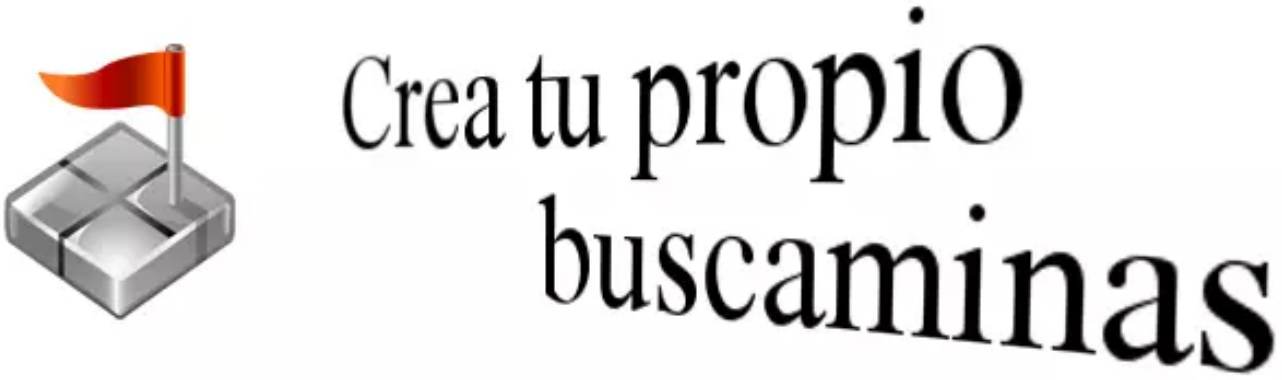
Crea tu propio buscaminas

Compartir



Sin Comentarios

HOY SE HABLA DE Rusia — Microsoft — Android — Amazon Prime — Amazon — HBO — CPU — Fondos de pantalla — Hacker — Intel

PUBLICIDAD

Síguenos

PUBLICIDAD

2011-10-26T16:38:37Z JORGE RUBIRA

Si hay algún juego favorito por profesores de universidad para enviar como prácticas, este es el **buscaminas**. Este juego fue creado por **Robert Donner en 1989**. Se hizo popular gracias a que se incluyó en el sistema operativo Windows 3.11. Desde entonces, ha permanecido en todas sus posteriores versiones.

A pesar de ser archiconocido este juego, es muy fácil desarrollar una versión aceptable o amateur. En este post vamos a explicar las funcionalidades más interesantes a tener en cuenta en el momento del desarrollo. Finalmente, dejaremos un **código fuente** en **Java** de ejemplo.

Reglas del juego

Este apartado lo explicaré resumido ya que en la actualidad es de conocimiento común las reglas de este juego. El juego se basa en un tablero de **NxM** casillas que pueden ser **minas o terreno**. El objetivo del juego es descubrir, haciendo clic sobre las casillas, las celdas que son terreno sin hacer clic a las **minas**.

Para ello, cuando se hace clic sobre terreno (sin mina) se visualiza un número con el total de **minas** que hay alrededor. Si se hace clic sobre terreno sin mina alrededor, se descubren las celdas continuas sucesivamente hasta que se encuentran terrenos con **minas** cerca. Si se hace clic sobre una mina se acaba la partida. Por último, si se tiene la sospecha de que existe una mina, se puede marcar la casilla pulsando el botón derecho.

En un **buscaminas** cualquiera es posible que el número de casillas y el número de minas pueda ser configurable por el usuario. En nuestro caso vamos a crear un tablero de **10×10 casillas** con **10 minas** en el tablero. Pintaremos de rojo las minas y de azul las celdas marcadas con una bandera.

Estructura del juego

La programación del juego se basa en dos algoritmos importantes de lógica y un algoritmo de visualización. Los algoritmos considerables a estudiar son: rellenar el **tablero** aleatoriamente y hacer que cuando se pulse una casilla sin mina cerca se despliegue el terreno hasta encontrar celdas cercanas a minas. En cuanto al algoritmo de visualización dependerá del modo de visualización que utilices: **lienzo, botones, html, etc**. Los dos algoritmos de lógica los veremos en próximos apartados.

Por otra parte, como elementos secundarios (pero esenciales), existirá una variable de estado que indicará si se está jugando **(0)**, hemos perdido **(1)** o hemos ganado **(2)**. Igualmente, cada celda tendrá también un estado que indicará si está tapada **(0)**, está descubierta **(1)** o si se ha puesto una bandera **(2)**.

También existe una variable casillasVistas que nos indicará cuantas casillas hemos descubierto para detectar si hemos conseguido el objetivo. Esta variable no es obligatoria ya que se puede calcular cada vez que destapamos una celda contando los estados de celdas descubiertas. Sin embargo, por motivos de eficiencia y evitar escribir más **código**, es bastante recomendable hacer esta variable.

Rellenar el tablero aleatorio

En muchas ocasiones al trasladar una imagen visual al problema tendemos a imitar lo que vemos. Por ello, lo más intuitivo es crear una matriz para almacenar el **tablero**. Sin embargo, esta no es la única solución ya que se podrían almacenar las minas en un vector dinámico y realizar las consultas únicamente recorriendo este vector de **minas**.

Para nuestro ejemplo … no vamos a ser tan malos y vamos a crear el tablero en forma de matriz para que sea más fácil de entender. En el momento de plantear un **tablero** en forma de matriz se pueden plantear dos posibles formas de trabajo. Poner únicamente las minas y calcular cuantas hay cerca al hacer clic o precalcular los números de las **casillas** con las minas cercanas al principio de la partida.

Para nuestro caso, programaremos la segunda opción y precalcularemos todo. Como consecuencia, como almacenaremos valores de **0 a 8** indicando cuantas minas hay alrededor, no podremos utilizarlos para representar una mina así que utilizaremos el valor **9** para indicar que hay una mina.

El primer problema que nos podemos encontrar es como poner 10 minas aleatoriamente. En el momento de poner una mina, al obtenerlo aleatoriamente hay que tener cuidado de no poner dos **minas** en la misma casilla. Por ello, deberemos repetir el resultado hasta que encuentre un sitio libre de **mina**. En todo caso es importante controlar que el número de minas es menor o igual al total de celdas ya que en caso contrario podría producirse un bucle infinito. Esta verificación puede realizarse en el momento de ejecutar el método o en el momento que el usuario cambia la configuración de minas, filas o columnas no permitiendo valores incoherentes al problema.

```
hacer{
    obtener fila y columna al azar
}mientras que (tablero[fila][columna]==mina);
```

Con esto ya tendríamos el algoritmo que pone las **minas**, pero faltaría obtener los valores 1-8 de las celdas cercanas a las **minas**. Esto se puede hacer de dos maneras: calcularlo al final de poner todas las minas o incrementar el alrededor de la celda cuando se una mina. Nosotros utilizaremos esta segunda.

En el momento de recorrer el alrededor de una mina hay que tener cuidado de no salirse del tablero en el caso de poner la mina en un **lateral o esquina**. La solución intuitiva es utilizar un if en el recorrido comprobando que no se sale del vector. Sin embargo, se puede sacar una solución más corta conociendo las funciones **mínimo y máximo**.

```
para fila2 = max(0,fila-1) hasta min(TAM-1, fila+1) hacer
     para columna2 = max(0,col-1) hasta min(TAM-1, col+1) hacer
           si (no es mina tablero[fila2, columna2]) entonces
                tablero[fila2,columna2]++
           fin de si
     fin de para
fin de para
```

Pulsar una celda sin minas alrededor

Otro reto que se encuentran los alumnos en este juego es descubrir más celdas en el caso de que no haya minas cerca. A simple vista, la solución intuitiva para un alumno novel es utilizar una iteración **while**, por no saber cuantas **celdas** (veces) se van a recorrer. Sin embargo, la siguiente duda que viene es como hacer que se recorra un área, bastante asimétrica, por cierto.

El while es una posible solución combinándola con un vector dinámico, sin embargo aprender a utilizar algoritmos recursivos para este caso es una buena idea. No hay que olvidar que una iteración **while** es una iteración lineal (de cada elemento solo puede ejecutar uno a continuación). Sin embargo, los algoritmos recursivos nos permiten que de cada elemento se ejecuten N elementos similares. En el caso del **buscaminas** de cada celda se pueden destapar **8** celdas más. El algoritmo de manera resumida sería algo así

```
si es celda a destapar entonces
    destapa celda
   si tiene valor 0 entonces
        para todas las celdas de alrededor hacer
                si no es mina entonces
                     hacer clic recursivamente en las celdas cercanas
                fin de si
           fin de para
   fin de si
fin de si
```

Código fuente del buscaminas

Finalmente aquí tenéis un **código fuente** de ejemplo de **buscaminas** que podeis tomar como ejemplo y, por supuesto, probarlo y modificarlo.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
public class Buscaminas extends JFrame {
    public static int TAM=10;

    private int tablero[][]=new int[TAM][TAM];  //Representación del tablero
    private int visible[][]=new int[TAM][TAM]; //0 tapado, 1 descubierto, 2 bandera
    private int estado=0; //0 jugando, 1 game over, 2 victoria
    private int casillasVistas=0;  //Contador de casillas vistas

    public Buscaminas(){
        //Configuracion de la ventana
        setVisible(true);
        setSize(405, 440);
        setTitle( &#34;Buscaminas casero. By Jorge Rubira&#34; );
        setResizable(false);

        //Crea el tablero
        crearTablero();

        //Eventos al pulsar el raton
        addMouseListener(new MouseListener() {

            public void mouseReleased(MouseEvent arg) {
                //Si estamos jugando
                if (estado==0){
                    //Obtiene fila y columna pulsada
                    int f=(arg.getY()-40)/40;
                    int c=arg.getX()/40;
                    if (arg.getButton()==MouseEvent.BUTTON1){
                        if (visible[f][c]==0){
                            if (tablero[f][c]==9){
```

```
                            //Si pulsa una mina acaba la partida
                            gameOver();
                        }else{
                            //Si pulsa un terreno lo visualiza ejecutando una funcion recursiv
                            clicCasilla(f,c);
                        }
                    }
                }else if (arg.getButton()==MouseEvent.BUTTON3){
                    if (visible[f][c]==0){
                        visible[f][c]=2;
                    }else if (visible[f][c]==2){
                        visible[f][c]=0;
                    }
                }
            }else{
                crearTablero();
            }
            repaint();
        }

        public void mousePressed(MouseEvent arg0) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseExited(MouseEvent arg0) {}
        public void mouseEntered(MouseEvent arg0) {}
    });

    addWindowListener(new WindowListener() {
        public void windowClosing(WindowEvent arg0) {
            System.exit(0);
        }
        public void windowOpened(WindowEvent arg0) {}
        public void windowIconified(WindowEvent arg0) {}
        public void windowDeiconified(WindowEvent arg0) {}
        public void windowDeactivated(WindowEvent arg0) {}
        public void windowClosed(WindowEvent arg0) {}
        public void windowActivated(WindowEvent arg0) {}
    });
}

public void gameOver(){
    estado=1;
}

public void victoria(){
    estado=2;
}

public void clicCasilla(int f, int c){
    //Si la casilla esta tapada
    if (visible[f][c]==0){
        //Descubre la casilla
        visible[f][c]=1;
        casillasVistas++;
        if (casillasVistas==90){
            //Si llega a las 90 casillas descubiertas gana
            victoria();
        }else{
            //Si no hay minas cercanas
            if (tablero[f][c]==0){
```

```
                //Recorre las casillas cercanas y tambien las ejecuta
                for (int f2=max(0, f-1);f2 &#60; min(TAM,f+2);f2++){
                    for (int c2=max(0,c-1);c2 &#60; min(TAM,c+2);c2++){
                        clicCasilla(f2, c2);
                    }
                }
            }
        }
    }

    public void crearTablero(){
        //Inicializa el tablero
        for (int f=0;f &#60; TAM;f++){
            for (int c=0;c &#60; TAM;c++){
                tablero[f][c]=0;
                visible[f][c]=0;
            }
        }
        estado=0;
        casillasVistas=0;

        //Pone diez minas
        for (int mina=0;mina &#60; 10;mina++){
            //Busca una posición aleatoria donde no haya otra bomba
            int f,c;
            do{
                f=(int)(Math.random()*10);
                c=(int)(Math.random()*10);
            }while(tablero[f][c]==9);
            //Pone la bomba
            tablero[f][c]=9;
            //Recorre el contorno de la bomba e incrementa los contadores
            for (int f2=max(0, f-1);f2 &#60; min(TAM,f+2);f2++){
                for (int c2=max(0,c-1);c2 &#60; min(TAM,c+2);c2++){
                    if (tablero[f2][c2]!=9){ //Si no es bomba
                        tablero[f2][c2]++; //Incrementa el contador
                    }
                }
            }
        }
    }

    public void update(Graphics g){
        paint(g);
    }
    public void paint(Graphics g){
        g.setFont(new Font( &#34;ARIAL&#34; , Font.BOLD, 14));
        g.clearRect(0, 0, getWidth(), 40);
        //Pinta las casillas
        for (int f=0;f &#60; TAM;f++){
            for (int c=0;c &#60; TAM;c++){
                int x=c*40;
                int y=f*40+40;
                if (visible[f][c]==0 &#38;&#38; estado==0){
                    g.setColor(Color.gray);
                    g.fillRect(x, y, 40, 40);
                }else if (visible[f][c]==2 &#38;&#38; estado==0){
                    g.setColor(Color.blue);
```

```
                    g.fillRect(x, y, 40, 40);
                }else if (tablero[f][c] &#60; 9){
                    g.setColor(Color.white);
                    g.fillRect(x, y, 40, 40);
                    if (tablero[f][c]&#62;0){
                        g.setColor(Color.black);
                        g.drawString( &#34;&#34;+tablero[f][c], x+15, y+25);
                    }
                }else{
                    g.setColor(Color.red);
                    g.fillRect(x, y, 40, 40);
                }
                g.setColor(Color.DARK_GRAY);
                g.drawRect(x, y, 40, 40);
            }
        }
        //Texto de estados (no jugando)
        if (estado==1){
            g.drawString( &#34;Game over&#34;, 10, 40);
        }
        if (estado==2){
            g.drawString( &#34;Conseguido!!!&#34;, 10, 40);
        }
    }

    //Funciones para abreviar letras.
    public int max(int a, int b){
        return Math.max(a,b);
    }
    public int min(int a, int b){
        return Math.min(a,b);
    }
    public static void main(String arg[]){
        new Buscaminas();
    }
}
```

Compartir

Temas:

GENBETA DEV DESARROLLO

PUBLICIDAD

Comentarios cerrados

PUBLICIDAD

1

Si te ha gustado, puedes recibir más en tu correo

Tu correo electrónico

SUSCRÍBETE

Te enviamos nuestra newsletter una vez al día, con todo lo que publicamos

TEMAS DE INTERÉS › Desactivar actualización Windows 10 — Deep web y dark web — Actualizar windows — Plantillas PowerPoint — Plantillas Excel — Plar

RECIBE UN EMAIL AL DÍA CON LOS ARTÍCULOS DE GENBETA:

Tu correo electrónico SUSCRIBIR

Síguenos

EN GENBETA HABLAMOS DE...

- Paso a paso
- Linux
- Windows
- desarrolladores
- Desarrollo
- Herramientas para Programadores
- Actualidad
- Web
- Herramientas
- Inteligencia Artificial
- Activismo Online
- Redes sociales

VER MÁS TEMAS

Buscar en Genbeta

SUBIR

TECNOLOGÍA

- Xataka
- Xataka Móvil
- Xataka Foto
- Xataka Android
- Xataka Smart Home
- Xataka Windows
- Xataka Ciencia
- Applesfera
- Genbeta
- Magnet
- Mundo Xiaomi

VIDEOJUEGOS

- 3DJuegos
- Vida Extra
- Millenium
- 3DJuegos PC
- 3DJuegos Guías

ENTRETENIMIENTO

- Sensacine
- Espinof

GASTRONOMÍA

- Directo al Paladar

ESTILO DE VIDA

- Vitónica
- Trendencias
- Trendencias Hombre
- Decoesfera
- Compradiccion
- Poprosa

LATINOAMÉRICA

Xataka México

Sensacine México

3DJuegos LATAM

Directo al Paladar México