LightsOut

Hammadi Rafik Ouariachi

January 11, 2023

1 Introduction

Lights Out est un jeu électronique commercialisé par Tiger Electronics en 1995. Le jeu est composé d'une grille de cinq par cinq lumières. Quand le jeu commence, un nombre aléatoire ou un motif enregistré de ces lumières s'allument. Appuyer sur l'une des lumières basculera la position des lumières adjacentes à celle-ci. Le but du jeu est d'éteindre toutes les lumières, de préférence avec le moins de coups possible.

2 Première Implementation

Une première tentative d'implementation était de définir la classe "Case" qui contient l'information de l'allumage modélisée par "0:false" ou "1:true", mais j'ai été conseillé par le professeur a utiliser une matrice d'éléments appartenant à l'ensemble 0,1 car un élement d'une telle matrice contient suffisemment d'information pour modeliser le jeu.

2.1 La classe "Case"

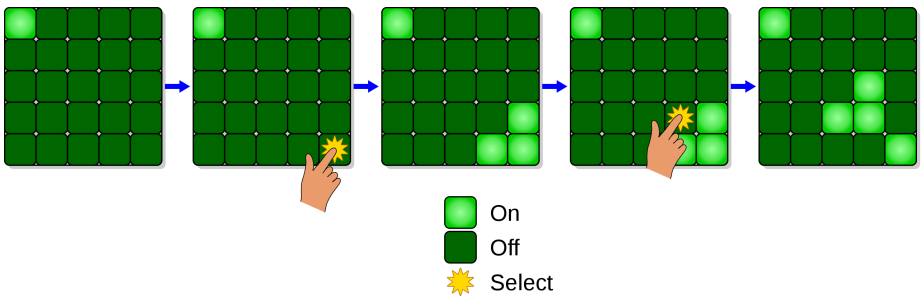

Figure 1: Le concept du jeu.
https://fr.wikipedia.org/wiki/LightsOut(jeu)

```java
public class Cases {
    //attributs
    private boolean isallume;
    private int x;
    private int y;
    //constructeurs
    public Cases(int x, int y, boolean isallume){
        this.isallume=isallume;
        this.x=x;
        this.y=y;
    }
    public Cases(){
        this(0,0,false);
    }
    //..
}
```

2.2 La classe du Jeu

J'ai fini l'implementation du jeu en utilisant la classe "Case" en le modélisant en tant qu'une matrice de Case (tableau bi-dimensionelle) et en ajoutant La methode du changement d"état.

```java
public class Game {
    //attributs
    private Cases[][] board;
    private int casesallumes;
    private int taille;
    //Constructeurs
    public Game(int casesallumes, int taille) {
        this.board = new Cases[taille][taille];
        this.casesallumes = casesallumes;
        this.taille = taille;
        for (int i=0; i<this.taille; i++){
            for (int j=0; j<this.taille;j++){
                this.board[i][j]=new Cases(i,j,false);
            }
        }
    }
    //methods

    public int getCasesallumes() {
        return casesallumes;
    }

    public void setCasesallumes(int casesallumes) {
        this.casesallumes = casesallumes;
    }
```

```
public int getTaille() {
    return taille;
}

public void setTaille(int taille) {
    this.taille = taille;
}

public void casechetatX(Cases case1){
    if (case1.getX()<taille-1 && case1.getX()>0){
        this.board[case1.getX()+1][case1.getY()].setIsallume
        (!this.board[case1.getX()+1][case1.getY()].isIsallume());
        this.board[case1.getX()-1][case1.getY()].setIsallume
        (!this.board[case1.getX()-1][case1.getY()].isIsallume());
    }
    else if (case1.getX()==0){
        this.board[case1.getX()+1][case1.getY()].setIsallume
        (!this.board[case1.getX()+1][case1.getY()].isIsallume());
    }
    else {
        this.board[case1.getX()-1][case1.getY()].setIsallume
        (!this.board[case1.getX()-1][case1.getY()].isIsallume());
    }
}
public void casechetatY(Cases case1){
    if (case1.getY()<taille-1 && case1.getY()>0){
        this.board[case1.getX()][case1.getY()+1].setIsallume
        (!this.board[case1.getX()][case1.getY()+1].isIsallume());
        this.board[case1.getX()][case1.getY()-1].setIsallume
        (!this.board[case1.getX()][case1.getY()-1].isIsallume());
    }
    else if (case1.getY()==0){
        this.board[case1.getX()][case1.getY()+1].setIsallume
        (!this.board[case1.getX()][case1.getY()+1].isIsallume());
    }
    else {
        this.board[case1.getX()][case1.getY()-1].setIsallume
        (!this.board[case1.getX()][case1.getY()-1].isIsallume());
    }
}
public void caseclicked(Cases case1){
    this.board[case1.getX()][case1.getY()].setIsallume
    (!case1.isIsallume());
    this.casechetatX(case1);
    this.casechetatY(case1);
}

public Cases[][] getBoard() {
    return board;
```

```
    }
}
```

2.3 L'interface graphique

Puis, j'ai implementé une classe "GameFrame" qui etend la classe JFrame et implemente le MouseListener pour l'interface graphique, en choisissant la couleure verte pour l'état "allumée" et rouge pour l'état "éteinte"

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.MouseListener;
import javax.swing.JFrame;
import java.awt.event.MouseEvent;

public class GameFrame extends JFrame implements MouseListener {

    private Game game;

    //
    public GameFrame(Game game) {
        super();
        this.setVisible(true);
        this.game = game;
        this.setBounds(0, 0, 1000, 1000);
        this.addMouseListener(this);
    }

    public Game getGame() {
        return game;
    }

    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        for (int i = 0; i < game.getTaille() * 200; i = i + 200) {
            for (int j = 0; j < game.getTaille() * 200; j = j + 200) {
                g2.setColor(Color.black);
                g2.drawRoundRect(i, j, 200, 200, 40, 40);
                if (this.game.getBoard()[(int) i / 200][(int) j / 200].
                isIsallume()) {
                    g2.setColor(Color.GREEN);
                    g2.fillRoundRect(i, j, 199, 199, 40, 40);
                    g2.fillRoundRect(i, j, 199, 199, 40, 40);
                } else {
                    g2.setColor(Color.RED);
                    ..
                }
```

```
        }
    }
}

public void PaintX(Cases case1, Graphics g) {
    int x = case1.getX();
    int y = case1.getY();
    Graphics2D g2 = (Graphics2D) g;
    if (x < this.game.getTaille() - 1 && x > 0) {
        if (this.game.getBoard()[x + 1][y].isIsallume()) {
            g2.setColor(Color.GREEN);
            g2.fillRoundRect((x + 1) * 200, y * 200, 199, 199, 40, 40);
        } else {
            g2.setColor(Color.RED);
            g2.fillRoundRect((x + 1) * 200, y * 200, 199, 199, 40, 40);
        }
        if (this.game.getBoard()[x - 1][y].isIsallume()) {
            g2.setColor(Color.GREEN);
            g2.fillRoundRect((x - 1) * 200, y * 200, 199, 199, 40, 40);
        } else {
            g2.setColor(Color.RED);
            g2.fillRoundRect((x - 1) * 200, y * 200, 199, 199, 40, 40);

        }
    } else if (x == 0) {
        if (this.game.getBoard()[x + 1][y].isIsallume()) {..
        }
    } else {
        if (this.game.getBoard()[x - 1][y].isIsallume()) {..

        }
    }
}

public void PaintY(Cases case1, Graphics g) {
..}
public void PaintXY(Cases case1, Graphics g) {
    int x = case1.getX();
    int y = case1.getY();
    Graphics2D g2 = (Graphics2D) g;
    if (this.game.getBoard()[x][y].isIsallume()) {
        g2.setColor(Color.GREEN);
        g2.fillRoundRect(x * 200, y * 200, 199, 199, 40, 40);
    } else {
        g2.setColor(Color.RED);
        g2.fillRoundRect(x * 200, y * 200, 199, 199, 40, 40);
    }
    this.PaintX(case1, g);
    this.PaintY(case1, g);
}
```

```
    public void mouseClicked(MouseEvent e) {
        int x = (int) (e.getX() / 200);
        int y = (int) (e.getY() / 200);
        Graphics g = getGraphics();
        this.game.caseclicked(this.game.getBoard()[x][y]);
        System.out.println(x + "," + y);
        System.out.println(e.getX() + "," + e.getY());
        this.PaintXY(this.game.getBoard()[x][y], g);
    }

    public void mousePressed(MouseEvent e) {
    ..}
}
```

3 Implementation matricielle

Après, j'ai crée un nouveau projet pour l'implementation matricielle, le jeu a été modélisé en tant qu'un matrice boolénne (true: allumée, false:éteinte), la démarche a été similaire qu'en avant.

```
public class Game {

    private Boolean[][] board;
    private int Taille;

    public int getTaille() {
        return Taille;
    }

    public void setTaille(int Taille) {
        this.Taille = Taille;
    }

    public Game(int taille) {
        this.Taille = taille;
        this.board = new Boolean[taille][taille];
        for (int i = 0; i < taille; i++) {
            for (int j = 0; j < taille; j++) {
                this.board[i][j] = false;
                //randomhavetoadd
            }
        }
    }
```

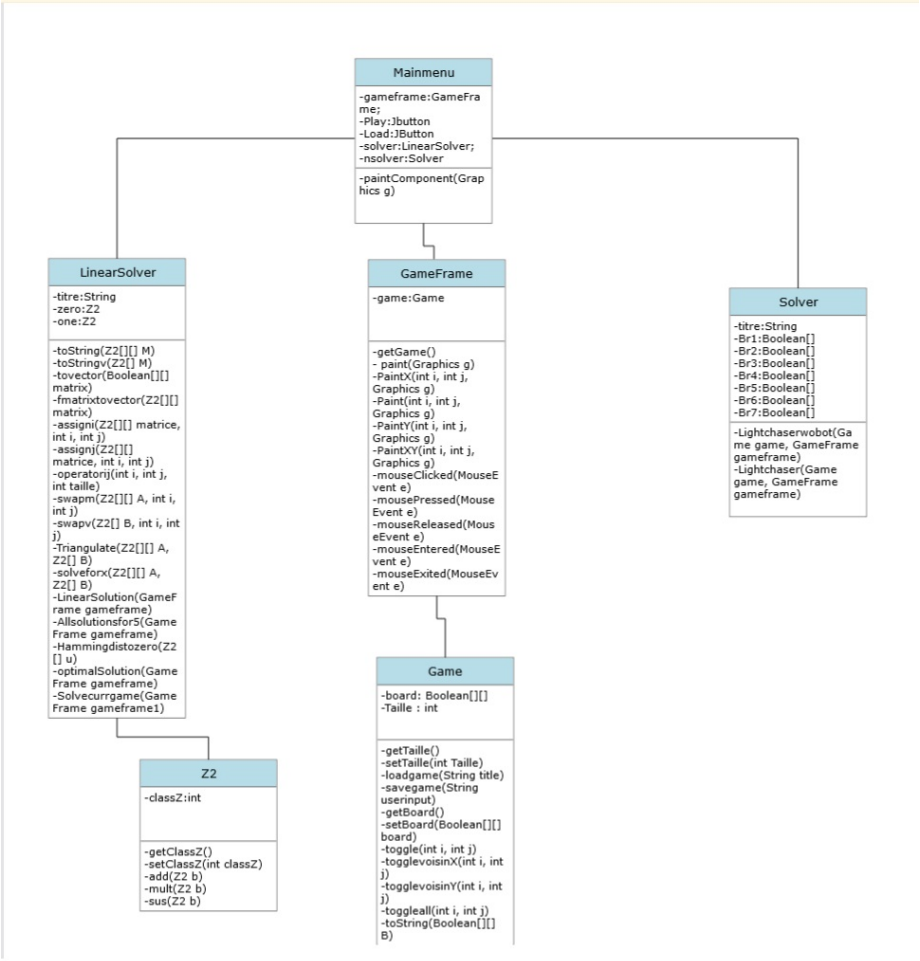

Figure 2: Diagramme UML du projet.

3.1 Sauvegarde et chargement

J'ai implementé deux méthodes dans la classe du Jeu pour la sauvegarde d'un jeu et son chargement, la méthode est simple, elle traduit la matrice du jeu en un chaine de caractères des états ligne par ligne et le sauvegarde dans un fichier texte, clairement la methode du chargement se repose sur la lecture du fichier texte sauvegardé préalablement.

```
public void loadgame(String title) {
    try {
        File savefile = new File(title);
        Scanner myReader = new Scanner(savefile);
        myReader.nextLine();
        while (myReader.hasNextLine()) {
            for (int i = 0; i < this.Taille; i++) {
                for (int j = 0; j < this.Taille; j++) {
                    this.board[i][j] = Boolean.parseBoolean(myReader.nextLine());
                }
            }
        }
        myReader.close();
    } catch (FileNotFoundException e) {
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}

public void savegame(String userinput) {
    try {
        File savegamefile = new File(userinput);
        if (savegamefile.createNewFile()) {
            try {
                FileWriter myWriter = new FileWriter(userinput);
                myWriter.write(this.Taille + "\n");
                for (int i = 0; i < getTaille(); i++) {
                    for (int j = 0; j < Taille; j++) {
                        myWriter.write(board[i][j].toString() + "\n");
                    }
                }
                myWriter.close();
            } catch (IOException g) {
                System.out.println("An error occurred.");
            }
        } else {
            System.out.println("File already exists.");
        }
    } catch (IOException f) {
        System.out.println("An error occurred.");
    }
}
```

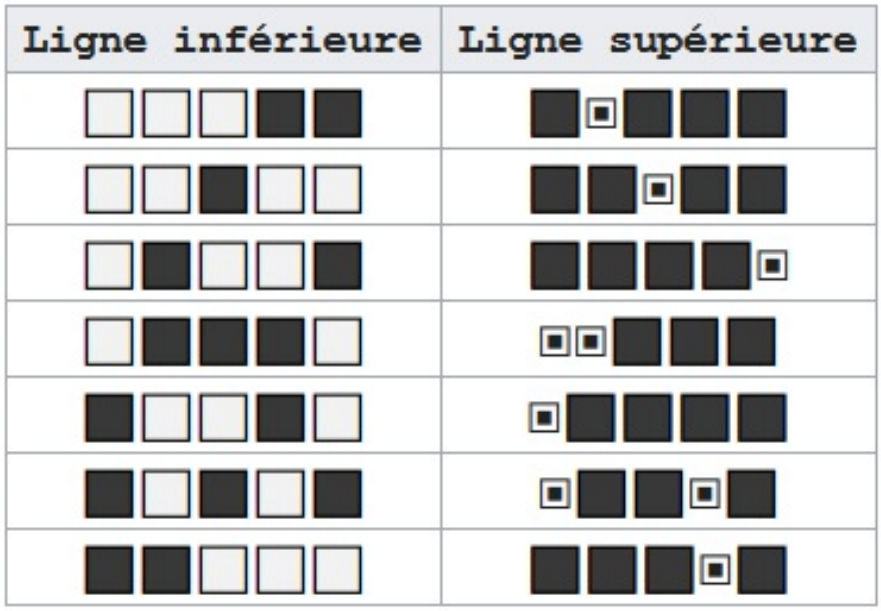

Figure 3: Les suites correspondentes.
https://fr.wikipedia.org/wiki/LightsOut(jeu)

---

4 Résolution du jeu

4.1 Implementation du Solveur-chasse à la lumière

La résolution du jeu à travèrs l'algorithme du "chasse à la lumière" se fait en cliquant sur chaque case dont celle qui est au dessus est allumée , après on retient une des combinaison suivante:
true,true,true,false,false;
true,true,false,true,true;
true,false,true,true,false;
true,false,false,false,true;
false,true,true,false,true;
false,true,false,true,false;
false,false,true,true,true;
Après, on fait une suite de clics sur la première ligne afin d'obtenir la combinaison correspondente suivant ceci : et puis on refait "la chasse à la lumière" en étant rassuré que toutes les cases seront éteintes maintenant. Pour l'animation

, on ajoute la ligne "Thread.Sleep()" pour donner un intervalle de temps entre les clics , ce qui donne la résolution un aspect animé.

```
import java.awt.Graphics;
import java.util.Arrays;

    public class Solver {
    private String titre;

    Boolean[] Br1= {true,true,true,false,false};
    Boolean[] Br2= {true,true,false,true,true};
    Boolean[] Br3= {true,false,true,true,false};
    Boolean[] Br4= {true,false,false,false,true};
    Boolean[] Br5= {false,true,true,false,true};
    Boolean[] Br6= {false,true,false,true,false};
    Boolean[] Br7= {false,false,true,true,true};
    public Solver(String titre){
        this.titre=titre;
    }
    public void Lightchaserwobot(Game game, GameFrame gameframe) throws InterruptedException
{
        if (game.getTaille()==5){
        for (int j = 1; j < 5; j++) {
            for (int i = 0; i < 5; i++) {
                if (game.getBoard()[i][j - 1]) {
                    game.toggleall(i, j);
                    Graphics g = gameframe.getGraphics();
                    gameframe.PaintXY(i, j, g);
                    Thread.sleep(500);
                }
            }
        }
        }
    }
    public void Lightchaser(Game game, GameFrame gameframe) throws InterruptedException{
        Graphics g = gameframe.getGraphics();
        this.Lightchaserwobot(game,gameframe);
        Boolean[] BottomRow = new Boolean[5];
        for (int i=0; i<5;i++){
            BottomRow[i]=game.getBoard()[i][4];
        }
        System.out.println(Arrays.toString(BottomRow));
        if(Arrays.equals(Br1, BottomRow)){
            game.toggleall(1, 0);
            gameframe.PaintXY(1, 0, g);
            Thread.sleep(500);
            this.Lightchaserwobot(game,gameframe);
        }
        if(Arrays.equals(Br2, BottomRow)){
        ..}
```

4.2 Résolution avec l'algebre linéaire

Maintenant pour des jeu de taille differentes de 5, et même pour ceux de taille de 5, on cherche un algorithme plus efficace et rapide et de nombre minimal de clics qui résolue le jeu, pour cela ,on prend recours à l'algebre linéaire, évidemment le jeu , étant modéliser par une matrice de booléen , on peut chercher à le résoudre en utilisant des outils algébrique.

4.2.1 La théorie

Premièrement, considèrant l'exemple d'un jeu de taille 5 de matrice :
$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$
Prenons par hasard les indices (1,3), on commence par 0 pour que ce soit le même du code, si on clique sur la case (1,3) la matrice devient: $\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$
on remarque alors que le clique est equivalent à l'ajout de la matrice :
$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ à la matrice du jeu en Z/2Z On remarque aussi que : si on fait une serie des cliques : $\{(i_k, j_k), k \in N^*\}$, l'ordre des cliques ne change rien , du coup pour un jeu au hasard, la serie des cliques qui résout le jeu est la même qui, en commencant d'un état résolu nous mène au jeu considéré.
On formalise cette idée : Pour un jeu de taille n si A_{ij} est la matrice modélisant un clic sur la case (i, j) et M est la matrice du jeu, Alors on a :

$$(E) : M = \sum_{(i,j)\in[|0,n-1|]^2} S_{ij}.A_{ij} \text{ avec } S_{ij} = \begin{cases} 1 & \text{si (i,j) appartient à la solution} \\ 0 & \text{sinon.} \end{cases}$$

Pour des raisons de commodité , on transforme chaque matrice en un vecteur de taille : n^2 en concaténant les vecteur lignes de la matrice Par exemple, cette

matrice:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

devient:

$$(0,0,0,0,0,1,0,0,0)^T$$

On retient donc n^2 vecteur de taille n^2 lors de la transformation des matrices A_{ij} , on réarrange ces vecteurs pour obtenir une matrice A de taille $n^2 * n^2$, la matrice M devient un vecteur de taille n^2 qu'on note [M] et les S_{ij} se regroupe dans un vecteur de taille n^2 qu'on note [S] tel que $S_{ij} = [S]_{n*i+j}$ l'equation (E) représente mainteant le systeme linéaire:

$$A * [S] = [M]$$

On résout le système à travers le pivot de Gauss or, on veut une solution unique cependant, cela n"est possible que si le rang de la matrice A est n^2 , condition qui n'est pas satisfaite pour des jeu de taille 5 dont la matrice est de rang 23.

$$A = \begin{pmatrix}
110001000000000000000000000 \\
111000100000000000000000000 \\
011100010000000000000000000 \\
001110001000000000000000000 \\
000111000100000000000000000 \\
100001100010000000000000000 \\
010001110001000000000000000 \\
001000111000100000000000000 \\
000100011100010000000000000 \\
000010001100001000000000000 \\
000001000110001000000000000 \\
000000100011100010000000000 \\
000000010001110001000000000 \\
000000001000111000100000000 \\
000000000100011100010000000 \\
000000000010000110001000000 \\
000000000001000111000100000 \\
000000000000100011100010000 \\
000000000000010001100001000 \\
000000000000001000011000100 \\
000000000000000100011100010 \\
000000000000000010001100001 \\
000000000000000001000011000 \\
000000000000000000100011100 \\
000000000000000000010001110 \\
000000000000000000001000111 \\
000000000000000000000100011
\end{pmatrix}$$

Une solution existe si et seulement si au complement orthogonal de $KerA$ En effet, le pivot de gauss, appliqué à la matrice A donne une matrice E telle qu'une solution existe si et seulement si [S] appartient au complement orthogonal de $KerE$ et puisque $KerA = KerE$, on retrouve une base de $KerE$ qui est (v_1, v_2 avec:

$$v_1 = (0,1,1,1,0,1,0,1,0,1,1,1,0,1,1,1,0,1,0,1,0,1,1,1,0)^T$$

$$v_2 = (1,0,1,0,1,1,0,1,0,1,0,0,0,0,0,1,0,1,0,1,1,0,1,0,1)^T$$

Finalement, si [S] est solution

$S_0 = [S] + v_1$, $S_1 = [S] + v_1$, $S_2 = [S] + v_2$, $S_3 = [S] + v_1 + v_2$

sont les solutions. Alors comment retrouver la meilleure solution? ie: la solution avec les moins de clic. Mathématiquement, la solution avec les moins de clic est celle avec la distance minimale à une solution sans clique (ie: le vecteur nul qu'on note par 0), donc on considère la distance de Hamming :

Pour deux vecteurs a et b , $d(a,b) = Card(\{i, a_i \neq b_i\}$

et donc La solution retenue S_{op} est :

$$S_{op} = \min_{\mathrm{d(S_i,0)}} \{S_i, i = 0,1,2,3\}$$

4.2.2 L'implementation de la résolution linéaire

Premièrement pour facilité les opération dans $Z/2Z$ j'ai crée la classe "Z2" des éléments appartenant à $Z/2Z$

```
public class Z2 {
private int classZ;
public Z2(int a){
    if (a==1||a==0){
        this.classZ=a;
    }
}

public int getClassZ() {
    return classZ;
}

public void setClassZ(int classZ) {
    this.classZ = classZ;
}

public Z2 add(Z2 b){
```

```
        if (this.classZ+b.classZ==2){
            return (new Z2(0));
        }
        else{
        return(new Z2(this.classZ+b.classZ));
        }
    }

    public Z2 mult(Z2 b){
        return(new Z2(this.classZ*b.classZ));
    }
    public Z2 sus(Z2 b){
        return(new Z2(Math.abs(this.classZ-b.classZ)));
    }
}
```

Après le test de la dernière j'ai crée la classe du solveur linéaire et j'ai rédiger les méthodes qui permet d'obtenir le vecteur du jeu en $Z/2Z$ directement à partir de la matrice booléenne et les méthodes pour l'impression.

```
    public class LinearSolver {

    private String titre;
    Z2 zero = new Z2(0);
    Z2 one = new Z2(1);
    public LinearSolver(String titre) {
        this.titre = titre;
    }

    public String toString(Z2[][] M) {
        String MtoS = "";
        for (int i = 0; i < M.length; i++) {
            for (int j = 0; j < M.length; j++) {
                MtoS += M[i][j].getClassZ() + " ";
            }
            MtoS += "\n";
        }
        return MtoS;
    }

    public String toStringv(Z2[] M) {
        String MtoS = "";

        for (int j = 0; j < M.length; j++) {
            MtoS += M[j].getClassZ() + " ";
        }
        MtoS += "\n";

        return MtoS;
```

```
}

public Z2[] tovector(Boolean[][] matrix) {
    Z2[] Bvector = new Z2[matrix.length * matrix.length];
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix.length; j++) {
            if (matrix[i][j]) {
                Bvector[i * matrix.length + j] = new Z2(1);
            } else {
                Bvector[i * matrix.length + j] = new Z2(0);
            }
        }
    }
    return Bvector;
}

public Z2[] fmatrixtovector(Z2[][] matrix) {
    Z2[] ourvector = new Z2[matrix.length * matrix.length];
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix.length; j++) {
            ourvector[i * matrix.length + j] = new Z2(matrix[i][j].getClassZ());
        }
    }
    return ourvector;
}
```

Puis, j'ai rédigé les methodse pour obtenir les matrices A_{ij} et la matrice A :

```
public void assigni(Z2[][] matrice, int i, int j) {
    if (i < matrice.length - 1 && i > 0) {
        matrice[i + 1][j] = new Z2(1);
        matrice[i - 1][j] = new Z2(1);
    } else if (i == 0) {
        matrice[i + 1][j] = new Z2(1);
    } else {
        matrice[i - 1][j] = new Z2(1);
    }
}

public void assignj(Z2[][] matrice, int i, int j) {
    if (j < matrice.length - 1 && j > 0) {
        matrice[i][j + 1] = new Z2(1);
        matrice[i][j - 1] = new Z2(1);
    } else if (j == 0) {
        matrice[i][j + 1] = new Z2(1);
    } else {
        matrice[i][j - 1] = new Z2(1);
    }
}
```

```
public Z2[][] operatorij(int i, int j, int taille) {
    Z2[][] Aij = new Z2[taille][taille];
    for (int ii = 0; ii < taille; ii++) {
        for (int jj = 0; jj < taille; jj++) {
            Aij[ii][jj]=new Z2(0);
        }
    }
    Aij[i][j] = new Z2(1);
    this.assigni(Aij, i, j);
    this.assignj(Aij, i, j);
    return Aij;
}
```

Maintenant il restait de rediger l'algorithme du pivot de gauss qui nécessitait de rédiger deux méthodes pour la permutation dse lignes dans une matrice ou un vecteur:

```
public void swapm(Z2[][] A, int i, int j) {
    for (int k = 0; k < A.length; k++) {
        Z2 tmp = new Z2(A[i][k].getClassZ());
        A[i][k] = new Z2(A[j][k].getClassZ());
        A[j][k] = new Z2(tmp.getClassZ());
    }
}

public void swapv(Z2[] B, int i, int j) {
    Z2 tmp = new Z2(B[i].getClassZ());
    B[i] = new Z2(B[j].getClassZ());
    B[j] = new Z2(tmp.getClassZ());
}

public void Triangulate(Z2[][] A, Z2[] B) {
    for (int i = 0; i < A.length; i++) {
        Z2 sup = new Z2(0);
        int j = 0;
        for (int k = i; k < A.length; k++) {
            if (sup.getClassZ() < A[k][i].getClassZ()) {
                sup = A[k][i];
                j = k;
            }
        }
        if (sup.getClassZ()>0){
        swapm(A, i, j);
        swapv(B, i, j);
        for (int k = i + 1; k < A.length; k++) {
            Z2 tmp = new Z2(0);
            tmp = A[k][i];
            for (int l = 0; l < A.length; l++) {
                A[k][l] = A[k][l].sus(tmp.mult(A[i][l]));
```

```
            }
            B[k] = B[k].sus(tmp.mult(B[i]));
        }
        }
    }

}

public Z2[] solveforx(Z2[][] A, Z2[] B) {
    this.Triangulate(A, B);
    //System.out.println(toString(A));
    Z2[] X = new Z2[A.length];
    X[A.length - 1] = B[A.length - 1];
    for (int i = A.length - 1; i >= 0; i--) {
        for (int j = i + 1; j < A.length; j++) {
            B[i] = B[i].sus(A[i][j].mult(X[j]));
        }
        X[i] = B[i];
    }
    return X;
}
```

Finalement, j'ai implementé les méthodes pour calculer la distance de Hamming et pour obtenir la solution optimale qui reposait sur un tri d'un tableau de 4 éléments:

```
public Z2[] LinearSolution(GameFrame gameframe) {
    Z2[][] A = new Z2[gameframe.getGame().getTaille() * gameframe.getGame().getTaille()]
    [gameframe.getGame().getTaille() * gameframe.getGame().getTaille()];
    for (int ii = 0; ii < gameframe.getGame().getTaille(); ii++) {
        for (int jj = 0; jj < gameframe.getGame().getTaille(); jj++) {
                Z2[][] Aiijj=operatorij(ii, jj, gameframe.getGame().getTaille());
            for (int j = 0;
            j < gameframe.getGame().getTaille() * gameframe.getGame().getTaille(); j++) {
                A[ii * gameframe.getGame().getTaille() + jj][j] = new Z2(fmatrixtovector(Aiijj)[j]
                .getClassZ());
            }

        }

    }
    System.out.println(toString(A));
    Z2[] C = new Z2[gameframe.getGame().getTaille() * gameframe.getGame().getTaille()];
    C = this.tovector(gameframe.getGame().getBoard());

    //System.out.println(toStringv(C));
    Z2[] S1 = new Z2[gameframe.getGame().getTaille() * gameframe.getGame().getTaille()];
    S1 = this.solveforx(A, C);
    //System.out.println(toStringv(S1));
```

```
    return S1;

}
public Z2[][] Allsolutionsfor5(GameFrame gameframe){
    Z2[][] S = new Z2[4][25];
    Z2[] S1 = this.LinearSolution(gameframe);
    Z2[] v1={zero,one,one,one,zero,one,zero,one,zero,one,
    one,one,zero,one,one,one,zero,one,zero,one,zero,one,one,one,zero};
    Z2[] v2={one,zero,one,zero,one,one,zero,one,zero,one,zero,
    zero,zero,zero,zero,one,zero,one,zero,one,one,zero,one,zero,one};
    for (int i=0; i<25; i++){
    S[0][i]=S1[i];
    S[1][i]=S1[i].add(v1[i]);
    S[2][i]=S1[i].add(v2[i]);
    S[3][i]=S1[i].add(v1[i]).add(v2[i]);
    }
    return S;
}
public int Hammingdistozero(Z2[] u){
    int Hammingdisto=0;
    for (int i=0; i<u.length; i++){
        if (u[i].getClassZ()!=0){
            Hammingdisto++;
        }
    }
    return Hammingdisto;
}
public Z2[] optimalSolution(GameFrame gameframe){
    int[] Harray=new int[4];
    Z2[] OpS = new Z2[25];
    Z2[][] S=this.Allsolutionsfor5(gameframe);
    int H=0;
    for (int i=0; i<4;i++){
        Z2[] U = new Z2[25];
        for (int j=0; j<25;j++){
            U[j]=S[i][j];
        }
        Harray[i]=this.Hammingdistozero(U);
    }
    for (int k=0;k<4;k++){
        if (H>Harray[k]){
            H=k;
        }
    }
    for (int i=0; i<25; i++){
        OpS[i]=S[H][i];
    }
    return OpS;
}
```

```
    public void Solvecurrgame(GameFrame gameframe1) throws InterruptedException {
        Graphics g = gameframe1.getGraphics();
        if(gameframe1.getGame().getTaille()!=5){
        for (int i = 0; i < gameframe1.getGame().getTaille(); i++) {
            for (int j = 0; j < gameframe1.getGame().getTaille(); j++) {
                if (this.LinearSolution(gameframe1)[gameframe1.getGame().
                getTaille() * i + j].getClassZ()==1) {
                    gameframe1.getGame().toggleall(i, j);
                    gameframe1.PaintXY(i, j, g);
                    Thread.sleep(500/gameframe1.getGame().getTaille());
                }
            }
        }
        }
        else {
            for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (this.optimalSolution(gameframe1)[gameframe1.getGame().getTaille() * i + j].
                getClassZ()==1) {
                    gameframe1.getGame().toggleall(i, j);
                    gameframe1.PaintXY(i, j, g);
                    Thread.sleep(100);
        }
    }

}
```

4.3 L'interface graphique finale

Finalement, pour l'interface graphique , j'ai crée un menu principal avec deux boutons , un pour le commencement d'un nouveau jeu et un pour le chargement d'un jeu sauvegardé. Le menu principal a été mis en place à travers la classe "Mainmenu" qui implemente ActionListener pour les boutons

```
    import java.awt.Graphics;
import ..

/**
 *
 * @author houariac
 */
public class Mainmenu extends JFrame {

    private GameFrame gameframe;
    JButton Play, Load;
    JLabel Label;
    private LinearSolver solver;
    private Solver nsolver;
```

Figure 4: La fenêtre du menu principal.

```
public Mainmenu() {
    super("LightsOut! by 7madox");
    this.setLayout(null);
    this.setVisible(true);
    this.setBounds(1000, 200, 600, 600);
    Graphics g = getGraphics();
    paintComponent(g);
    Icon playbutton = new ImageIcon("Z:\\Mes documents\\NetBeansProjects\\
    LightsOutversionMatricielle\\UIimg\\Play-button.png");
    Icon loadbutton = new ImageIcon("Z:\\Mes documents\\NetBeansProjects\\
    LightsOutversionMatricielle\\UIimg\\Load-button.png");
    Play = new JButton(playbutton);
    Play.setBounds(0, 200, 600, 100);
    Play.setVisible(true);
    Play.setBorderPainted(false);
    Load = new JButton(loadbutton);
    Load.setBounds(0, 300, 600, 100);
    Load.setVisible(true);
    Load.setBorderPainted(false);
    //setboundsand graphics
    this.add(Play);
    this.add(Load);
    Play.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            int taille = parseInt(JOptionPane.showInputDialog(null, "Taille?"));
            gameframe = new GameFrame(taille);
            Graphics g = gameframe.getGraphics();
            gameframe.paint(g);
            System.out.println(gameframe.getGame().toString(gameframe.getGame().getBoard()));
            Icon savebutton = new ImageIcon("Z:\\Mes documents\\NetBeansProjects\\
            LightsOutversionMatricielle\\UIimg\\Save-Button.png");
            Icon resolvebrutally = new ImageIcon("Z:\\Mes documents\\NetBeansProjects\\
            LightsOutversionMatricielle\\UIimg\\Resolve-Brutally.png");
            Icon resolveLA = new ImageIcon("Z:\\Mes documents\\NetBeansProjects\\
            LightsOutversionMatricielle\\UIimg\\Resolve-LA.png");
            JButton Resolve = new JButton(resolveLA);
            JButton NResolve = new JButton(resolvebrutally);
            JButton Save = new JButton(savebutton);
            Resolve.setBounds(0, 970, 333, 100);
            Resolve.setVisible(true);
            Resolve.setBorderPainted(false);
            gameframe.add(Resolve);
            NResolve.setBounds(334, 970, 333, 100);
            NResolve.setVisible(true);
            NResolve.setBorderPainted(false);
            gameframe.add(NResolve);
            Save.setBounds(667, 970, 333, 100);
```

```
            Save.setVisible(true);
            Save.setBorderPainted(false);
            gameframe.add(Save);
            Resolve.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
                    solver = new LinearSolver("Lsolver");
                    try {
                        solver.Solvecurrgame(gameframe);
                    } catch (InterruptedException ex) {
                        Logger.getLogger(Mainmenu.class.getName()).log(Level.SEVERE, null, ex);
                    }

                }
            });
            NResolve.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
                    nsolver = new Solver("Nsolver");
                    try {
                        nsolver.Lightchaser(gameframe.getGame(), gameframe);
                    } catch (InterruptedException ex) {
                        Logger.getLogger(Mainmenu.class.getName()).log(Level.SEVERE, null, ex);
                    }

                }
            });
            Save.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
                    String userinput = JOptionPane.showInputDialog(null, "Name?");
                    gameframe.getGame().savegame(userinput);

                }
            });

        }
    });

}

public void paintComponent(Graphics g) {
    try {
        BufferedImage img = ImageIO.read(new File("Z:\\Mes documents\\NetBeansProjects\\
        LightsOutversionMatricielle\\UIimg\\background.jpg"));
        g.drawImage(img, 0, 0, null);
    } catch (IOException ex) {
```

```
            Logger.getLogger(Mainmenu.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
```

On remarque le clic sur le bouton "Play" crée une boite de dialogue qui demande la taille du jeu souhaité et puis une "GameFrame" se crée, c'est la classe de l'interface graphique du jeu. Elle contient 3 boutons, un pour la sauvegarde, un pour la résolution "chasse à la lumière" et un pour la résolution linéaire.

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.Timer;

/**
 *
 * @author houariac
 */
public class GameFrame extends JFrame implements MouseListener {

    private Game game;


    //
    public GameFrame(int taille) {
        super("LightsOut! by 7madox");
```

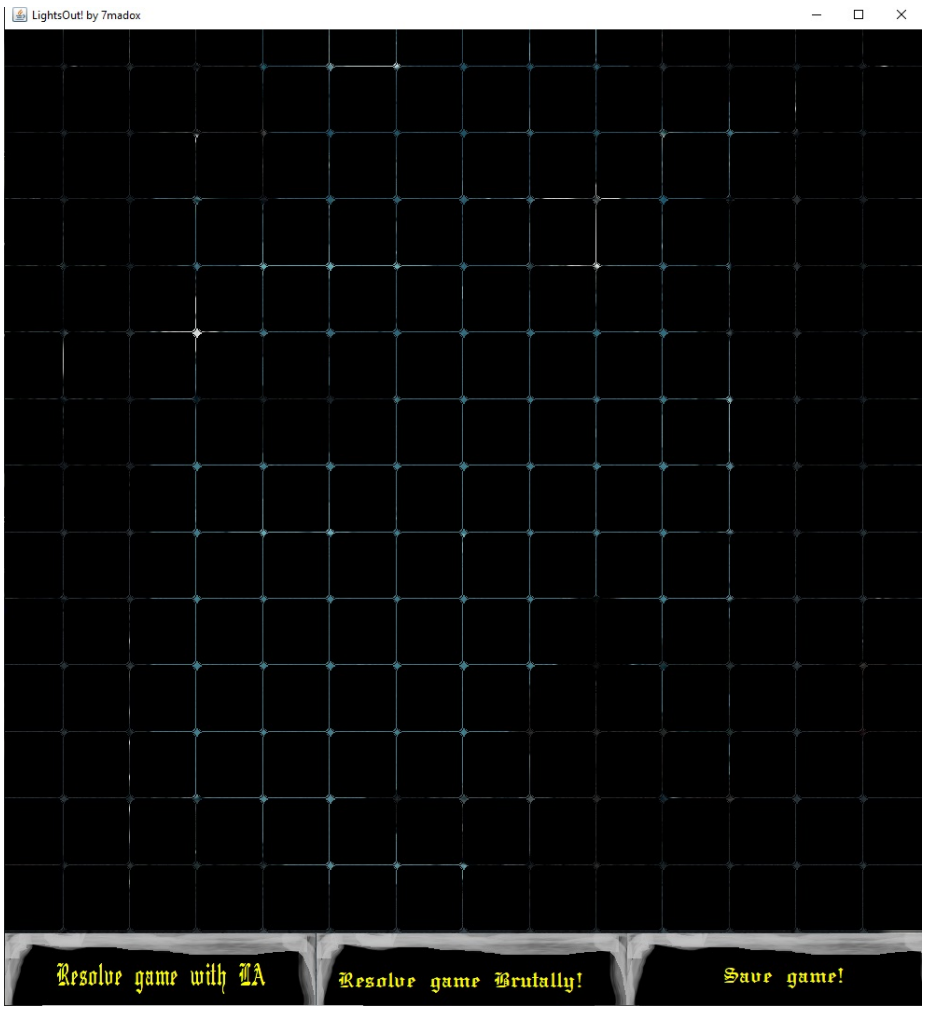

Figure 5: La fenêtre du Jeu.

```
        this.setVisible(true);
        this.game = new Game(taille);
        this.setBounds(0, 0, 1000, 1200);
        this.addMouseListener(this);
        this.setLayout(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setBackground(Color.yellow);


        /*
        add panel for resolution
         */
    }

    public Game getGame() {
        return game;
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        try {
            BufferedImage img = ImageIO.read(new File("Z:\\Mes documents\\NetBeansProjects\\
            LightsOutversionMatricielle\\UIimg\\Gbackground.jpg"));
            g.drawImage(img, 0, 0, null);
        } catch (IOException ex) {
            Logger.getLogger(Mainmenu.class.getName()).log(Level.SEVERE, null, ex);
        }
        Graphics2D g2 = (Graphics2D) g;
        for (double i = 0; Math.round(i) < 1000; i = i + (1000.0 / (double) game.getTaille())) {
            for (double j = 0; Math.round(j) < 1000; j = j + (1000.0 / (double) game.getTaille())) {
                System.out.println(i + j);
                g2.setColor(Color.black);
                g2.drawRoundRect((int) i, (int) j, (1000 / game.getTaille()),
                (1000 / game.getTaille()), (1000 / game.getTaille()) / 5,
                (1000 / game.getTaille()) / 5);
                if (this.game.getBoard()[(int) (i / (1000 / game.getTaille()))]
                [(int) (j / (1000 / game.getTaille()))]) {
                    g2.setColor(Color.YELLOW);
                    g2.fillRoundRect((int) i, (int) j, (1000 / game.getTaille()) - 1,
                    (1000 / game.getTaille()) - 1, (1000 / game.getTaille()) / 5,
                    (1000 / game.getTaille()) / 5);
                    g2.fillRoundRect((int) i, (int) j,
                    (1000 / game.getTaille()) - 1, (1000 / game.getTaille()) - 1,
                    (1000 / game.getTaille()) / 5, (1000 / game.getTaille()) / 5);
                } else {
                    g2.setColor(Color.BLACK);
                    g2.fillRoundRect((int) i, (int) j,
                    (1000 / game.getTaille()) - 1, (1000 / game.getTaille()) - 1,
                    (1000 / game.getTaille()) / 5, (1000 / game.getTaille()) / 5);
```

```
                g2.fillRoundRect((int) i, (int) j,
                (1000 / game.getTaille()) - 1, (1000 / game.getTaille()) - 1,
                (1000 / game.getTaille()) / 5, (1000 / game.getTaille()) / 5);
            }
        }
    }
}

public void PaintX(int i, int j, Graphics g) {
    if (i < this.game.getTaille() - 1 && i > 0) {
        this.Paint(i + 1, j, g);
        this.Paint(i - 1, j, g);
    } else if (i == 0) {
        this.Paint(i + 1, j, g);
    } else {
        this.Paint(i - 1, j, g);
    }
}

public void Paint(int i, int j, Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    if (this.game.getBoard()[i][j]) {
        g2.setColor(Color.YELLOW);
        g2.fillRoundRect((int)
        (i * (1000.0 / (double) game.getTaille())),
        (int) (j * (1000.0 / (double) game.getTaille())),
        (1000 / game.getTaille()) - 1,
        (1000 / game.getTaille()) - 1,
        (1000 / game.getTaille()) / 5, (1000 / game.getTaille()) / 5);
    } else {
        g2.setColor(Color.BLACK);
        g2.fillRoundRect((int)
        (i * (1000.0 / (double) game.getTaille())), (int)
        (j * (1000.0 / (double) game.getTaille())),
        (1000 / game.getTaille()) - 1,
        (1000 / game.getTaille()) - 1,
        (1000 / game.getTaille()) / 5, (1000 / game.getTaille()) / 5);
    }
}

public void PaintY(int i, int j, Graphics g) {
    if (j < this.game.getTaille() - 1 && j > 0) {
        this.Paint(i, j + 1, g);
        this.Paint(i, j - 1, g);
    } else if (j == 0) {
        this.Paint(i, j + 1, g);
    } else {
        this.Paint(i, j - 1, g);
    }
}
```

```
public void PaintXY(int i, int j, Graphics g) {
    this.Paint(i, j, g);
    this.PaintX(i, j, g);
    this.PaintY(i, j, g);
}

public void mouseClicked(MouseEvent e) {
    int i = (int) (e.getX() / (1000 / game.getTaille()));
    int j = (int) (e.getY() / (1000 / game.getTaille()));
    Graphics g = getGraphics();
    this.game.toggleall(i, j);
    System.out.println(i + "," + j);
    System.out.println(e.getX() + "," + e.getY());
    this.PaintXY(i, j, g);
}


public void mousePressed(MouseEvent e) {
}

public void mouseReleased(MouseEvent e) {
}

public void mouseEntered(MouseEvent e) {
}

public void mouseExited(MouseEvent e) {
}
```