Code

MCWW (MODIFIED CLASSIC WUMPUS WORLD) X

-Author: Carlos Pelta

```
package Wumpus;
import java.util.Stack;
/*
 * Agente.java
 *
 *
 * Clase para datos y recorridos
 */

public class Agente {
   private Tablero del mundo; // Mundo a recorrer
   public Tablero del mapa; // Base de conocimientos
   public Ventana ventana;

   private boolean WumpusVivo=true;
   private boolean flecha=true;
   private boolean salir;

   private int pasosWumpus; // Pasos hasta que el Wumpus despierte
   private int pasos=0; // Pasos recorridos de momento

   //Variables de control de ejecución (Paso a paso o Contínuo)
   private boolean bystep=true; // Inicio en pausa
   private boolean restart;
   private boolean go;


    public Agente(Tablero tablero, Ventana ventana) {
      // Seteo de Variables
      this.ventana=ventana;
      mundo=tablero;
      mapa=new Tablero(mundo.getAlto( ),mundo.getAncho( ));
      pasosWumpus=(int) (java.lang.Math.random( )*20)+1;
```

```
        // Agregar los mapas a la ventana
        ventana.addMapa(new Mapa(mundo));
        ventana.addMapa(new Mapa(mapa));
    }

    public void iniciar( ){
        int X=0,Y=0;
        boolean seguirCamino=false;
        Stack camino=new Stack( );
        Point p;
        while(true){
            while(!((salir&&X==0&&Y==0)||restart)){
                marcarMapa(X,Y); // Marca las entidades en el mapa (Buscar al
Wumpus si se ha despertado)
                ventana.repaint( ); //Redibujar la ventana
                try{  //Espera o para si está en modo Paso a Paso
                    if(bystep&&!go){
                        while(!go) Thread.sleep(100);
                    }else Thread.sleep(700);
                }catch(Exception e){ }
                go=false;
                // Elimina al agente del mapa, y no aparece repetido
                mundo.setNoAgente(X,Y);
                mapa.setNoAgente(X,Y);
                if(!(restart||(salir&&X==0&&Y==0))) //Sale si se reinicia
                if(mundo.getPercepcion(X,Y).getOro( )){ //Sale si hay oro
                    ventana.mensaje("Encuentro oro");
                    mundo.getPercepcion(X,Y).setOro(false);
                    camino=buscarCamino(new Point(X,Y),new Point( ));
                    p = (Point)camino.pop( );
                    X = p.x;
                    Y = p.y;
                    salir=true;

                    seguirCamino=true;

                    }else{
                    //Si hay algún casillero seguro, avanza hacia él
                    if(seguirCamino&&!camino.empty( )){
                        p=(Point)camino.pop( );
```

```
                X=p.x;
                Y=p.y;
             }else{
                seguirCamino=false;

if(mapa.inMapa(X+1,Y)&&mapa.getSeguro(X+1,Y)&&!(mapa.getVisita
do(X+1,Y))){
                   X++; //Derecha
                }else if(mapa.inMapa(X-1,Y)&&mapa.getSeguro(X-
1,Y)&&!(mapa.getVisitado(X-1,Y))){
                   X--; //Izquierda
                }else
if(mapa.inMapa(X,Y+1)&&mapa.getSeguro(X,Y+1)&&!(mapa.getVisita
do(X,Y+1))){
                   Y++; //Arriba
                }else if(mapa.inMapa(X,Y-1)&&mapa.getSeguro(X,Y-
1)&&!(mapa.getVisitado(X,Y-1))){
                   Y--; //Abajo
                }else if(mapa.hayLibres( )){ // Si no, busca casillero libre
                   p = mapa.getLibre( );
                   camino=buscarCamino(new Point(X,Y),p);
                   if(!camino.empty( )){
                   p = (Point)camino.pop( );
                   X = p.x;
                   Y = p.y;
                   seguirCamino=true;
                   }
                }else{ // Y si no, salir
                   ventana.mensaje("No Hay más Caminos");
                   camino=buscarCamino(new Point(X,Y),new Point( ));
                   if(!camino.empty( )){
                      p = (Point)camino.pop( );

                      X = p.x;
                      Y = p.y;
                   }
                   seguirCamino=true;
                   salir=true;
                   .
                   .
```

.

```
   public void buscarWumpus( ){
      /*  La funcion se encarga de matar al Wumpus, sólo si tiene 50% o
más de
       *  certeza (Si sólo hay 2 cuadros donde pueda estar).
       *  Recorre todo el mapa conocido contando cada aparición del
Wumpus
       * sólo intenta disparar si cuenta 2 apariciones exactas.
       */
      int i,j,contar=contarWumpus( );
      if(contar==2||contar==1)
         for(i=0;i<mapa.getAncho( );i++)
            for(j=0;j<mapa.getAlto( );j++)
               if(mapa.getWumpus(i,j)&&flecha&&wumpusVivo){

                  if(mundo.callar(i,j)){
                     wumpusVivo=false;
                     ventana.mensaje("Mato al Wumpus");
                     mapa.setNoWumpus(i,j);
                  }else ventana.mensaje("Fallé");
                  flecha=false;
               }
   }
   public int contarWumpus( ){
      int i,j,contar=0;
      for(i=0;i<mapa.getAncho( );i++)
         for(j=0;j<mapa.getAlto( );j++)
            if(mapa.getWumpus(i,j))
               contar++;
      return contar;
   }
   // Funciones para controlar el movimiento (Paso a paso, contínuo o
reinicio)
   public void step( ){
      bystep=true;
      go=true;
   }
   public void go( ){
      bystep=false;
```

```
        go=true;
    }
    public void restart( ){
        restart=true;
    }
}
```

```
package Wumpus;
/*
 *Main. Java
 *
 *
 */
public class Main {

    public static Tablero del mundo;
    public static Agente agente;
    public static void main(String[ ] args) {
        //Inicializar variables
        mundo=new Tablero(8,8);
        Ventana ventana=new Ventana( );
        mundo.inicializar( );
        //Mostrar ventana
        ventana.setVisible(true);
        //Comenzar recorrido
        agente=new Agente(mundo,ventana);
        agente.iniciar( );
    }
}
```

```
package Wumpus;
/*
 * Mapa.java
 *
 *
 * Clase que muestra en pantalla el Mundo y la Base de conocimientos
 * Extiende JPanel y sobreescribe su método para representarla.
 */
import java.awt.*;
```

```
import javax.swing.*;
/**
 *
 *
 */
public class Mapa extends JPanel {
   private Tablero tablero;
   private Image wumpus;
   private Image oro;
   private Image pozo;
   private Image agente;
   public Mapa(Tablero t){
      tablero=t;
      Toolkit tk = Toolkit.getDefaultToolkit( );

      // Define el color de fondo
      setBackground(Color.WHITE);
      // Marca los bordes
      setBorder(BorderFactory.createLineBorder(new Color(0, 0, 0)));

      // Determina el tamaño inicial del mapa
      this.setPreferredSize(new Dimension(300,300));

      // Cargar las imágenes
      wumpus=tk.getImage("img/wumpus.gif");
      oro=tk.getImage("img/oro.gif");
      pozo=tk.getImage("img/pozo.gif");
      agente=tk.getImage("img/agente.gif");
   }

   public void paintComponent(Graphics g){
      // Dibujar el fondo
      super.paintComponent(g);

      // Inicializar Variables
      Rectangle rect=g.getClipBounds( );
      int i,j;
      int alto=tablero.getAlto( );
      int ancho=tablero.getAncho( );
```

```
      //Determina el tamaño de la cuadrícula. Depende de la cantidad de
cuadros
      // Y el tamaño de la ventana
      int separacionH=(int)(rect.getWidth( )/ancho);
      int separacionV=(int)(rect.getHeight( )/alto);

      //Entidades
      for(i=0;i<ancho;i++)
         for(j=0;j<alto;j++){
            Percepcion estado=tablero.getPercepcion(i,j);
            if(estado.getOro( ))
g.drawImage(oro,i*separaciónH,j*separaciónV,separaciónH,separaciónV
,this);
            if(estado.getAgente( )){

g.fillRect(i*separaciónH,j*separaciónV,separaciónH,separaciónV);

g.drawImage(agente,i*separaciónH,j*separaciónV,separaciónH,se-
paraciónV,this);
            }
            // Si se cree que existe pozo y Wumpus, dibujar juntos
            if(estado.getPozo( )&&estado.getWumpus( )){

g.drawImage(wumpus,i*separaciónH,j*separaciónV,separaciónH/2,se-
paraciónV/2,this);

g.drawImage(pozo,i*separaciónH+separaciónH/2,j*separaciónV+se-
paraciónV/2,separaciónH/2,separaciónV/2,this);
            } else { // Si no, dibujar ocupando todo el cuadro

               if(estado.getPozo( ))
g.drawImage(pozo,i*separaciónH,j*separaciónV,separaciónH,separación
V,this);
                if(estado.getWumpus( ))
g.drawImage(wumpus,i*separaciónH,j*separaciónV,separaciónH,se-
paraciónV,this);
            }
         }

      //Cuadrícula
```

```
        g.setColor(Color.BLACK);
        for(i=1;i<alto;i++){

g.drawLine(i*separaciónH,rect.y,i*separaciónH,rect.y+rect.height);
        }
        for(i=1;i<ancho;i++){

g.drawLine(rect.x,i*separaciónV,rect.width+rect.x,i*separaciónV);
        }
    }
}

package Wumpus;
/*
 * Percepción.java
 *
 *
 * La clase contiene una serie de booleanos y las funciones
correspondientes
 * para leer o escribirlos
 */

public class Percepción {
    private boolean grito;   //Pistas
    private boolean brillo;
    private boolean brisa;

    private boolean wumpus;
    private boolean pozo;   //Entidades en el mundo, o ayuda de memorias
en el mapa
    private boolean oro;
    private boolean agente;

    private boolean seguro;
    private boolean visitado;  // En el mapa, marcan lo único de lo que
tengo certeza
          .
          .
          .
```

```
/*
 * Point.java
 *
 *
 * Clase que marca una posición; incluye una propiedad Padre, que
 * el algoritmo A* utiliza para reconstruir el camino.
 */

package Wumpus;

public class Point {
public int x;
public int y;
public Point padre;

   public Point( ){
      this(0,0);
   }
   public Point(int x, int y) {
      this.x=x;
      this.y=y;
   }
   public Point(int x, int y, Point parent) {
      this.x=x;
      this.y=y;
      padre=parent;
   }
   public boolean equals(Point p){
      if(p.x==x && p.y==y) return true;
      return false;
   }
}

package Wumpus;
/**
 * Tablero.java
 *
 *
 */
public class Tablero {
```

```
    private int alto;
    private int ancho;
    Percepción[ ][ ] mapa; //Array de Percepción, con las entidades
correspondientes

    /** Crea un nuevo Tablero
     *@param tamañoH Ancho
     *@param tamañoV Alto
     */
    public Tablero(int tamañoH,int tamañoV) {
        alto=tamañoV;
        ancho=tamañoH;
        mapa = new Percepción[tamañoV][tamañoH];
        clean( );
    }
    public int getAncho( ){
        return ancho;
    }
    public int getAlto( ){
        return alto;
    }
    public boolean getVisitado(int X, int Y){
        return !(inMapa(X,Y))||mapa[X][Y].getVisitado( );
    }
    public Percepción getPercepción(int X,int Y){
        return mapa[X][Y];
    }


    /*
     * Es necesario hacer la función de esta manera, en caso contrario copia
un
     * puntero, y puede sobreescribir propiedades del Mundo

     */
    public void setPercepción(int X, int Y, Percepción percepción){
        mapa[X][Y].setBrillo(percepción.getBrillo( ));
        mapa[X][Y].setBrisa(percepción.getBrisa( ));
        mapa[X][Y].setGrito(percepción.getGrito( ));
        mapa[X][Y].setWumpus(percepción.getWumpus( ));
```

```
        mapa[X][Y].setPozo(percepción.getPozo( ));
        mapa[X][Y].setOro(percepción.getOro( ));
    }
                .
                .
                .

package Wumpus
/*
 *Ventana. Form
 */

 /**
 *
 *
 */

<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.0"
type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <Properties>
    <Property name="defaultCloseOperation" type="int" value="3"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <AuxValues>
                .
                .
package Wumpus;
/*
 * Ventana.java
 *
 */

/**
 *
 *
 */
```

```
public class Ventana extends javax.swing.JFrame {

    /** Creates new form Ventana */
    public Ventana( ) {
        initComponents( );
    }
        /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code
">//GEN-BEGIN:initComponents
    private void initComponents( ) {
        javax.swing.JPanel Abajo;
        javax.swing.JPanel Arriba;
        javax.swing.JButton btnIniciar;
        javax.swing.JButton btnNvoTblr;
        javax.swing.JButton btnPaso;
        javax.swing.JPanel jPanel1;
        javax.swing.JPanel jPanel3;
        javax.swing.JScrollPane jScrollPane1;
        javax.swing.JLabel lblMapa;
        javax.swing.JLabel lblMundo;
                    .
                    .
    // End of variables declaration//GEN-END:variables

    // Dibujar los mapas en la ventana
    public void addMapa(javax.swing.JPanel mapa){
        jPanel2.add(mapa);
        pack( );
    }
    public void mensaje(String text){
        // Agregar el mensaje al texto de la caja de texto
        jTextArea1.append("\n"+text);
        jTextArea1.select(jTextArea1.getText( ).length(
),jTextArea1.getText( ).length( ));
    }
}
```