLabel jLabel3;

private JPanel jPanel1;

private JPanel jPanel2;

private JPanel jPanel3;
```

En nuestro método main creamos la ventana y mostramos una instancia de nuestra clase.

```
  public static void main(String[] args) {

      SwingUtilities.invokeLater(() -> {

          try {


UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
          }

          catch (Exception e){}

          frame = new JFrame("Buscaminas");

          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

          frame.setUndecorated(true);

          frame.setOpacity((float) 1.0);

          applet = new JavaFXSwingBuscaminas();

          applet.init();

          frame.setContentPane(applet.getContentPane());

          frame.pack();

          frame.setLocationRelativeTo(null);
```

```
        frame.setVisible(true);

        applet.start();
    });
}
```

En el metodo init() se inicializan las variables y objetos necesarios tanto para gráfica como para la lógica.

```
@Override
public void init() {

    jPanel2 = new JPanel();

    jLabel1 = new JLabel();

    jLabel2 = new JLabel();

    jLabel3 = new JLabel();

    jPanel3 = new JPanel();

    jPanel2.setSize(new Dimension(720, 720));

    jPanel2.setOpaque(false);

    jPanel2.setLayout(new AbsoluteLayout());

    jLabel1.setFont(new Font("Segoe UI", 3, 36)); // NOI18N

    jLabel1.setForeground(new Color(255, 255, 255));
```

```
        jLabel1.setText("Bombas "+bombas);

        jLabel2.setFont(new Font("Segoe UI", 3, 36)); // NOI18N

        jLabel2.setForeground(new Color(255, 255, 255));

        jLabel2.setText("Nivel "+nivel);

        jLabel3.setFont(new Font("Segoe UI", 3, 36)); // NOI18N

        jLabel3.setForeground(new Color(255, 255, 255));

        jLabel3.setText("Puntos "+puntos);

        jPanel3.setBackground(new Color(255, 153, 0));

        jPanel3.setBorder(BorderFactory.createTitledBorder(null, "Buscaminas",
javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.ABOVE_TOP,
new java.awt.Font("Segoe UI", 0, 12), new java.awt.Color(255, 255, 255))); // NOI18N

        ran=new Random();

        matris=new String[ancho][largo];

        for(int x=0;x<ancho;x++)

            for(int y=0;y<largo;y++)

            {matris[x][y]="0";}
```

```
int c=0;
```

Se crea un objeto Random ram y se inicializa una matriz ancho*largo con el texto ”0” y tomamos un contador c que inicializamos en cero también.

Repetimos las siguientes operaciones que se encuentran dentro del “while” mientras que el contador c sea menor que el número de bombas.

```
while(c<bombas){

   int x=(int)(ancho*ran.nextFloat());

   int y=(int)(largo*ran.nextFloat());
```

Tomamos una posición x , y dentro de la matriz[x][y] al azar y ….

```
   if (!matris[x][y].equals("B"))
   {
      matris[x][y]="B";
      c++;
   }

}
```

De modo que ahora tenemos una matriz de largo*ancho que tiene una cantidad de bombas dispuestas al azar dentro de la matriz.

```
for(int x=0;x<ancho;x++)

   for(int y=0;y<largo;y++)

   {

      c=0;
```

```
try{if (matris[x-1][y].equals("B"))c++;}catch(Exception  e){}

try{if (matris[x-1][y-1].equals("B"))c++;}catch(Exception  e){}

try{ if (matris[x][y-1].equals("B"))c++;}catch(Exception  e){}

try{if (matris[x+1][y-1].equals("B"))c++;}catch(Exception  e){}

try{if (matris[x+1][y].equals("B"))c++;}catch(Exception  e){}

try{if (matris[x+1][y+1].equals("B"))c++;}catch(Exception  e){}

try{if (matris[x][y+1].equals("B"))c++;}catch(Exception  e){}

try{if (matris[x-1][y+1].equals("B"))c++;}catch(Exception  e){}
```

Hora recorremos la matriz y en cada casilla vemos si las 8 casillas vecinas tienen bombas y contamos , entonces tendremos en un contador c las bombas que hay en las casillas vecinas , este proceso se hace para cada una de las casillas, el try&catch lo usamos para evitar los desbordamientos cuando nos coloquemos en las casillas de los bordes de la matriz, en cuyo caso se lanza una excepción de "index out of..." que se captura y el programa continua en la siguiente casilla.

```
if (!matris[x][y].equals("B"))  matris[x][y]=c+"";
```

Y ahora en todas las casillas de la matriz si no hay bomba se coloca la cantidad de bombas que hay en las casillas vecinas.

```
}
```

Así que ya tenemos una matriz de String de largo*ancho con bombas al azar y en el resto de las casillas la cantidad de bombas que hay en las vecinas

```
b=new JButton[ancho][largo];
```

```
l=new JLabel[ancho][largo];
```

El siguiente paso es crear esa matriz pero con elementos gráficos , pues crearemos una matriz de etiquetas JLabel [largo][ancho] y otra de botones JButton[largo][ancho]

```
for(int x=0;x<ancho;x++)

   for(int y=0;y<largo;y++)
      {

         l[x][y]=new JLabel();

         l[x][y].setForeground(Color.WHITE);

         l[x][y].setHorizontalAlignment(JLabel.CENTER);

         if (!matris[x][y].equals("0"))  l[x][y].setText("  "+matris[x][y]);
```

Recorremos otra vez nuestra matriz y en la matriz de etiquetas escribimos el contenido de nuestra matriz[lx][y] de String si no es “0” .

```
         b[x][y]=new JButton();

         b[x][y].setBackground(Color.WHITE);
```

Colocamos en nuestra matriz de botones un botón en cada posición y declaramos un Evento del Mouse especificamos que si sobre el botón se realiza un evento de clic realizamos el siguiente procedimiento.

```
         b[x][y].addMouseListener(new MouseAdapter() {

           @Override
           public void mouseClicked(MouseEvent evt) {

              try {
```

```
                if(evt.getButton()==MouseEvent.BUTTON3) Button3(evt);
```

Si el evento del clic se ha hecho con el botón derecho del mouse realizar Button3(evt) donde se describe el procedimiento a seguir para marcar la casilla como una posible bomba

```
                if(evt.getButton()==MouseEvent.BUTTON1) Button1(evt);
```

Si el evento del clic se ha hecho con el botón izquierdo del mouse realizar Button1(evt) donde se describe el procedimiento a seguir para descubrir si hay una bomba . Si es así hemos perdido sino, descubriremos todas las casillas vacías hasta el borde de las zonas minadas.

```
                }

                catch (URISyntaxException ex) {

Logger.getLogger(JavaFXSwingBuscaminas.class.getName()).log(Level.SEVERE, null, ex);

                }

        }


            });

        jPanel2.add(b[x][y], new AbsoluteConstraints(x*largo,y*ancho,29,29));

        jPanel2.add(l[x][y], new AbsoluteConstraints(x*largo,y*ancho,29,29));
```

Y por último agregamos a panel “JPanel2” en la posición x, y de la cuadricula una etiqueta que donde se escribió el numero de bombas vecinas en cada una y encima un botón que oculta temporalmente el contenido de la etiqueta hasta que ocurre un evento sombre el botón

```
}
```

Hasta aquí tenemos el área de la cuadricula que representa el campo minado

```
        jPanel1 = new javax.swing.JPanel();

        jPanel1.setSize(new
java.awt.Dimension(JFXPANEL_WIDTH_INT,JFXPANEL_WIDTH_INT));

        jPanel1.setLayout(new AbsoluteLayout());

        jPanel1.add(jLabel1, new AbsoluteConstraints(110, 10,270, 60));

        jPanel1.add(jLabel2, new AbsoluteConstraints(330, 10, 190, 60));

        jPanel1.add(jLabel3, new AbsoluteConstraints(540, 10, 240, 60));

        jPanel1.add(jPanel3, new AbsoluteConstraints(100, 10, 670, 60));

        jPanel1.add(jPanel2, new AbsoluteConstraints(50, 80, -1, -1));

        setLayout(new AbsoluteLayout());

        jPanel1.setOpaque(false);

        add(jPanel1,new AbsoluteConstraints(0,0, -1,-1));

        fxContainer = new JFXPanel();

        fxContainer.setPreferredSize(new
Dimension(JFXPANEL_WIDTH_INT,JFXPANEL_HEIGHT_INT));
```

```
add(fxContainer,new AbsoluteConstraints(0,0,-1,-1));


Platform.runLater(new Runnable() {

    @Override
    public void run() {
        try {
            createScene();

            File filefondo=new File("./fondo.mp4");

            Media mediafondo=new Media(filefondo.toURI().toString());

            MediaPlayer fondo=new MediaPlayer(mediafondo);

            MediaView mv=new MediaView(fondo);

            fxContainer.setScene(new Scene(new Group(mv)));

            fondo.setCycleCount(MediaPlayer.INDEFINITE);

            fondo.setVolume(0.7);

            mv.setFitHeight(2000);

            mv.setFitWidth(2000);

            fondo.play();
```

```
            } catch (Exception ex) {}

        }

    });
```
Terminamos de configurar el resto de características graficas de la interface
```
}


public void analiza(JButton bot){

    if (!bot.isVisible()) return ;

    int x=bot.getX()/30;

    int y=bot.getY()/30;

    bot.setVisible(false);

    if (matris[x][y].equals("0"))
    {

        puntos++;

        jLabel3.setText("Puntos "+puntos);

        try {analiza(b[x-1][y]);}catch(Exception e){}

        try {analiza(b[x-1][y-1]);}catch(Exception e){}
```

```
        try {analiza(b[x][y-1]);}catch(Exception  e){}

        try {analiza(b[x+1][y-1]);}catch(Exception  e){}

        try {analiza(b[x+1][y]);}catch(Exception  e){}

        try {analiza(b[x+1][y+1]);}catch(Exception  e){}

        try {analiza(b[x][y+1]);}catch(Exception  e){}

        try { analiza(b[x-1][y+1]);}catch(Exception  e){}

    }

}
```

Este método es llamado cuando damos clic izquierdo sobre el botón , aquí descubrimos la casilla, si hay mina perdemos, sino se descubren en cascada el resto de casillas sin minas hasta los bordes de las zonas minadas

```
private void Button1(MouseEvent  evt) throws URISyntaxException {

    Component com=((Component) evt.getSource());
```

Obtenemos el botón sobre el que se dio el clic

```
    int x=com.getX()/30;

    int y=com.getY()/30;
```

Obtenemos la posición x, y en la matriz a partir de las coordenadas del botón (x , y) en el panel grafico, como las casillas tienen un tamaño de 30*30, al dividir las coordenadas por 30 obtenemos la posición en la matriz de objetos String .

```
    if (matris[x][y].equals("B"))
    {
```

Si en la matriz de strings témenos una bomba en la posición x,y obtenida, entonces hemos perdido.

File fileperder=new File("./perder.mp3");

Media mediaperder=new Media(fileperder.toURI().toString());

MediaPlayer efecto=new MediaPlayer(mediaperder);

efecto.setVolume(0.7);

efecto.play();

Aquí se reproduce un efecto de audio que indica que hemos perdido

javax.swing.JOptionPane.showMessageDialog

(null,"Has Perdido","Fin del juego....",javax.swing.JOptionPane.OK_OPTION);

Lanzamos una ventana dialogo con un mensaje que indica al usuario que ha perdido detenemos el applet y ocultamos la ventana.

applet.stop();

frame.setVisible(false);

Referenciamos nuestra ventana a una nueva Ventana y dejamos sin referencia la anterior que ya se ocupara de ella el colector de basura del sistema.

frame = new JFrame("Buscaminas");

frame.setUndecorated(true);

frame.setOpacity((float) 1.0);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

applet = new JavaFXSwingBuscaminas();

puntos=0;

```
            nivel=0;

            bombas=40;

            applet.init();

            frame.setContentPane(applet.getContentPane());

            frame.pack();

            frame.setLocationRelativeTo(null);

            frame.setVisible(true);

            applet.start();
            Aquí terminamos de lanzar la nueva ventana en el caso de que hayamos perdido
           }

      else{

        En el caso de que no sea una bomba la casilla en la posición (x , y) de la matriz Llamamos al método analiza(b[x][y]) y le pasamos como argumento, el botón de la matriz de botones en la posición (x, y) como resultado se descubrirán en cascada todas las casillas sin bombas hasta el borde de las zonas minadas.
        analiza(b[x][y]);

         puntos++;
         Y sumamos 1 punto y lo escribimos en la etiqueta
         jLabel3.setText("Puntos "+puntos);
```

```
File fileclick=new File("./click.mp3");

Media mediaclick=new Media(fileclick.toURI().toString());

MediaPlayer efectoclick=new MediaPlayer(mediaclick);

efectoclick.setVolume(0.7);

efectoclick.play();
```

Y a continuación reproducimos un audio que indica que hemos dado un clic izquierdo del mouse. Como se puede apreciar estos archivos de audio se encuentran ubicados dentro de la carpeta del proyecto.

```
int c=0;

for(int i=0;i<ancho;i++)

   for(int j=0;j<largo;j++)

      if ( b[i][j].isVisible())   c++;
```

Aquí recorremos la matriz de botones y contamos los que están visibles, si coinciden con el número de bombas esto quiere decir que hemos descubierto todas las casillas vacías y hemos ganado y pasamos al siguiente nivel.

```
 if (c==bombas)

 {
    File fileganar=new File("./ganar.mp3");

    Media mediaganar=new Media(fileganar.toURI().toString());

    MediaPlayer efectoganar=new MediaPlayer(mediaganar);

    efectoganar.setVolume(0.7);
```

efectoganar.play();

Aquí reproduce el audio que indica que hemos ganado.

javax.swing.JOptionPane.showMessageDialog

(null,"Has Ganado","Fin del juego....",javax.swing.JOptionPane.OK_OPTION);

Se lanza una ventana dialogo con un mensaje para el usuario indicándole que ha ganado a continuación se detiene el applet y se oculta la ventana.

applet.stop();

frame.setVisible(false);

frame = new JFrame("Buscaminas");

Referencia la ventana a una nueva dejando la anterior para el colector de basura.

frame.setUndecorated(true);

frame.setOpacity((float) 1.0);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

applet = new JavaFXSwingBuscaminas();

bombas=(bombas+1>ancho*largo/2)?bombas:bombas+1;

nivel++;

puntos+=200;

Lanzamos una nueva ventana con 1 bomba más que la anterior y actualizamos la variable bomba hasta el límite de largo*ancho/2, o sea, la mitad de las casillas de la matriz, subimos un nivel y 200 puntos.

applet.init();

```
            frame.setContentPane(applet.getContentPane());

            frame.pack();

            frame.setLocationRelativeTo(null);

            frame.setVisible(true);

            applet.start();

        }

    }
}
```

En este método se describen las acciones a seguir cuando se desencadena un evento del clic izquierdo. Estás consisten en marcar y desmarcar alternativamente en rojo y blanco.

```
private void Button3(MouseEvent evt) {

    Component com=((Component) evt.getSource());
```

Obtenemos el botón sobre el que se realizó el clic izquierdo.

```
    if (com.getBackground().equals(Color.red))
```

{ Si es de color rojo lo pondremos en blanco y retornamos a la posición del programa desde donde fue llamado el método .

```
        com.setBackground(Color.WHITE);

return;
        }

    if (com.getBackground().equals(Color.WHITE))
```

Si es de color blanco lo pondremos de color rojo y reproducimos un audio que indica que hemos marcado una bomba.

```
        com.setBackground(Color.red);

    File filemarcar=new File("./marcar.mp3");

    Media mediamarcar=new Media(filemarcar.toURI().toString());

    MediaPlayer efectomarcar=new MediaPlayer(mediamarcar);

    efectomarcar.setVolume(0.7);

    efectomarcar.play();

}
```

Este método se llama al crear la ventana para inicializar una serie de características graficas

```
    private void createScene() {

        StackPane root = new StackPane();

        fxContainer.setScene(new Scene(root));

    }

}
```

Y aquí finaliza el código de la clase JavaFXSwingBuscaminas.