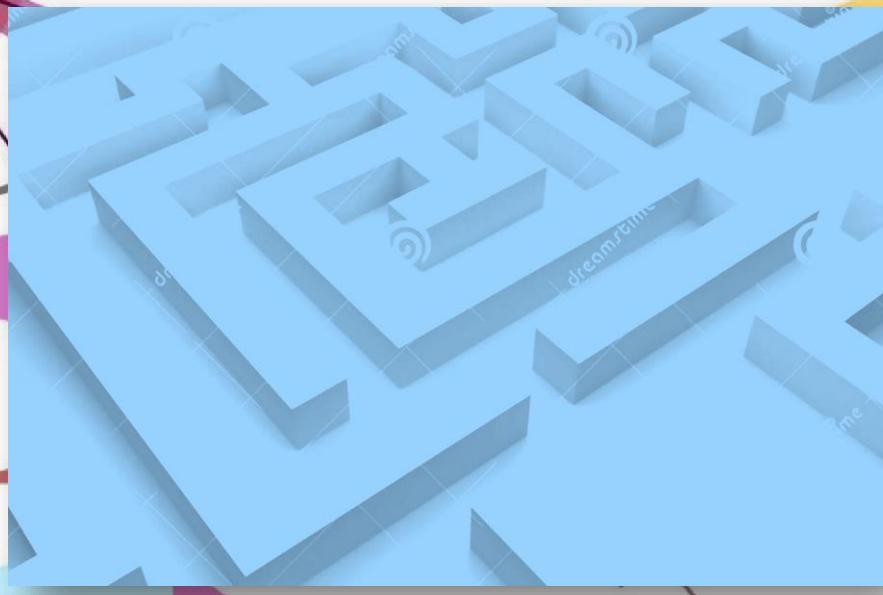


LSI MAZE GAME



Méthodologies de l'intelligence artificielle

Encadré par : MAIT KBIR

Réalisé par :

AIT KADDI Ikram

Sommaire

	Générale
Introduction	3
Objectif	4
Description de l'application	4
Structure du projet	5
Code Source.....	6
Interfaces	34
<i>Lancement du jeu</i>	34
<i>Choix de labyrinthe</i>	34
<i>Demande d'aide</i>	35
<i>Labyrinthe(21* 21)</i>	35
<i>Labyrinthe(41* 41)</i>	36
<i>Labyrinthe(61* 61)</i>	37
<i>Labyrinthe(81* 81)</i>	38
<i>Visualisation du chemin</i>	39
<i>Refus d'aide</i>	39
<i>Fin du jeu</i>	40
Conclusion	40

Introduction Générale

Les algorithmes de recherche de chemins traitent le problème de trouver un chemin partant d'une source vers une destination tout en évitant les obstacles et minimisant les coûts (temps, distance, risques, carburant, prix, etc.). Il s'agit d'un défi courant en matière de programmation. Principalement connus dans la navigation et les jeux vidéos et d'autres.

Parmis les algorithmes utilisés:

L'algorithme BFS garantit le chemin le plus court, la recherche en largeur explore de manière uniforme dans toutes les directions. Il est aussi utile pour la génération procédurale de carte, la mesure de topologie et d'autres types d'analyse. Cela peut être l'algorithme de choix pour identifier les lieux d'intérêt proches dans le GPS.

L'algorithme DFS ne garantit pas le chemin le plus court. Traverse en explorant autant que possible chaque chemin avant de revenir en arrière.

Aussi importante que la recherche en largeur, celle-ci peut être utilisée pour générer un ordre topologique, construire des arbres de décisions, générer des labyrinthes,,

L'algorithme A* trouve des chemins vers un endroit, il priorise les chemins qui semblent se rapprocher d'un objectif. C'est plus ou moins l'algorithme standard (de premier choix) pour trouver rapidement un itinéraire entre deux lieux.

Objectif

Développement d'une application Desktop Java pour la réalisation du jeu de labyrinthe, en implémentant les algorithmes de recherche de chemins (BFS, DFS ,A*).

Description de l'application

“**LSI MAZE GAME**” permet de manipuler un agent, l’unique objet mobile, à l'aide du clavier, pour atteindre la sortie du labyrinthe. Le joueur dispose d'un score nul au début qu'il peut alimenter en

gagnant des bonus, 5 points par bonus. Il perd 2 points quant-il traverse un obstacle, aussi il perd des points lorsqu'il demande de l'aide pour visualiser temporairement un chemin pour atteindre un bonus ou la sortie (1 point pour un bonus et 3xa points pour la sortie, a dépend de la stratégie à utiliser).

Les obstacles se trouvent initialement sur un des chemins vers la sortie et les bonus sont répartis aléatoirement sur le labyrinthe. Chaque fois qu'un bonus est consommé, un autre apparaît dans une autre zone choisie aléatoirement.

L'application prend en considération ce qui suit :

- Lors de la demande de d'aide, permettre le choix de la méthode de résolution : recherche en profondeur d'abord ($a=1/3$), recherche en largeur d'abord ($a=2/3$) ou A* ($a=1$) ;
- Choisir un des modèles de labyrinthes, chargés à partir des fichiers ;
- Fixer le nombre des obstacles et le nombre des bonus ;
- Gérer le temps alloué au joueur et arrêter la partie si ce temps expire ou si l'agent atteint la sortie.

Structure du projet



Code Source

La classe AStarSearchEngine.java

```

1 package jeu.classes;
2
3 import java.awt.Dimension;
4 import java.util.*;
5 // class pour algo astar
6 public class AStarSearchEngine extends AbstractSearchEngine {
7
8     ArrayList<Dimension> L2= new ArrayList<Dimension>(), T=new ArrayList<Dimension>();
9     PriorityQueue<Dimension> L1 = new PriorityQueue<Dimension>();
10    Dimension predecessor[][][];
11    Dimension loc = new Dimension();
12    Dimension G = new Dimension();
13    public AStarSearchEngine(int width, int height, Dimension Loc, Dimension g) {
14        super(width, height);
15        this.loc = Loc;
16        this.g = g;
17        predecessor = new Dimension[width][height];
18        for (int i=0; i<width; i++)
19            for (int j=0; j<height; j++)
20                predecessor[i][j] = null;
21
22        L1.add(new Case(loc,0,-1.0));
23        // On suppose que T contient un seul But
24        T.add(g);
25
26        doAStar();
27
28        // now calculate the shortest path from the predecessor array:
29        maxDepth = 0;
30        if (!isSearching) {
31            searchPath[maxDepth++] = G;
32            for (int i=0; i<2000; i++) {
33                searchPath[maxDepth] = predecessor[searchPath[maxDepth - 1].width][searchPath[maxDepth - 1].height];
34                maxDepth++;
35                if (equals(searchPath[maxDepth - 1], Loc)) break; // back to starting node
36            }
37        }
38    }
39
40    // On suppose que T contient un seul But (globalLoc)
41    public double h(Dimension d) {
42        return Math.sqrt((G.height-d.height)*(G.height-d.height)+(G.width-d.width)*(G.width-d.width));
43    }
44
45    private void doAStar() {
46
47        while (!L1.isEmpty()) {
48
49            currentLoc = L1.remove(); // éléments triés dans L1 par ordre croissant de f
50
51            L2.add(currentLoc);
52            if(T.contains(currentLoc)){
53                System.out.println("But trouvé : (" + currentLoc.width +", " + currentLoc.height+ ")");
54                isSearching = false;
55                break;
56            }
57            else{
58
59                Dimension [] connected = getPossibleMoves(currentLoc);
60                for (int i=0; i<4; i++) {
61
62                    if ( connected[i]!=null && !L1.contains(connected[i]) && !L2.contains(connected[i])) {
63                        Case casei= new Case(connected[i],((Case)currentLoc).getG()+1,((Case)currentLoc).getG()+1+h(connected[i]));
64                        predecessor[connected[i].width][connected[i].height] = currentLoc;
65                        L1.add(casei);
66                    }
67                }
68            }
69        }
70    }
71
72}
73
74

```

La classe AbstractSearchEngine.java

```
1 package jeu.classes;
2 /**
3  * Title: AbstractSearchEngine<p>
4  * Description: Abstract search engine for searching paths in a maze<p>
5  * Copyright: Copyright (c) Mark Watson, Released under Open Source Artistic License<p>
6  * Company: Mark Watson Associates<p>
7  * @author Mark Watson
8  * @version 1.0
9 */
10
11 import java.awt.Dimension;
12 //Abstract class pour les algorithmes de recherche
13 public class AbstractSearchEngine {
14     public AbstractSearchEngine(int width, int height) {
15         maze = new Maze(width, height);
16         initSearch();
17     }
18     public Maze getMaze() { return maze; }
19     protected Maze maze;
20 /**
21     * We will use the Java type Dimension (fields width and height will
22     * encode the coordinates in x and y directions) for the search path:
23     */
24     protected Dimension [] searchPath = null;
25     protected int pathCount;
26     protected int maxDepth;
27     protected Dimension startLoc, goalLoc, currentLoc;
28     protected boolean isSearching = true;
29
30     protected void initSearch() {
31         if (searchPath == null) {
32             searchPath = new Dimension[2000];
33             for (int i=0; i<100; i++) {
34                 searchPath[i] = new Dimension();
35             }
36         }
37         pathCount = 0;
38         startLoc = maze.startLoc;
39         currentLoc = startLoc;
40         goalLoc = maze.goalLoc;
41         searchPath[pathCount++] = currentLoc;
42     }
43
44     protected boolean equals(Dimension d1, Dimension d2) {
45         return d1.getWidth() == d2.getWidth() && d1.getHeight() == d2.getHeight();
46     }
47
48     public Dimension [] getPath() {
49         Dimension [] ret = new Dimension[maxDepth];
50         for (int i=0; i<maxDepth; i++) {
51             ret[i] = searchPath[i];
52         }
53         System.out.println("max sdepyh " + maxDepth);
54         return ret;
55     }
56
57     public Dimension [] getPathInLocation(Dimension loc) {
58         Dimension [] ret = new Dimension[maxDepth];
59         for (int i=0; i<maxDepth; i++) {
60             ret[i] = searchPath[i];
61         }
62         return ret;
63     }
64     protected Dimension [] getPossibleMoves(Dimension loc) {
65         Dimension tempMoves [] = new Dimension[4];
66         tempMoves[0] = tempMoves[1] = tempMoves[2] = tempMoves[3] = null;
67         int x = loc.width;
68         int y = loc.height;
69         int num = 0;
70         if (maze.getValue(x - 1, y) == 0 || maze.getValue(x - 1, y) == Maze.GOAL_LOC_VALUE) {
71             tempMoves[num++] = new Dimension(x - 1, y);
72         }
73         if (maze.getValue(x + 1, y) == 0 || maze.getValue(x + 1, y) == Maze.GOAL_LOC_VALUE) {
74             tempMoves[num++] = new Dimension(x + 1, y);
75         }
76         if (maze.getValue(x, y - 1) == 0 || maze.getValue(x, y - 1) == Maze.GOAL_LOC_VALUE) {
77             tempMoves[num++] = new Dimension(x, y - 1);
78         }
79         if (maze.getValue(x, y + 1) == 0 || maze.getValue(x, y + 1) == Maze.GOAL_LOC_VALUE) {
80             tempMoves[num++] = new Dimension(x, y + 1);
81         }
82         return tempMoves;
83     }
84 }
85 }
```

La classe Aide.java

```
1 package jeu.classes;
2 import java.awt.Color;
3 import javax.swing.JButton;
4 import javax.swing.JFrame;
5 import javax.swing.JLabel;
6 import javax.swing.Timer;
7 // class qui permet au joueur d'observer les options de l'aide
8 public class Aide extends JFrame{
9     public static int seconds, minutes, hours;
10    public Timer chronometre;
11    private JButton runStop, restart;
12    private JLabel time;
13    public Aide(){
14
15        this.setTitle("Help me :)");
16        this.setSize(400, 200);
17        //this.setDefaultCloseOperation(EXIT_ON_CLOSE);
18        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
19        this.setLocationRelativeTo(null);
20        this.getContentPane().setLayout(null);
21
22        // option
23        JLabel a1 = new JLabel("Press the B key to view the path with BFS");
24        a1.setForeground(new Color(13,144,178));
25        this.add(a1);
26        a1.setBounds(10, 10, 400, 100);
27        JLabel a2 = new JLabel("Press the C key to view the path with DFS");
28        a2.setForeground(new Color(13,144,178));
29        this.add(a2);
30        a2.setBounds(10, 30, 400, 100);
31        JLabel a3 = new JLabel("Press the Space key to view BONUS");
32        a3.setForeground(new Color(13,144,178));
33        this.add(a3);
34        a3.setBounds(10, 50, 400, 100);
35        JLabel a4 = new JLabel("Press the E key to view the path with ASTAR");
36        a4.setForeground(new Color(13,144,178));
37        a4.setBounds(10, 70, 400, 100);
38        this.add(a4);
39
40    }
41 }
42 
```

La classe Bonus.java

```
1 package jeu.classes;
2
3 import java.awt.Dimension;
4 import java.awt.Graphics;
5 import java.awt.Rectangle;
6 import java.awt.image.BufferedImage;
7 import java.io.File;
8 import java.io.IOException;
9 import javax.imageio.ImageIO;
10 // class pour afficher les bonus et leur modification dans l'interface
11 public class Bonus extends Entité {
12     private BufferedImage image;
13     protected final int LARGEURE = JeuDeLabyrinthe.tailleCaze;
14     protected final int HAUTEURE = JeuDeLabyrinthe.tailleCaze;
15     public Dimension Loc = new Dimension();
16     public Bonus(int x, int y, IDEntité id) throws IOException {
17         super(x, y, id);
18         this.Loc.width = x;
19         this.Loc.height = y;
20         this.setLocB(Loc);
21         if(JeuDeLabyrinthe.tailleCaze == 12){
22             image = ImageIO.read(new File("files/momo/bonus.png"));
23         }else{
24             image = ImageIO.read(new File("files/momo/bonus22.png"));
25         }
26     }
27     //modification de bonus
28     @Override
29     protected void update() {
30         this.Loc.width = x;
31         this.Loc.height = y;
32         this.setLocB(Loc);
33     }
34     //dessiner le bonus
35     @Override
36     protected void render(Graphics g) {
37         g.drawImage(image, x, y, null);
38     }
39 }
```

```

34     //dessiner le bonus
35     @Override
36     protected void render(Graphics g) {
37         g.drawImage(image, x, y, null);
38     }
39     @Override
40     protected Rectangle entourer() {
41         return new Rectangle(x,Y,LARGEURE ,HAUTEURE);
42     }
43     // determiner la location de bonus
44     public Dimension getLocB(){
45
46         this.Loc.width = this.Loc.width /JeuDeLabyrinthe.tailleCaze;
47         this.Loc.height = this.Loc.height /JeuDeLabyrinthe.tailleCaze;
48     return this.Loc;
49 }
50     public void setLocB(Dimension loc){
51         this.Loc=loc;
52     }
53 }
54
55

```

La classe BreadthSearchEngine.java

```

1 package jeu.classes;
2
3 import java.awt.Dimension;
4 import java.util.*;
5 // class pour definir l'algorithme BFS
6 public class BreadthFirstSearchEngine extends AbstractSearchEngine {
7
8     ArrayList<Dimension> L= new ArrayList<Dimension>(), T=new ArrayList<Dimension>();
9     LinkedList<Dimension> file = new LinkedList<Dimension>();
10    Dimension predecessor[][][];
11    Dimension loc = new Dimension();
12    Dimension G = new Dimension();
13    public BreadthFirstSearchEngine(int width, int height, Dimension Loc,Dimension g) {
14        super(width, height);
15        this.loc = Loc;
16        this.G =g;
17
18        predecessor = new Dimension[width][height];
19        for (int i=0; i<width; i++)
20            for (int j=0; j<height; j++)
21                predecessor[i][j] = null;
22
23        //L.add(startLoc);
24        //file.add(startLoc);
25        file.add(loc);
26        // T.add(goalLoc);
27        // T.add(g);
28        doBFS();
29
30        // now calculate the shortest path from the predecessor array:
31        maxDepth = 0;
32        if (!isSearching) {
33            searchPath[maxDepth++] = G;
34            for (int i=0; i<1000; i++) {
35                searchPath[maxDepth] = predecessor[searchPath[maxDepth - 1].width][searchPath[maxDepth - 1].height];
36                maxDepth++;
37                if (equals(searchPath[maxDepth - 1], loc)) break; // back to starting node
38            }
39        }
40    }
41

```

```

42     private void doBFS() {
43
44         while (!file.isEmpty()) {
45
46             currentLoc = file.removeFirst();
47             L.add(currentLoc);
48             if(T.contains(currentLoc)){
49                 System.out.println("But trouvé : (" + currentLoc.width +", " + currentLoc.height+ ")");
50                 isSearching = false;
51                 break;
52             }
53             else{
54
55                 Dimension [] connected = getPossibleMoves(currentLoc);
56                 for (int i=0; i<4; i++) {
57
58                     if ( connected[i]!=null && !file.contains(connected[i]) && !L.contains(connected[i])) {
59
60                         predecessor[connected[i].width][connected[i].height] = currentLoc;
61                         file.addLast(connected[i]);
62
63                     }
64                 }
65             }
66         }
67     //285 //185
68

```

La classe Case.java

```

2 package jeu.classes;
3 import java.awt.Dimension;
4 // comparaison de chemin avec Astar
5 public class Case extends Dimension implements Comparable{
6
7     private double g;    // Estimation du coût du chemin parcouru
8     private double f;    // Estimation du coût du chemin qui reste
9
10    public Case(int width, int height) {
11        super(width, height);
12        f=0;
13        g=0;
14    }
15
16    public Case(Dimension d, double g,double f) {
17        super(d);
18        this.g = g;
19        this.f = f;
20    }
21
22    public double getF() {
23        return f;
24    }
25
26    public void setF(double f) {
27        this.f = f;
28    }
29
30    public double getG() {
31        return g;
32    }
33
34    public void setG(double g) {
35        this.g = g;
36    }
37

```

```

37
38     @Override
39     public int compareTo(Object o) {
40         if(f<((Case)o).f)
41             return -1;
42         else if(f>((Case)o).f)
43             return 1;
44         else
45             return 0;
46     }
47
48     // Pour de l'affichage en cas de besoin
49     public String toString() {
50         String s;
51         s = "("+this.width+", "+this.height+"), f= "+f+" et g= "+g;
52         return s;
53     }
54
55 }
56

```

La classe Chemin.java

```

1 package jeu.classes;
2
3 import java.awt.Color;
4 import java.awt.Graphics;
5 import java.awt.Rectangle;
6
7 //classe qui permet la représentation des chemins dans maze
8 public class Chemin extends Entité {
9     protected final int LARGEURE = 6;
10    protected final int HAUTEURE = 6;
11    //constructeur
12    public Chemin(int x, int y, IDENTité id) {
13        super(x, y, id);
14    }
15    //modification de chemin
16    @Override
17    protected void update() {
18    }
19    //dessin de chemin avec une case de taille JeuDeLabyrinthe.tailleCase
20    @Override
21    protected void render(Graphics g) {
22        g.setColor(Color.white);
23        g.fillRect( x , y , JeuDeLabyrinthe.tailleCase , JeuDeLabyrinthe.tailleCase );
24    }
25    //methode implementer pour les intersections
26    @Override
27    protected Rectangle entourer() {
28        return new Rectangle(x,y,LARGEURE,HAUTEURE);
29    }
30
31 }
32

```

La classe Chronometer.java

```
package jeuendetire.classes;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Timer;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import java.awt.Font;
import static jeuendetire.classes.Fenetre.frame;

//Lancer un chrono pendant un délai défini

public class Chronometer extends JFrame implements ActionListener {

    public static int seconds, minutes, hours;
    public Timer chronometer;
    private JButton runStop;
    private JLabel time;
    private JLabel timeOver;

    //Constructeur pour initialiser les attributs et les éléments de l'interface Chronometer
    public Chronometer() {
        this.setTitle("Chronometer");
        this.setBounds(0, 0, 300, 100);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        frame.setResizable(false);
        this.getContentPane().setLayout(new FlowLayout(FlowLayout.LEFT));
        runStop = new JButton("Stop");
        runStop.setBackground(new Color(13,144,178));
        runStop.addActionListener(this);

        seconds = 0;
        minutes = 0;
        hours = 0;

        time = new JLabel(hours + " : " + minutes + " : " + seconds + " | " + "0 : " + JeuDeTire.minuteOver + " : " + JeuDeTire.secondeOver);
        time.setFont(new Font("Serif", Font.BOLD, 19));
        this.add(runStop);
        this.add(time);

        chronometer = new Timer(1000, this);
        chronometer.start();
    }

    //Redéfinition de la méthode actionPerformed pour stoper le chronomètre après un délai
    @Override
    public void actionPerformed(ActionEvent e) {
        if (minutes == 1 && seconds == 0 && JeuDeTire.minuteOver == 1 && JeuDeTire.secondeOver == 0) {
            JeuDeTire.minuteJeu = 1;
            chronometer.stop();
        }
        if (minutes == 1 && seconds == 30 && JeuDeTire.minuteOver == 1 && JeuDeTire.secondeOver == 30) {
            JeuDeTire.minuteJeu = 1;
            chronometer.stop();
        }
        if (seconds == 50 && minutes == 1 && JeuDeTire.minuteOver == 1 && JeuDeTire.secondeOver == 50) {
            JeuDeTire.minuteJeu = 1;
            chronometer.stop();
        }
        if (minutes == 3 && seconds == 0 && JeuDeTire.minuteOver == 3) {
            JeuDeTire.minuteJeu = 1;
            chronometer.stop();
        }
        if (e.getSource() instanceof Timer) {
            seconds++;
            if (seconds == 60) {
                minutes++;
                seconds = 0;
                if (minutes == 60) {
                    hours++;
                    minutes = 0;
                }
            }
        }
        time.setText(hours + " : " + minutes + " : " + seconds + " | " + "0 : " + JeuDeTire.minuteOver + " : " + JeuDeTire.secondeOver);
        time.setFont(new Font("Serif", Font.BOLD, 19));
        return;
    }
    else if (e.getSource() instanceof JButton) {
        JButton button = (JButton) e.getSource();
        if (button.getText().equals("run")) {
            chronometer.start();
            button.setText("Stop");
        } else if (button.getText().equals("Stop")) {
            chronometer.stop();
            button.setText("run");
        }
        else if (button.getText().equals("restart")) {
            seconds = 0;
            minutes = 0;
            hours = 0;
            chronometer.stop();
            time.setText(hours + " : " + minutes + " : " + seconds);
        }
    }
}
}
```

La classe Clavier.java

```
1 package jeu.classes;
2
3 import java.awt.event.KeyEvent;
4 import java.awt.event.KeyListener;
5 import javax.swing.JFrame;
6 import javax.swing.JOptionPane;
7 //la classe clavier qui permet d'écouter les évènements de clavier
8 public class Clavier implements KeyListener{
9     GérantEntité gérant;
10    public Clavier(GérantEntité gérant){
11        this.gérant = gérant;
12    }
13    @Override
14    public void keyTyped(KeyEvent e) {
15    }
16    // on tape sur la touche
17    @Override
18    public void keyPressed(KeyEvent e) {
19        int touche = e.getKeyCode();
20        for(int i = 0; i < gérant.getEntités().size(); i++){
21            Entité entité = gérant.getEntités().get(i);
22            if(entité.getId() == IDEntité.JOUEUR){
23                if(touche == KeyEvent.VK_UP) gérant.setHaut(true);
24                if(touche == KeyEvent.VK_DOWN) gérant.setBas(true);
25                if(touche == KeyEvent.VK_LEFT) gérant.setGauche(true);
26                if(touche == KeyEvent.VK_RIGHT) gérant.setDroite(true);
27                if(touche == KeyEvent.VK_A){ Aide aide = new Aide();
28                                aide.setVisible(true);
29                            }
30                //ici on demande l'aide le score va diminuer selon la méthode demandée
31                if( Joueur.SCORE>=2){
32                    if(touche == KeyEvent.VK_B) {
33
34                        gérant.setBfs(true);
35                        JeuDeLabyrinthe.s = 0;
36                        Joueur.SCORE=Joueur.SCORE-2;
37                    }
38                    if( Joueur.SCORE>=1){
39                        if(touche == KeyEvent.VK_C){
40                            gérant.setDfs(true);
41                            JeuDeLabyrinthe.s = 0;
42                            Joueur.SCORE=Joueur.SCORE-1;
43                        }
44
45                    if( Joueur.SCORE>=1){
46                        if(touche == KeyEvent.VK_SPACE){
47                            gérant.setBonus(true);
48                            System.out.println("bonus sélectionné");
49                            JeuDeLabyrinthe.s = 0;
50
51                            Joueur.SCORE=Joueur.SCORE-1;
52                        }
53                    if( Joueur.SCORE>=3){
54                        if(touche == KeyEvent.VK_E){
55                            JeuDeLabyrinthe.s = 0;
56                            Joueur.SCORE = Joueur.SCORE-3;
57                            gérant.setAstar(true);
58                        }
59                    // si le score est insuffisant pour avoir l'aide ,une boîte de dialogue s'affichera.
60                    if(touche == KeyEvent.VK_E && Joueur.SCORE<3){
61                        JFrame frame = new JFrame();
62                        JOptionPane.showMessageDialog(frame,"vous devez avoir au moins 3");
63                    }
64                    if(touche == KeyEvent.VK_SPACE&& Joueur.SCORE<1){
65                        JFrame frame = new JFrame();
66                        JOptionPane.showMessageDialog(frame,"vous devez avoir au moins 1");
67                    if(touche == KeyEvent.VK_C && Joueur.SCORE<1){
68                        JFrame frame = new JFrame();
69                        JOptionPane.showMessageDialog(frame,"vous devez avoir au moins 1");
70                    if(touche == KeyEvent.VK_B && Joueur.SCORE<2){
71                        JFrame frame = new JFrame();
72                        JOptionPane.showMessageDialog(frame,"vous devez avoir au moins 2 ");
73                    }
74                }
75            }
76        }
77    }
78}
```

```

77 //la touche est lâchée
78 @Override
79 public void keyReleased(KeyEvent e) {
80     int touche = e.getKeyCode();
81     gérant.getEntités().stream().filter((entité) -> (entité.getId() == IDEntité.GOUEUR)).map((_item) -> {
82         if(touche == KeyEvent.VK_UP) gérant.setHaut(false);
83         return _item;
84     }).map((_item) -> {
85         if(touche == KeyEvent.VK_DOWN) gérant.setBas(false);
86         return _item;
87     }).map((_item) -> {
88         if(touche == KeyEvent.VK_LEFT) gérant.setGauche(false);
89         return _item;
90     }).map((_item) -> {
91         if(touche == KeyEvent.VK_RIGHT) gérant.setDroite(false);
92         return _item;
93     }).map((_item) -> {
94         if(touche == KeyEvent.VK_B) gérant.setBfs(false);
95         return _item;
96     }).map((_item) -> {
97         if(touche == KeyEvent.VK_C) gérant.setDfs(false);
98         return _item;
99     }).map((_item) -> {
100        if(touche == KeyEvent.VK_SPACE) gérant.setBonus(false);
101        return _item;
102    }).filter((_item) -> (touche == KeyEvent.VK_E)).forEachOrdered((_item) -> {
103        gérant.setAstar(false);
104    });
105 }
106 }
107 }
108 }
```

La classe DepthFirstSearchEngine.java

```

1 package jeu;
2 import java.awt.Dimension;
3 import java.util.*;
4 public class DepthFirstSearchEngine extends AbstractSearchEngine {
5     // $S1
6     ArrayList<Dimension> I= new ArrayList<Dimension>(), T=new ArrayList<Dimension>();
7     Stack<Dimension> pile=new Stack<Dimension>();
8     Dimension predecessor[][] = new Dimension[maze.getWidth()][maze.getHeight()];
9     Dimension loc = new Dimension();
10
11     //la classe qui utilise l'algorithme DFS
12     public DepthFirstSearchEngine(int width, int height, Dimension Loc) {
13         super(width, height);
14
15         // $S2
16         for (int i=0; i<width; i++)
17             for (int j=0; j<height; j++)
18                 predecessor[i][j] = null;
19         this.loc = Loc;
20         pile.push(Loc);
21         System.out.println("in pile" + pile.size());
22         T.add(goalLoc);
23         System.out.println("search path" + searchPath.length);
24         // Algorithme de recherche en profondeur
25         doDFS();
26
27         // now calculate the shortest path from the predecessor array:
28         maxDepth = 0;
29         if (!isSearching) {
30             searchPath[maxDepth++] = goalLoc;
31             for (int i=0; i<2000; i++) {
32                 searchPath[maxDepth] = predecessor[searchPath[maxDepth - 1].width][searchPath[maxDepth - 1].height];
33                 maxDepth++;
34                 if (equals(searchPath[maxDepth - 1], Loc)) break; // back to starting node
35             }
36         }
37     }
38 }
39 }
```

```

40     private void doDFS() {
41
42         while(!pile.isEmpty()){
43             currentLoc = pile.pop();
44             L.add(currentLoc);
45
46             if(L.contains(currentLoc)){
47                 System.out.println("But trouvé : (" + currentLoc.width +", " + currentLoc.height+ ")");
48                 isSearching = false;
49                 break;
50             }
51             else{
52                 Dimension [] connected = getPossibleMoves(currentLoc);
53                 for(int i=0;i<connected.length;i++){
54
55                     if( connected[i]!=null && !pile.contains(connected[i]) && !L.contains(connected[i])){
56                         pile.push(connected[i]);
57                         predecessor[connected[i].width][connected[i].height] = currentLoc;
58                     }
59                 }
60             }
61         }
62     }
63 }
64
65 }
66
67 }
```

La classe Enemie.java

```

1 package jeu.classes;
2
3 import java.awt.Dimension;
4 import java.awt.Graphics;
5 import java.awt.Rectangle;
6 import java.awt.image.BufferedImage;
7 import java.io.File;
8 import java.io.IOException;
9 import javax.imageio.ImageIO;
10 //classe pour représenter ennemie de joueur
11 public class Enemie extends Entité {
12
13     protected final int LARGEURE = JeuDeLabyrinthe.tailleCaze;
14     protected final int HAUTEURE = JeuDeLabyrinthe.tailleCaze;
15     private BufferedImage image;
16     Dimension Loc = new Dimension();
17
18     public Enemie(int x, int y, IDEntité id) throws IOException {
19         super(x, y, id);
20         this.Loc.width = x;
21         this.Loc.height = y;
22         this.setLocE(Loc);
23         if(JeuDeLabyrinthe.tailleCaze == 12){
24             image = ImageIO.read(new File("files/momo/ennemi.png"));
25         }else{
26             image = ImageIO.read(new File("files/momo/ennemi24.png"));
27         }
28         // modification de l'ennemie dans le maze
29     @Override
30     protected void update() {
31         this.Loc.width = x;
32         this.Loc.height = y;
33         this.setLocE(Loc);
34     }
}
```

```

35     //dessin de l'ennemie
36     @Override
37     protected void render(Graphics g) {
38         g.drawImage(image, x, y, null);
39     }
40     // methode pour les collision avec les autres entités
41     @Override
42     protected Rectangle entourer() {
43         return new Rectangle(x,y,LARGEURE , HAUTEURE);
44     }
45     // getter et setter pour la location de l'ennemie
46     public Dimension getLoce(){
47         this.Loc.width = this.Loc.width /JeuDeLabyrinthe.tailleCaze;
48         this.Loc.height = this.Loc.height /JeuDeLabyrinthe.tailleCaze;
49         return this.Loc;
50     }
51     public void setLoce(Dimension loc){
52         this.Loc=loc;
53     }
54 }
55

```

La classe Entité.java

```

1 package jeu.classes;
2
3 import java.awt.Dimension;
4 import java.awt.Graphics;
5 import java.awt.Rectangle;
6 // Entités de jeu
7 public abstract class Entité {
8     protected int x, y; //position
9     protected int dx, dy; // déplacement
10    protected IDEntité id; // id de l'entité qui permet de différencier entre les entités
11    public Dimension Loc = new Dimension();
12    //constructor
13    public Entité(int x, int y, IDEntité id){
14        this.x = x;
15        this.y = y;
16        this.id= id;
17        dx = 0;
18        dy = 0;
19    }
20
21    //*****getter and setters *****/
22    public IDEntité getId() {
23        return id;
24    }
25    public void setId(IDEntité id) {
26        this.id = id;
27    }
28    public int getX() {
29        return x;
30    }
31    public void setX(int x) {
32        this.x = x;
33    }
34    public int getY() {
35        return y;
36    }
37    public void setY(int y) {
38        this.y = y;
39    }
40    public float getDx() {
41        return dx; }
42    public void setDx(int dx) {
43        this.dx = dx; }
44    public float getDy() {
45        return dy; }
46    public void setDy(int dy) {
47        this.dy = dy; }
48    public Dimension getLoc(){
49        this.Loc.width = this.Loc.width /JeuDeLabyrinthe.tailleCaze;
50        this.Loc.height = this.Loc.height /JeuDeLabyrinthe.tailleCaze;
51        return this.Loc; }
52    public void setLoc(Dimension loc){
53        this.Loc=loc; }
54    //modification de l'entité
55    protected abstract void update();
56    //dessin de l'entité
57    protected abstract void render(Graphics g);
58    //methode pour diffir les collisions
59    protected abstract Rectangle entourer();
60

```

La classe Fenetre.java

```
2 package jeu.classes;
3 import java.awt.BorderLayout;
4 import java.awt.Dimension;
5 import javax.swing.JFrame;
6 import javax.swing.JPanel;
7 import javax.swing.JRadioButton;
8 //Fenêtre principale
9 public class Fenetre extends JFrame{
10     public static JRadioButton BFS;
11     public static JRadioButton DFS;
12     public static JRadioButton AStar;
13     static JFrame frame = new JFrame("LSI MAZE GAME ");
14     public static int status = 0;
15     public Fenetre(String titre, JeuDeLabyrinthe jeu, int largeur, int hauteur){
16         JPanel panel = new JPanel();
17         panel.setPreferredSize(new Dimension(largeur, hauteur));
18         frame.setLayout(new BorderLayout());
19         frame.add(panel);
20         frame.setPreferredSize(new Dimension(largeur, hauteur));
21         frame.setMaximumSize(new Dimension(largeur, hauteur));
22         frame.setMinimumSize(new Dimension(largeur, hauteur));
23         frame.setSize(largeur, hauteur);
24         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         frame.setResizable(false);
26         frame.setLocationRelativeTo(null);
27         frame.setVisible(true);
28     }
29
30 }
31
```

La classe FenetreDepat.java

```
1 package jeuetu.classes;
2 import java.awt.BorderLayout;
3 import java.awt.Color;
4 import java.awt.Dimension;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7 import javax.swing.JFrame;
8 import javax.swing.JLabel;
9 import javax.swing.JPanel;
10 import javax.swing.Timer;
11
12 //Affichage de cette interface pendant 6 secondes
13 public class FenetreDepat extends JFrame {
14
15     private JLabel label;
16     private Timer timer;
17     private int count = 0;
18
19     public FenetreDepat(String titre, int largeur, int hauteur) {
20         System.out.println("Maze Game");
21
22         JFrame frame = new JFrame(titre);
23         frame.setLayout(new BorderLayout());
24
25         frame.setPreferredSize(new Dimension(largeur, hauteur));
26         frame.setMaximumSize(new Dimension(largeur, hauteur));
27         frame.setMinimumSize(new Dimension(largeur, hauteur));
28
29         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30         frame.setResizable(false);
31         frame.setLocationRelativeTo(null);
32     }
33 }
```

```

32         JPanel panel = new JPanel();
33         panel.setBackground(new Color(255, 105, 180));
34         panel.setBounds(0, 0, largeur, hauteur);
35         frame.getContentPane().add(panel);
36         frame.add(new Test.TestPane());
37
38     timer = new Timer(1000, new ActionListener() {
39         @Override
40         public void actionPerformed(ActionEvent e) {
41             count++;
42             if (count <= 6) {
43                 frame.setVisible(true);
44
45             } else {
46                 ((Timer) (e.getSource())).stop();
47                 frame.setVisible(false);
48
49             }
50         }
51     });
52     timer.setInitialDelay(0);
53     timer.start();
54
55 }
56
57 }
58

```

La classe GérantEntité.java

```

2 import java.awt.Graphics;
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.Random;
6 // le gérant des entités
7 public class GérantEntité {
8     Random rand = new Random(); // position aléatoire de bonus
9     private boolean haut, bas, gauche, droite, bfs, dfs, bonus, astar;
10    public static int max_nb_bonus = 0;
11    public static int max_nb_ennemis = 0;
12    private ArrayList<Entité> entités = new ArrayList(); // table des entités
13
14    private ArrayList<Entité> entites = new ArrayList(); // table des entités
15    ////////////////////getters and setters ///////////////////
16    public boolean isAstar() {
17        return astar;
18    }
19    public void setAstar(boolean astar) {
20        this.astar = astar;
21    }
22    public boolean isHaut() {
23        return haut;
24    }
25    public boolean isBonus() {
26        return bonus;
27    }
28    public void setBonus(boolean bonus) {
29        this.bonus = bonus;
30    }
31    public boolean isBfs() {
32        return bfs;
33    }
34    public void setBfs(boolean bfs) {
35        this.bfs = bfs;
36    }
37    public boolean isDfs() {
38        return dfs;
39    }
40    public void setDfs(boolean dfs) {
41        this.dfs = dfs;
42    }
43    public void setHaut(boolean haut) {
44        this.haut = haut;
45    }
46    public boolean isBas() {
47        return bas;
48    }
49    public void setBas(boolean bas) {
50        this.bas = bas;
51    }

```

19

LSI MAZE GAME

Faculté des sciences & techniques de Tanger – Département Génie Informatique

```

52 // modification des entités
53 public void update(){
54     for(int i=0;i<entités.size(); i++){
55         Entité entité = entités.get(i);
56         entité.update();
57     }
58 }
59 // dessin de l'entité
60 public void render(Graphics g) throws IOException{
61     // vérifier le nb de bonus
62     while(this.max_nb_bonus != max_bonus){
63         System.out.println("Hello in vérifie bonus");
64         int nombreAleatoire = rand.nextInt(42 + 1);
65         System.out.println(this.getEntités().size());
66         Entité entité = this.getEntités().get(nombreAleatoire);
67         if(this.getEntités().get(nombreAleatoire).getId() == IDENTité.CHEMIN){
68             this.max_nb_bonus++;
69             Bonus bn = new Bonus(this.getEntités().get(nombreAleatoire).getX(),this.getEntités().get(nombreAleatoire).getY(), IDENTité.BONUS);
70             this.ajouterEntité(bn);
71             System.out.println(" x et y " +this.getEntités().get(nombreAleatoire).getX() + " " +this.getEntités().get(nombreAleatoire).getY());
72         }
73     }
74     for(int i=0;i<entités.size(); i++){
75         Entité entité = entités.get(i);
76         entité.render(g);
77     }
78 }
79 // ajoute
80 public void ajouterEntité(Entité entité){
81     entités.add(entité);
82 }
83 // suppression
84 public void supprimerEntité(Entité entité){
85     entités.remove(entité);
86 }
87 }
88

```

La classe Hello.java

```

1 package jeu;
2 import java.awt.BorderLayout;
3 import java.awt.Color;
4 import java.awt.Dimension;
5 import java.awt.Font;
6 import java.awt.GridLayout;
7 import java.awt.event.ItemEvent;
8 import java.awt.event.ItemListener;
9 import javax.swing.BorderFactory;
10 import javax.swing.JButton;
11 import javax.swing.JFrame;
12 import javax.swing.JLabel;
13 import javax.swing.JPanel;
14 import javax.swing.JRadioButton;
15 import javax.swing.border.LineBorder;
16 import javax.swing.border.TitledBorder;
17 // fenêtre après la fenêtre de départ pour choisir le labyrinthe
18 public class Hello extends JFrame{
19     public static JRadioButton un;
20     public static JRadioButton deux;
21     public static JRadioButton trois;
22     public static JRadioButton quatre;
23     public static int status = 0;
24     public Hello(String titre, int largeur, int hauteur){
25         System.out.println("hello");
26         JFrame frame = new JFrame(titre);
27         frame.setLayout(new BorderLayout());
28         JButton btn= new JButton("btn");
29         JPanel panl = new JPanel();
30         panl.setPreferredSize(new Dimension( largeur,hauteur));
31         panl.setBackground(new Color(255,192,203));
32         panl.setLayout(null);
33         frame.getContentPane().add(panl, BorderLayout.CENTER);
34         JPanel zone = new JPanel(new GridLayout(3,2));
35         LineBorder linedBorder = new LineBorder(Color.black);
36         TitledBorder titledBorder = BorderFactory.createTitledBorder(linedBorder, "Maze size");
37         zone.setBorder(titledBorder);
38         zone.setBounds(200, 80, 400,140);
39         zone.setBackground(new Color(255,192,203));
40         panl.add(zone);
41

```

```

40     panel.add(zone);
41     //JButtonGroupe
42     ButtonGroup btngroupe= new ButtonGroup();
43     un = new JRadioButton();
44     un.setText("MAZE 21*21");
45     btngroupe.add(un);
46     zone.add(un);
47     deux= new JRadioButton();
48     deux.setText("MAZE 41*41");
49     btngroupe.add(deux);
50     zone.add(deux);
51
52     trois = new JRadioButton();
53     trois.setText("MAZE 61*61");
54     btngroupe.add(trois);
55     zone.add(trois);
56
57     quatre = new JRadioButton();
58     quatre.setText("MAZE 81*81");
59     btngroupe.add(quatre);
60     zone.add(quatre);
61     JLabel lb = new JLabel("During the game, please click the A key if you want help ;)");
62     lb.setFont(new Font("Serif", Font.BOLD, 19));
63     lb.setForeground(new Color(13,144,178));
64
65     lb.setBounds(200,200,500,300);
66     panel.add(lb);
67     frame.setPreferredSize(new Dimension(largeur, hauteur));
68     frame.setMaximumSize(new Dimension(largeur, hauteur));
69     frame.setMinimumSize(new Dimension(largeur, hauteur));
70     //frame.add(jeu);
71     frame.setDefaultCloseOperation(this.EXIT_ON_CLOSE);
72     frame.setResizable(false);
73     frame.setLocationRelativeTo(null);
74     frame.setVisible(true);

75     // listerner de choix de taille de labyrinthe
76     un.addItemListener(new ItemListener() {
77         public void itemStateChanged(ItemEvent event) {
78             if (event.getStateChange() == ItemEvent.SELECTED) {
79                 System.out.println("hello");
80                 JeuDeLabyrinthe.widthM = 21;
81                 JeuDeLabyrinthe.hieghM = 21;
82                 JeuDeLabyrinthe.verif = 1;
83                 JeuDeLabyrinthe.tailleCaze = 35;
84                 JeuDeLabyrinthe.secondeOver = 0;
85                 JeuDeLabyrinthe.minuteOver = 1;
86                 GérantEntité.max_nb_bonus=4;
87                 GérantEntité.max_bonus=4;
88                 GérantEntité.max_nb_ennemis =4;
89                 //JeuDeTire jeu = new JeuDeLabyrinthe();
90                 //jeu.boucleDuJeu();
91                 frame.setVisible(false);
92             }
93         }
94     );
95     deux.addItemListener(new ItemListener() {
96         public void itemStateChanged(ItemEvent event) {
97             if (event.getStateChange() == ItemEvent.SELECTED) {
98                 System.out.println("hello");
99                 JeuDeLabyrinthe.widthM = 41;
100                JeuDeLabyrinthe.hieghM = 41;
101                JeuDeLabyrinthe.verif = 1;
102                JeuDeLabyrinthe.tailleCaze = 23;
103                JeuDeLabyrinthe.minuteOver = 1;
104                JeuDeLabyrinthe.secondeOver =30;
105                GérantEntité.max_nb_bonus=5;
106                GérantEntité.max_bonus=5;
107                GérantEntité.max_nb_ennemis =8;
108
109                frame.setVisible(false);
110            }
111        }
112    );
}

```

```

115     trois.addItemListener(new ItemListener() {
116         public void itemStateChanged(ItemEvent event) {
117             if (event.getStateChange() == ItemEvent.SELECTED) {
118                 System.out.println("hello");
119                 JeuDeLabyrinthe.widthM = 61;
120                 JeuDeLabyrinthe.hieghtM = 61;
121                 JeuDeLabyrinthe.verif = 1;
122                 JeuDeLabyrinthe.tailleCaze = 16;
123                 GérantEntité.max_nb_ennemis = 12;
124                 GérantEntité.max_nb_bonus = 9;
125                 GérantEntité.max_bonus = 9;
126                 JeuDeLabyrinthe.minuteOver = 1;
127                 JeuDeLabyrinthe.secondeOver = 50;
128
129             }
130         }
131     });
132     quatre.addItemListener(new ItemListener() {
133         public void itemStateChanged(ItemEvent event) {
134             if (event.getStateChange() == ItemEvent.SELECTED) {
135                 System.out.println("hello");
136                 JeuDeLabyrinthe.widthM = 81;
137                 JeuDeLabyrinthe.hieghtM = 81;
138                 JeuDeLabyrinthe.verif = 1;
139                 JeuDeLabyrinthe.tailleCaze = 12;
140                 GérantEntité.max_nb_ennemis = 16;
141                 GérantEntité.max_nb_bonus = 13;
142                 GérantEntité.max_bonus = 13;
143                 JeuDeLabyrinthe.minuteOver = 3;
144                 JeuDeLabyrinthe.secondeOver = 0;
145
146             }
147         }
148     });
149 }
150 public int getStatus() {
151     System.out.println("hello in fenaitrestatus" +this.status);
152     return this.status;
153 }

```

La classe IDEntité.java

```

6     package jeu.classes;
7
8
9     // id pour faire la différence entre les entités
10    public enum IDEntité {
11        JOUEUR(),
12        ENEMIE(),
13        OBSTACLE(),
14        CHEMIN(),
15        SORTIE(),
16        TRAJECTOIRE(),
17        BONUS(),
18        SCORE();
19    }
20

```

La classe JeuDeTire.java

```
1 package jeu.classes;
2 import java.awt.Canvas;
3 import java.awt.Color;
4 import java.awt.Dimension;
5 import java.awt.Graphics;
6 import java.awt.Graphics2D;
7 import java.awt.image.BufferStrategy;
8 import java.io.IOException;
9 import java.util.ArrayList;
10 import java.util.Random;
11 import java.util.Timer;
12 import java.util.TimerTask;
13 import javax.swing.JFrame;
14 import javax.swing.JOptionPane;
15 // class pour lancer le jeu
16 public class JeuDeLabyrinthe extends Canvas {
17     public static int widthM = 0; //width de maze
18     public static int heightM = 0; // height de maze
19     AStarSearchEngine cSearchEngineASTAR= null;
20     BreadthFirstSearchEngine currentSearchEngineBFS = null;
21     DepthFirstSearchEngine currentSearchEngineDFS = null;
22     public static final int LARGEUR = 1400; // largeur de fenêtre
23     public static final int HAUTEUR=2000; // hauteur de fenêtre
24     // entités de jeu
25     GérantEntité gérant;
26     Joueur joueur ;
27     Score score ;
28     Bonus bn ;
29     Sortie sortie;
30     Trajectoire trajectoire;
31     public static int val =0;
32     // arraylist des entités
33     public static ArrayList<Bonus> bonus= new ArrayList<>();
34     ArrayList<Chemin> chemins = new ArrayList<>();
35     ArrayList<Obstacle> obstacles = new ArrayList<>();
36     ArrayList<Trajectoire> trajectoires = new ArrayList<>();
37     ArrayList<Dimension> locEnemie = new ArrayList<>();
38     ArrayList<Enemie> enemies= new ArrayList<>();
39
40     // taille de la case de labyrinthe
41     public static int tailleCase = 16 ;
42     int k = 1;
43     public static int verif = 0; //pour vérification
44     public Random randomGenerator; // pour les nombres aléatoire de bonus
45     Random rand = new Random();
46     public Dimension loc; // variable pour les location
47     public static int s=0; // pour le temps de visualisation
48     public static int n =0;
49     // variable d'initialisation pour le temps de jeu
50     public static int minuteJeu =0;
51     public static int minuteOver = 0;
52     public static int secondeOver = 0;
53     //maze
54     Maze maze =null;
55     // constructeur
56     public JeuDeLabyrinthe() throws IOException{
57         Fenetre fenetre = new Fenetre("LSI MAZE GAME ",this, LARGEUR, HAUTEUR);
58         loc = new Dimension();
59         gérant = new GérantEntité();
60         System.out.println("selected"+heightM + widthM);
61         // maze
62         drawMaze();
63         //ajouter Bonus
64         ajouterBonus();
65         //ajouter Enemis
66         ajouterEnemies();
67     }
68     this.addKeyListener(new Clavier(gérant));
69     public static void main(String[] args) throws IOException {
70         FenetreDepat fen = new FenetreDepat("LSI MAZE GAME ", 700, 600);
71         Hello hell = new Hello("LSI MAZE GAME ", 700, 600);
72         System.out.println("selected"+heightM + widthM);
73         do{
74             System.out.println("selected"+heightM + widthM);
75             if(widthM != 0 && heightM != 0) {
76                 JeuDeLabyrinthe jeu = new JeuDeLabyrinthe();
77                 Chronometre chronometre = new Chronometre();
78                 chronometre.setVisible(true);
79                 jeu.boucleDuJeu();
80             }while(verif == 0);
81     }
82 }
```

```

80 //*****draw maze*****
81 private void drawMaze() throws IOException{
82     this.maze = new Maze(widthM,heightM);
83     maze.setValue(0, 0, Maze.START_LOC_VALUE);
84     for (int x=0; x<maze.getHeight(); x++) {
85         for (int y=0; y<maze.getWidth(); y++) {
86             short val = maze.getValue(x,y);
87             if(val == Maze.START_LOC_VALUE) {
88                 joueur = new Joueur(0,0, IDEntité.JOUEUR, gérant);
89             if(val == Maze.GOAL_LOC_VALUE) {
90                 sortie =new Sortie(x*tailleCaze,y*tailleCaze, IDEntité.SORTIE);
91             else if(val == Maze.OBSTACLE){
92                 obstacles.add( new Obstacle(x*tailleCaze,y*tailleCaze,IDEntité.OBSTACLE));
93             }else{
94                 chemins.add( new Chemin(x*tailleCaze,y*tailleCaze,IDEntité.CHEMIN));}}
95         //fin mize
96         for(Chemin chemin:chemins){
97             gérant.ajouterEntité(chemin);
98             for(Obstacle obstacle:obstacles){
99                 gérant.ajouterEntité(obstacle);}
100             gérant.ajouterEntité(joueur);
101             gérant.ajouterEntité(sortie);
102     //*****boucle de jeu*****
103     public void boucleDuJeu() throws IOException{
104         boolean running = true;
105         long lastTime = System.nanoTime();
106         double amountOfTicks = 60.0;
107         double ns = 1000000000 / amountOfTicks;
108         double delta = 0;
109         long timer = System.currentTimeMillis();
110         int updates = 0;
111         int frames = 0;
112         while(running){
113             long now = System.nanoTime();
114             delta +=(now -lastTime) / ns;
115             lastTime = now;
116             while(delta >= 1){
117                 update();
118                 updates++;
119                 delta--;}
120             render();
121             frames++;
122             if(System.currentTimeMillis() - timer >1000 ){
123                 timer += 1000;
124                 System.out.println("FPS" + frames +"TICKS" +updates);
125                 frames = 0;
126                 updates = 0;}}
127     //modification au cours de jeu
128     private void update(){
129         if(minuteJeu == 1){
130             gérant.supprimerEntité(joueur);
131             JFrame frame = new JFrame();
132             minuteJeu = 0;
133             JOptionPane.showMessageDialog(frame,"FIN DE JEU");
134         }
135         gérant.update();
136         if( gérant.isBfs()){
137             trajetBFS();}
138         if( gérant.isDfs()){
139             trajetDFS();}
140         if(gérant.isBonus()){
141             System.out.println("selectedbonus *****");
142             int nombreAleatoire;
143             Entité entité ;
144             do{
145                 nombreAleatoire = rand.nextInt(gérant.getEntités().size()-1 + 1);
146                 // System.out.println(gérant.getEntités().size());
147                 entité = gérant.getEntités().get(nombreAleatoire);
148                 while(gérant.getEntités().get(nombreAleatoire).getId() != IDEntité.BONUS);
149                 Dimension lc = new Dimension();
150                 lc.width=entité.getX()/tailleCaze;
151                 lc.height=entité.getY()/tailleCaze;
152                 trajetAstarBonus(lc);
153                 gérant.setBonus(false);}
154             if( gérant.isAstar()){
155                 trajetASTAR();}
156     // dessin de le maze
157     private void render() throws IOException{
158         BufferStrategy buffer = this.getBufferStrategy();
159         if(buffer == null){
160             this.createBufferStrategy(3);

```

```

161         return ;
162     Graphics g = (Graphics2D)buffer.getDrawGraphics();
163     //Dessiner ici
164     g.setColor(Color.red);
165     g.fillRect(0,0,width*tailleCase, height*tailleCase);
166     // case blanc pour écrire le score
167     gérant.ajouterEntité(new Chemin(1340,200,IDENTité.CHEMIN));
168     gérant.ajouterEntité(new Chemin(1353,200,IDENTité.CHEMIN));
169     gérant.ajouterEntité(new Chemin(1366,200,IDENTité.CHEMIN));
170     //écrire le score
171     score = new Score(1340, 215, IDENTité.SCORE);
172     gérant.ajouterEntité(score);
173     gérant.render(g);
174     //fin de dessin
175     g.dispose();
176     buffer.show();
177     /*ajouter bonus***** */
178     private void ajouterBonus() throws IOException{
179         while(x <= gérant.max_nb_bonus){
180             System.out.println("max bonus"+ gérant.max_nb_bonus );
181             int nombreAleatoire = rand.nextInt(gérant.getEntités().size()-1 + 1);
182             System.out.println("size entités"+ gérant.getEntités().size());
183             Entité entité = gérant.getEntités().get(nombreAleatoire);
184             if(gérant.getEntités().get(nombreAleatoire).getId() == IDENTité.CHEMIN){
185                 k++;
186                 bonus.add( new Bonus(gérant.getEntités().get(nombreAleatoire).getX(),gérant.getEntités().get(nombreAleatoire).getY(), IDENTité.BONUS));
187             }
188             //System.out.println(" x et y " +gérant.getEntités().get(nombreAleatoire).getX() + " " +gérant.getEntités().get(nombreAleatoire).getY());
189         }
190         for(Bonus bns:bonus){
191             gérant.ajouterEntité(bns);
192         }
193     }/*ajouter des ennemis***** */
194     private void ajouterEnemies() throws IOException{
195         /****** path***** */
196         currentSearchEngineDFS = new DepthFirstSearchEngine(maze.getWidth(),maze.getHeight(),joueur.getLoc());
197         if (currentSearchEngineDFS == null) return;
198         //affichage des enemies
199         Dimension [] path = currentSearchEngineDFS.getPath();
200
201         for (int i=1; i< (path.length-1); i++) {
202             int x = path[i].width;
203             int y = path[i].height;
204             locEnnemie.add(path[i]);
205             System.out.println("loc de ennemie" +locEnnemie.size());
206             for(int n = 0; n<GénérantEntité.max_nb_enemies;n++){
207                 int nb = rand.nextInt(locEnnemie.size()-20 -1 -1)+1;
208                 System.out.println("alatoire" +nb);
209                 enemies.add( new Enemie(locEnnemie.get(nb).width*tailleCase ,locEnnemie.get(nb).height*tailleCase, IDENTité.ENEMIE));
210                 System.out.println("Enemie in "+locEnnemie.get(nb).width*tailleCase +" fkfkfk"+ locEnnemie.get(nb).height*tailleCase);
211                 for(Enemie enn:enemies){
212                     gérant.ajouterEntité(enn);
213                 }
214             }
215         }
216     }/*visualisation***** */
217     /*avec bfs***** */
218     public void trajetBFS(){
219         System.out.println("in trajet bfs" );
220         System.out.println("in trajet bfs" );
221         currentSearchEngineBFS = new BreadthFirstSearchEngine(maze.getWidth(),maze.getWidth(),joueur.getLoc(),maze.goalLoc);
222         if (currentSearchEngineBFS == null) return;
223         Timer t = new Timer();
224         TimerTask tt = new TimerTask() {
225             @Override
226             public void run() {
227                 s++;
228                 System.out.println("le temps passe est :" + s);
229                 //do something
230                 if(s<18) {
231                     Dimension [] path = currentSearchEngineBFS.getPath();
232                     for (int i=1; i< (path.length-1); i++) {
233                         int x = path[i].width;
234                         int y = path[i].height;
235                         short val = maze.getValue(x,y);
236                         trajectoires.add( new Trajectoire(">", x, y, IDENTité.TRAJECTOIRE, gérant));
237                         for(Trajectoire trajectoire: trajectoires){
238                             gérant.ajouterEntité(trajectoire);
239                             trajectoire.clear();
240                         else if(s>8){
241                             // suppression de trajet

```

```

241     // suppression de trajet
242     Dimension [] pathI = currentSearchEngineBFS.getPath();
243     for (int i=1; i< (pathI.length-1); i++) {
244         int x = pathI[i].width;
245         int y = pathI[i].height;
246         short val = maze.getValue(x,y);
247         for(Trajectoire trajectoire:trajectoires){
248             gérant.supprimerEntité(trajectoire);
249             for(int p =0; p<gérant.getEntités().size(); p++){
250                 Entité entité = gérant.getEntités().get(p);
251                 if(entité.getId() == IDEntité.TRAJECTOIRE){
252                     gérant.supprimerEntité( entité);}}}
253             trajectoires.clear();
254             System.out.println("trajet"+trajectoires);
255             //trajectoires.clear();
256             trajectoires.clear();
257             t.cancel();}}};;
258 t.schedule(tt,0,1000);
259     System.out.println("fin task");
260     /****** avec dfs *****/
261     public void trajetDFS(){
262         Dimension lc = new Dimension();
263         lc.width=joueur.getX()/tailleCase;
264         lc.height=joueur.getY()/tailleCase;
265         currentSearchEngineDFS = new DepthFirstSearchEngine(maze.getWidth(), maze.getHeight(),lc);
266         if (currentSearchEngineDFS == null) return;
267         Timer t = new Timer();
268         TimerTask tt = new TimerTask() {
269             @Override
270             public void run() {
271                 s++;
272                 System.out.println("le temps passe est :" + s);
273                 //do something
274                 if(s<18) {
275                     Dimension [] path = currentSearchEngineDFS.getPath();
276                     for (int i=1; i< (path.length-1); i++) {
277                         int x = path[i].width;
278                         int y = path[i].height;
279                         short val = maze.getValue(x,y);
280                         System.out.println("path dfs"+ ( path.length-1));
281
282                         trajectoires.add (new Trajectoire(">", x, y, IDEntité.TRAJECTOIRE,gérant));}
283                     for(Trajectoire trajectoire:trajectoires){
284                         gérant.ajouterEntité(trajectoire);
285                     trajectoires.clear();}}
286                 else if(s>8){
287                     // suppression de trajet
288                     Dimension [] pathI = currentSearchEngineBFS.getPath();
289                     for (int i=1; i< (pathI.length-1); i++) {
290                         int x = pathI[i].width;
291                         int y = pathI[i].height;
292                         short val = maze.getValue(x,y);
293                         for(Trajectoire trajectoire:trajectoires){
294                             gérant.supprimerEntité(trajectoire);
295                             for(int p =0; p<gérant.getEntités().size(); p++){
296                                 Entité entité = gérant.getEntités().get(p);
297                                 if(entité.getId() == IDEntité.TRAJECTOIRE){
298                                     gérant.supprimerEntité( entité);}}}
299                     trajectoires.clear();
300                     System.out.println("trajet"+trajectoires);
301                     //trajectoires.clear();
302                     trajectoires.clear();
303                     t.cancel();}}};;
304 t.schedule(tt,0,1000);
305     System.out.println("fin task");
306     /****** avec astar *****/
307     public void trajetASTAR(){
308         System.out.println("in trajet bfs");
309         currentSearchEngineBFS = new BreadthFirstSearchEngine(maze.getWidth(),maze.getWidth(),joueur.getLocJ(),maze.goalLoc);
310         if (currentSearchEngineBFS == null) return;
311         Timer t = new Timer();
312         TimerTask tt = new TimerTask() {
313             @Override
314             public void run() {
315                 s++;
316                 System.out.println("le temps passe est :" + s);
317                 //do something
318                 if(s<18) {
319                     Dimension [] path = currentSearchEngineBFS.getPath();
320                     for (int i=1; i< (path.length-1); i++) {
321                         int x = path[i].width;
322                         int y = path[i].height;

```

```

322     short val = maze.getValue(x,y);
323     trajectoires.add (new Trajectoire(">", x, y, IDEntité.TRAJECTOIRE, gérant));}
324     for(Trajectoire trajectoire:trajectoires){
325         gérant.ajouterEntité(trajectoire);
326         trajectoires.clear();
327     else if(s>8){
328         // suppression de trajet
329         Dimension [] pathl = currentSearchEngineBFS.getPath();
330         for (int i=1; i< (pathl.length-1); i++) {
331             int x = pathl[i].width;
332             int y = pathl[i].height;
333             short val = maze.getValue(x,y);
334             for(Trajectoire trajectoire:trajectoires){
335                 gérant.supprimerEntité(trajectoire);
336                 for(int p =0; p<gérant.getEntités().size(); p++){
337                     Entité entité = gérant.getEntités().get(p);
338                     if(entité.getId() == IDEntité.TRAJECTOIRE){
339                         gérant.supprimerEntité( entité);}}

```

La classe Joueur.java

```
1 package jeu.classes;
2 import java.awt.Dimension;
3 import java.awt.Graphics;
4 import java.awt.Rectangle;
5 import java.awt.image.BufferedImage;
6 import java.io.File;
7 import java.io.IOException;
8 import java.util.Stack;
9 import javax.imageio.ImageIO;
10 public class Joueur extends Entité {
11     GérantEntité gérant;
12     private BufferedImage image;
13     private final int LARGEURJ = 10;
14     private final int HAUTEURJ = 10;
15     public static int SCORE = 0;
16     public static int f = 0;
17     //pile pour stocké la dernière position
18     Stack<Dimension> list=new Stack<Dimension>();
19     public Dimension list = new Dimension();
20     Dimension retour = new Dimension();
21     Maze maze = new Maze(JeuDeLabyrinthe.widthM, JeuDeLabyrinthe.heightM);
22     boolean bool = false;
23     public Joueur(int x, int y, Identité id, GérantEntité gérant) throws IOException {
24         super(x, y, id);
25         this.gérant = gérant;
26         if(JeuDeLabyrinthe.tailleCaze ==12){
27             image = ImageIO.read(new File("files/momo/thinking2.png"));
28         }else if (JeuDeLabyrinthe.tailleCaze ==35){
29             image = ImageIO.read(new File("files/momo/thinking24.png"));
30         }else if (JeuDeLabyrinthe.tailleCaze ==38){
31             image = ImageIO.read(new File("files/momo/thinking24.png"));
32         }else{
33             image = ImageIO.read(new File("files/momo/thinking.png"));
34         }
35     }
36     @Override
37     protected void update() {
38         dx=0;
39         dy=0;
40         this.Loc.width = x;
41         this.Loc.height = y;
42         this.setLocJ(Loc);
43         list.add(Loc);
44         // System.out.println("xet y joueur " + x/25+"/"+ " "+y/25);
45         if(gérant.isHaut()) {
46             dy =-1;
47             bool= true;
48         }
49         if(gérant.isBas()) {
50             dy =1;
51             bool= true;
52         }
53         if(gérant.isGauche()) {
54             dx =-1;
55             bool= true;
56         }
57         if(gérant.isDroite()) {
58             dx =1;
59             bool= true;
60         }
61         if(x<0) x=0;
62         if(y<0) y=0;
63         if(x>maze.getWidth()*JeuDeLabyrinthe.tailleCaze ) x=maze.getWidth()*JeuDeLabyrinthe.tailleCaze ;
64         if(y>maze.getHeight()*JeuDeLabyrinthe.tailleCaze) y=maze.getHeight()*JeuDeLabyrinthe.tailleCaze;
65         x+=dx;
66         y+=dy;
67         collision();
68         // collisionEnemie();
69     }
70     @Override
71     protected void render(Graphics g) {
72         g.drawImage(image, x, y, null);
73     }
74     @Override
75     protected Rectangle entourer() {
76         return new Rectangle(x,y,LARGEURJ,HAUTEURJ);
77     }
78 }
```

```

77     private void collision() {
78         for(int i = 0; i<gérant.getEntités().size(); i++){
79             Entité entité = gérant.getEntités().get(i);
80             if(entité.getId() == IDEntité.OBSTACLE){
81                 if(entourer().intersects(entité.entourer())){
82                     retour = [REDACTED].pop();
83                     x=retour.width;
84                     y=retour.height;
85                 }
86             }
87             if(entité.getId() == IDEntité.BONUS){
88                 if(entourer().intersects(entité.entourer())){
89                     gérant.supprimerEntité(entité);
90                     SCORE = SCORE + 5;
91                     --gérant.max_nb_bonus;
92                     System.out.println("score = "+SCORE);
93                 }
94             }
95             if(entité.getId() == IDEntité.ENEMIE){
96                 if(entourer().intersects(entité.entourer())){
97                     if(SCORE > 1 ){
98                         gérant.supprimerEntité(entité);
99                         SCORE = SCORE - 2;
100                        System.out.println("score = in ennemie "+SCORE);
101                    }
102                    if(SCORE <2){
103                        retour = [REDACTED].pop();
104                        x=retour.width;
105                        y=retour.height;
106                        System.out.println("joueur in "+retour.width+" "+retour.height);
107                    }
108                }
109            }
110            if(entité.getId() == IDEntité.SORTIE){
111                if(entourer().intersects(entité.entourer())){
112                    gérant.supprimerEntité(entité);
113                    JeuDeLabyrinthe.minuteJeu = 1;
114                }
115            }
116        }
117    }

118     public Dimension getLocJ(){
119         this.Loc.width = this.Loc.width /JeuDeLabyrinthe.tailleCaze;
120         this.Loc.height = this.Loc.height /JeuDeLabyrinthe.tailleCaze;
121         return this.Loc;
122     }
123     public void setLocJ(Dimension loc){
124         this.Loc=loc;
125     }
126 }
127 }
128

```

La classe Maze.java

```
1 package jeu.classes;
2 import java.awt.Dimension;
3 import java.io.IOException;
4 import java.nio.file.Files;
5 import java.nio.file.Path;
6 import java.nio.file.Paths;
7 import java.util.logging.Level;
8 import java.util.logging.Logger;
9
10 //Class Maze - private class for representing search space as a two-dimensional maze
11 public class Maze {
12     public static short OBSTACLE = -1;
13
14     public static short START_LOC_VALUE = -2;
15     public static short GOAL_LOC_VALUE = -3;
16     private int width = 0;
17     private int height = 0;
18     public final Dimension startLoc = new Dimension();
19     public final Dimension goalLoc = new Dimension();
20     Path path;
21
22     /**
23      * The maze (or search space) data is stored as a short integer rather than
24      * as a boolean so that breadth-first style searches can use the array to store
25      * search depth. A value of -1 indicates a barrier in the maze.
26     */
27     public final short[][][] maze;
28     public Maze(int width, int height) {
29         System.out.println("New maze of size " + width + " by " + height);
30         this.width = width;
31         this.height = height;
32         maze = new short[width+2][height+2];
33         for (int i=0; i<width+2; i++) {
34             for (int j=0; j<height+2; j++) {
35                 maze[i][j] = 0;
36             }
37
38             //Randomize the maze by putting up arbitrary obstacles
39             for (int i=0; i<height+2; i++) {
40                 maze[0][i] = maze[width+1][i] = OBSTACLE;
41             }
42             for (int i=0; i<width+2; i++) {
43                 maze[i][0] = maze[i][height+1] = OBSTACLE;
44             }
45             //41 41
46             if(JeuDeLabyrinthe.widthM == 41 && JeuDeLabyrinthe.widthM == 41 ){
47                 path = (Path) Paths.get("files/momo/LABY_41x41.txt");
48             }
49             //61 61
50             if(JeuDeLabyrinthe.widthM == 61 && JeuDeLabyrinthe.widthM == 61 ){
51                 path = (Path) Paths.get("files/momo/LABY_61x61.txt");
52             }
53             //81 81
54             if(JeuDeLabyrinthe.widthM == 81 && JeuDeLabyrinthe.widthM == 81 ){
55                 path = (Path) Paths.get("files/momo/LABY_81x81.txt");
56             }
57             // 21/21
58             if(JeuDeLabyrinthe.widthM == 21 && JeuDeLabyrinthe.widthM == 21 ){
59                 path = (Path) Paths.get("files/momo/LABY_21x21.txt");
60             }
61             java.util.List <String>lignes = null;
62
63             try {
64                 lignes = Files.readAllLines(path);
65             } catch (IOException ex) {
66                 Logger.getLogger(Maze.class.getName()).log(Level.SEVERE, null, ex);
67             }
68         }
69     }
70 }
```

```

68     int x=0;
69     for(String ligne: lignes ){
70         //System.out.println(ligne.length());
71         if(ligne.length() == width -1){
72             //System.out.println("hello");
73             for(int y = 0; y<width -1 ;y++){
74                 char l= ligne.charAt(y);
75                 if(l!=' '){
76                     maze[x+1][y+1] = OBSTACLE;
77                     // System.out.println("hello" +OBSTACLE);
78                 }
79                 //System.out.println("le char" + l);
80             }else{
81                 for(int y = 0; y<width;y++){
82                     char l= ligne.charAt(y);
83                     if(l!=' '){
84                         maze[x+1][y+1] = OBSTACLE;
85                         //System.out.println("hello" +OBSTACLE);
86                     }
87                     x++;
88                 if(x ==width)break;
89             }
90             * Specify the starting location
91             */
92             startLoc.width = 0;
93             startLoc.height = 0;
94             maze[1][1] = START_LOC_VALUE;
95             */
96             * Specify the goal location
97             */
98             goalLoc.width = width - 1;
99             goalLoc.height =height - 1;
100            maze[width][height] = GOAL_LOC_VALUE;
101        }
102    }
103    synchronized public short getValue(int x, int y) { return maze[x+1][y+1]; }
104    synchronized public void setValue(int x, int y, short value) { maze[x+1][y+1] = value; }
105    public int getWidth() { return width; }
106    public int getHeight() { return height; }
107}

```

La classe Obstacle.java

```

2 package jeu.classes;
3 import java.awt.Color;
4 import java.awt.Graphics;
5 import java.awt.Rectangle;
6
7 //représenter les obstacles céd les murs de maze
8 public class Obstacle extends Entité {
9
10     protected final int LARGEURE = JeuDeLabyrinthe.tailleCase;
11     protected final int HAUTEURE = JeuDeLabyrinthe.tailleCase;
12     public Obstacle(int x, int y, IDEntité id) {
13         super(x, y, id);
14     }
15     @Override
16     protected void update() {
17     }
18
19     @Override
20     protected void render(Graphics g) {
21
22         g.setColor(new Color(13,144,178));
23         g.fillRect( x , y ,LARGEURE , HAUTEURE );
24     }
25     @Override
26     protected Rectangle entourer() {
27         return new Rectangle(x,y,LARGEURE ,HAUTEURE );
28     }
29 }

```

La classe Score.java

31

```

1 package jeu.classes;
2 import java.awt.Color;
3 import java.awt.Font;
4 import java.awt.Graphics;
5 import java.awt.Rectangle;
6
7 //représente le score de joueur
8 public class Score extends Entité {
9     GérantEntité gérant;
10    protected final int LARGEURE = 1300;
11    protected final int HAUTEURE = 240;
12    public Score(int x, int y, IDEntité id) {
13        super(x, y, id);
14    }
15
16    @Override
17    protected void update() {
18    }
19    @Override
20    protected void render(Graphics g) {
21        g.setFont(new Font("TimesRoman", Font.PLAIN, 22));
22        g.setColor(Color.black);
23        g.drawString("SCORE" , x-40, y-30);
24        g.setColor(Color.black);
25        g.drawString(" " +Joueur.SCORE , x, y);
26    }
27
28    @Override
29    protected Rectangle entourer() {
30        return new Rectangle(x,y,LARGEURE ,HAUTEURE);
31    }
32
33 }
34

```

La Sortie.java

```

1 package jeu.classes;
2 import java.awt.Graphics;
3 import java.awt.Rectangle;
4 import java.awt.image.BufferedImage;
5 import java.io.File;
6 import java.io.IOException;
7 import javax.imageio.ImageIO;
8
9 // la sortie de maze
10 public class Sortie extends Entité {
11     private BufferedImage image;
12     protected final int LARGEURE = 25;
13     protected final int HAUTEURE = 25;
14     public Sortie(int x, int y, IDEntité id) throws IOException {
15         super(x, y, id);
16         if(JeuDeLabyrinthe.tailleCaze == 12){
17             image = ImageIO.read(new File("files/momo/finish-flag.png"));
18         }else {
19             image = ImageIO.read(new File("files/momo/finish-flag24.png"));
20         }
21     }
22     @Override
23     protected void update() {
24     }
25
26     @Override
27     protected void render(Graphics g) {
28
29         g.drawImage(image, x, y, null);
30
31     }
32     @Override
33     protected Rectangle entourer() {
34         return new Rectangle(x,y,LARGEURE ,HAUTEURE);
35     }
36
37 }
38

```

La classe Test.java

```

package jeuetuile.classes;
import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
//Afficher l'incrémentation du conteur pendant 5 secondes
public class Test {
    private static int c = 0;
    public static class TestPane extends JPanel {
        private JLabel label;
        private Timer timer;
        private int count = 0;
        public TestPane() {
            label = new JLabel("...");
            setLayout(new GridBagLayout());
            label.setForeground(Color.cyan);
            add(label);
            timer = new Timer(1000, new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    count++;
                    if (count <= 5) {
                        label.setFont(new Font(Integer.toString(count), Font.ITALIC, 300));
                        label.setText(Integer.toString(count));
                    } else {
                        ((Timer) (e.getSource())).stop();
                        label.setFont(new Font("GOOOOO !!!", Font.ITALIC, 100));
                        label.setText("GOOOOO !!!");
                    }
                }
            });
            timer.setInitialDelay(0);
            timer.start();
        }
    }
}

```

La classe Trajectoire.java

```

2 package jeu.classes;
3
4 import java.awt.Color;
5 import java.awt.Dimension;
6 import java.awt.Font;
7 import java.awt.Graphics;
8 import java.awt.Rectangle;
9 // dissiner des fléches lors de demande de l'aide
10 public class Trajectoire extends Entité {
11
12     GérantEntité gérant;
13     protected final int LARGEURE = 6;
14     protected final int HAUTEURE = 6;
15     public Dimension path;
16     public static String i;
17     public Trajectoire(String i,int x, int y, IDENTité id, GérantEntité gérant) {
18         super(x, y, id);
19         this.i = i;
20         this.gérant = gérant;
21     }
22
23     @Override
24     protected void update() {
25     }
26     @Override
27     protected void render(Graphics g) {
28         g.setFont(new Font("TimesRoman", Font.PLAIN, 17));
29         g.setColor(Color.red);
30
31         g.drawString(""+i, 3 + x * JeuDeLabyrinthe.tailleCase,10+ y * JeuDeLabyrinthe.tailleCase);
32
33     }
34     @Override
35     protected Rectangle entourer() {
36         return new Rectangle(x,LARGEURE ,HAUTEURE);
37     }
38
39 }
40

```

Interfaces

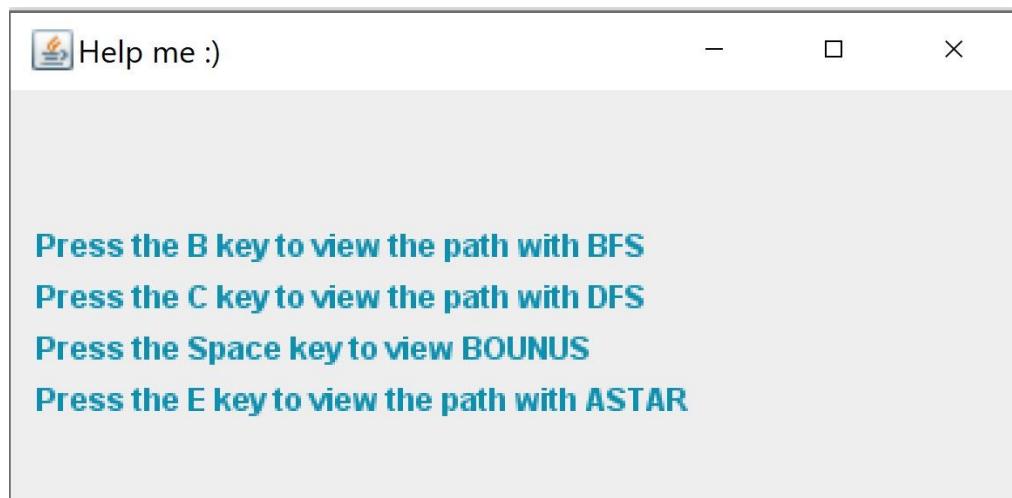
Lancement du jeu



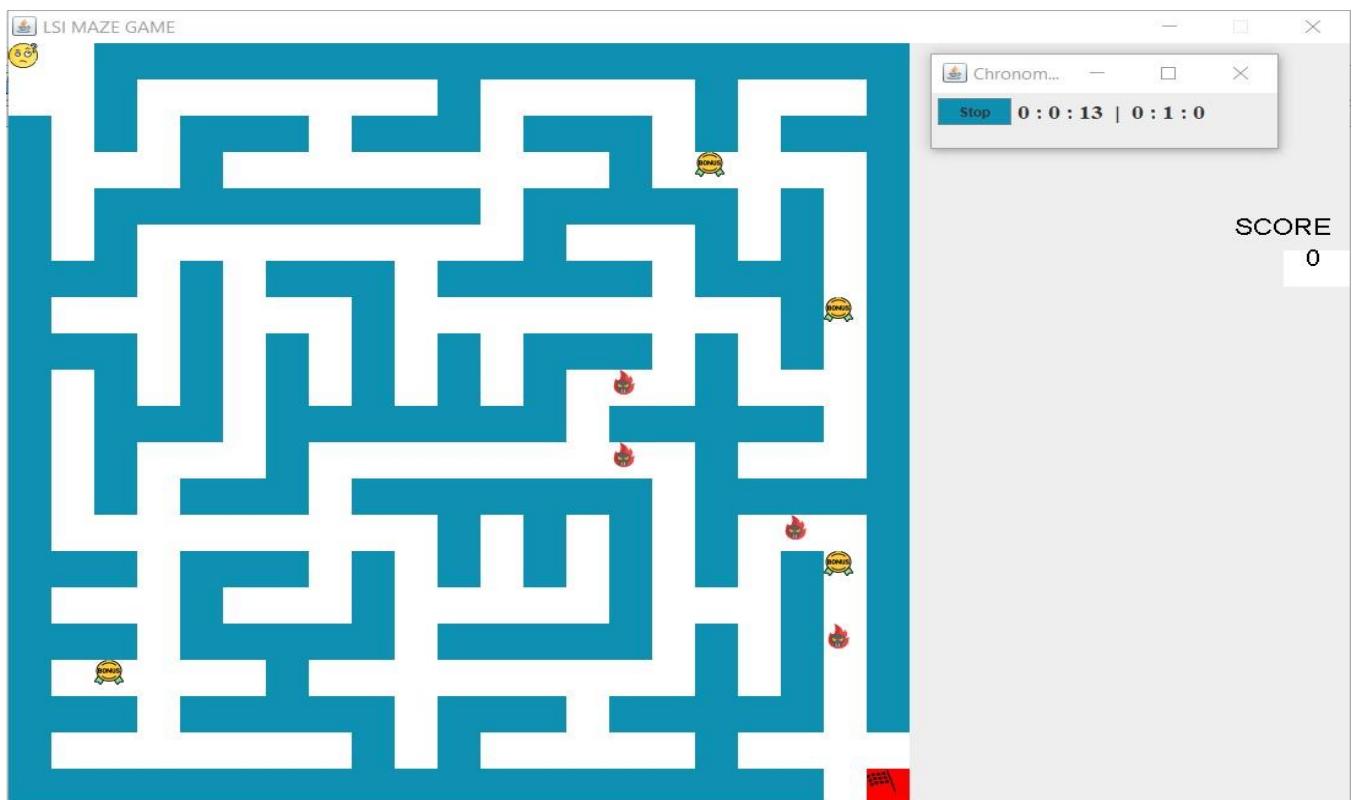
Choix de labyrinthe



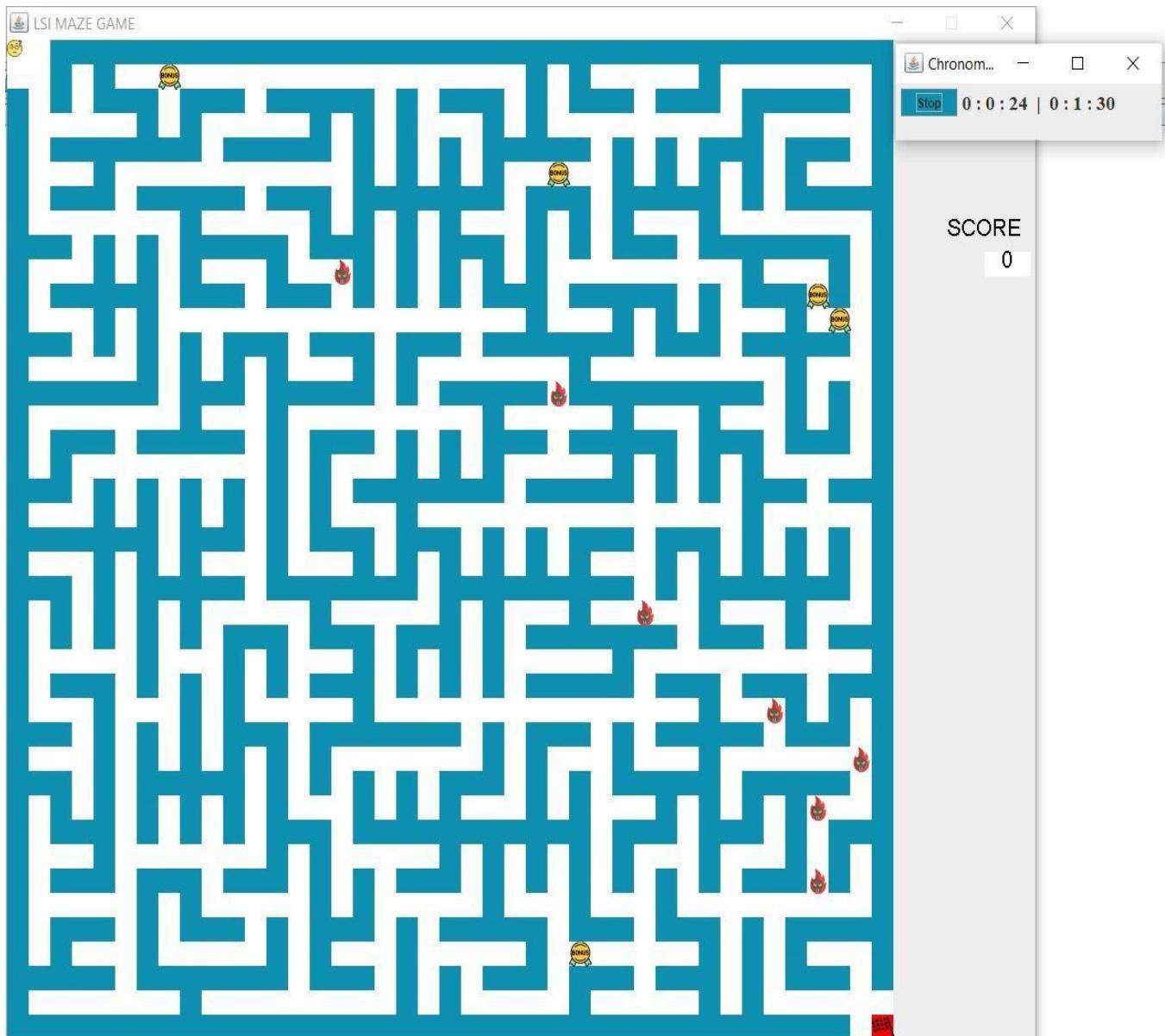
Demande d'aide



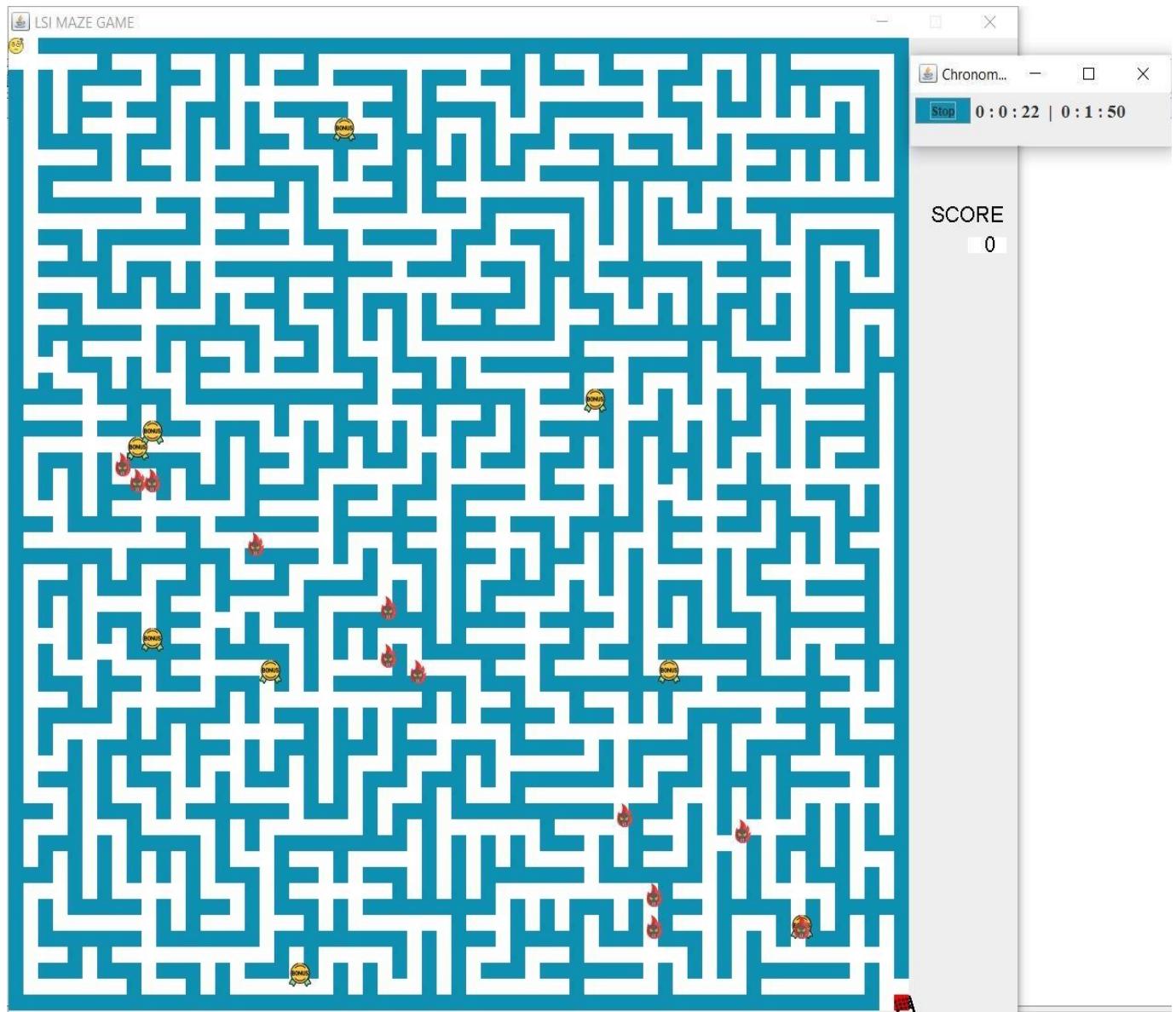
Labyrinthe(21* 21)



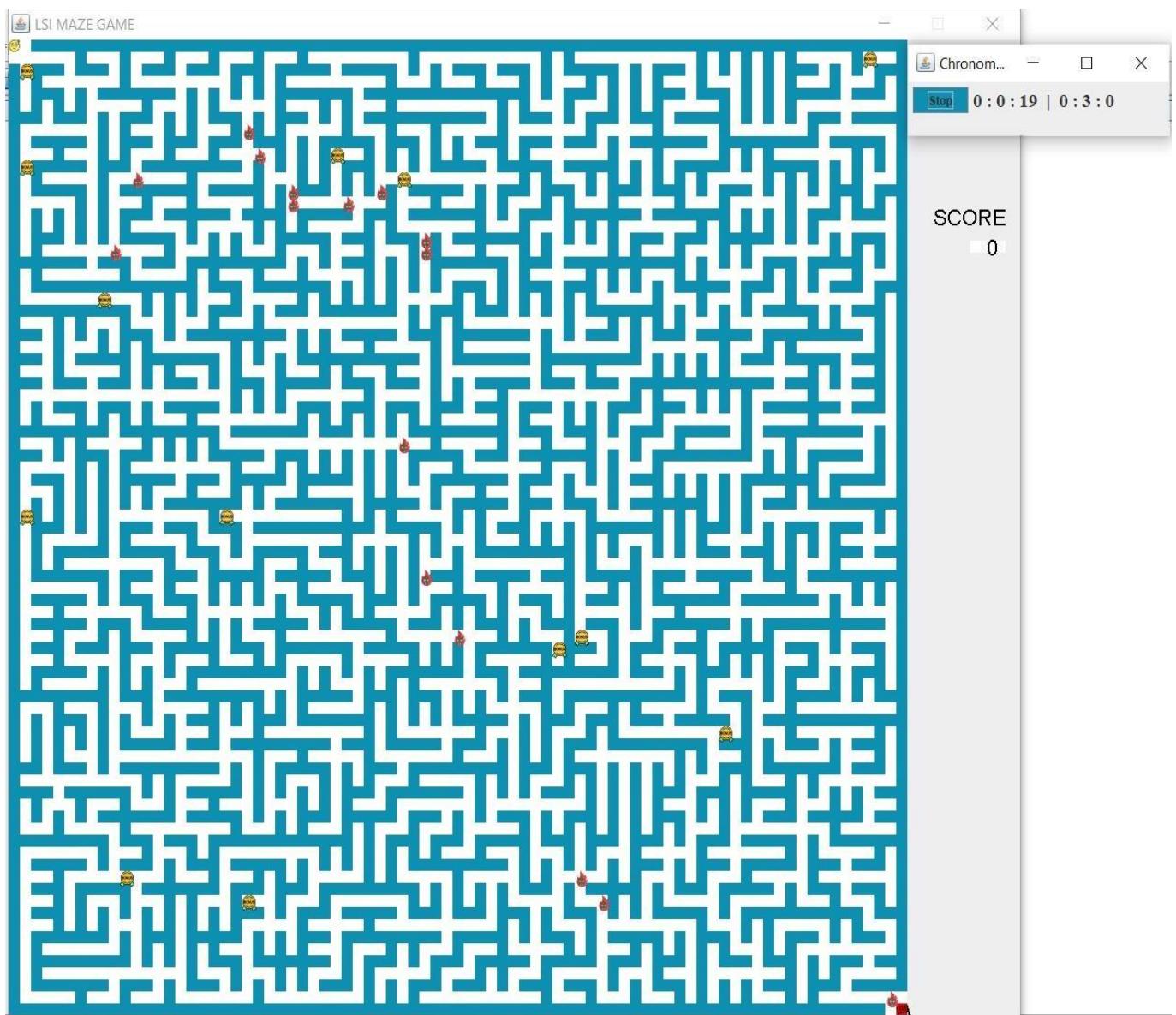
Labyrinthe(41* 41)



Labyrinthe(61* 61)



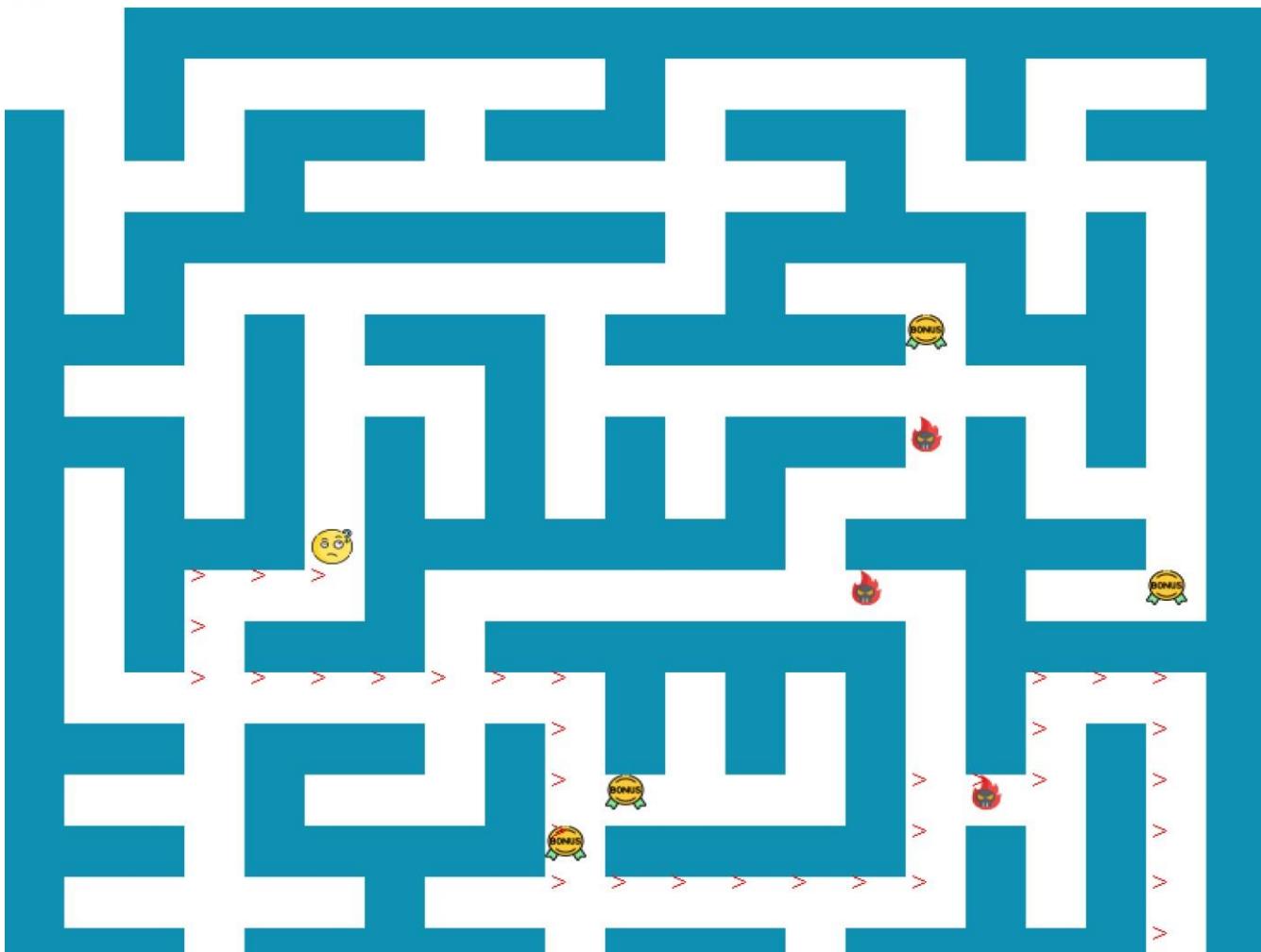
Labyrinthe(81* 81)



Visualisation du chemin

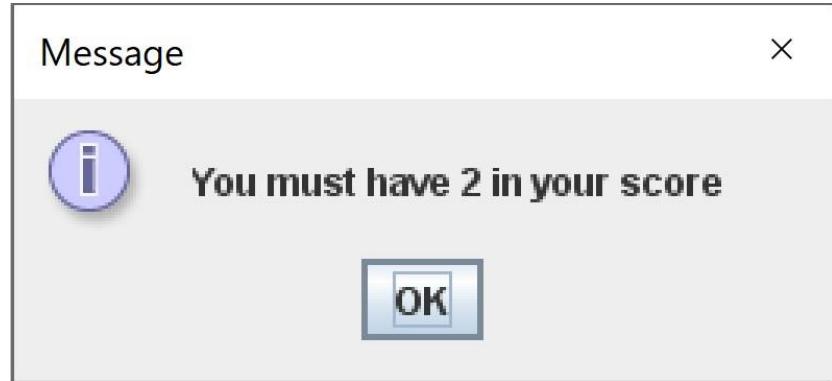
Si le joueur a un score minimum prédéfini, il peut visualiser le chemin vers la sortie selon sa position.

LSI MAZE GAME



Refus d'aide

Si le joueur n'a pas un score minimum prédéfini, il ne peut pas visualiser le chemin vers la sortie.



Fin du jeu

Message pour indiquer la fin du jeu lorsque le délai programmé est terminé.



Conclusion

L'application "**LSI MAZE GAME**" est basée sur les algorithmiques avancées pour l'Intelligence Artificielle tels que BFS, DFS et Astar; qui

40

constituent la base du moteur d'intelligence artificielle pour la recherche du chemin optimal dans les jeux .