Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Série 2

Mine Hunt - Création d'un démineur

David Alvarez Luca Moos

18 avril 2017

Table des matières

1 Introduction

1.1 Objectifs

- Créer une application *Java* comportant une interface utilisateur graphique
- Réaliser une vue en créant un graphe de scène et en configurant les conteneurs et les composants
- Insérer des images (ressources externes) dans une application
- Structurer l'application en basant la conception et le codage sur l'architecture *MVC*[1]. Créer une interface du modèle permettant des variantes d'implémentation
- Gérer les actions de l'utilisateur en traitant des événements de type **ActionEvent** et **MouseEvent**
- Gérer un menu et créer des boîtes de dialogue simples pour informer ou questionner l'utilisateur

2 Application

2.1 Vues

2.1.1 Vue principale

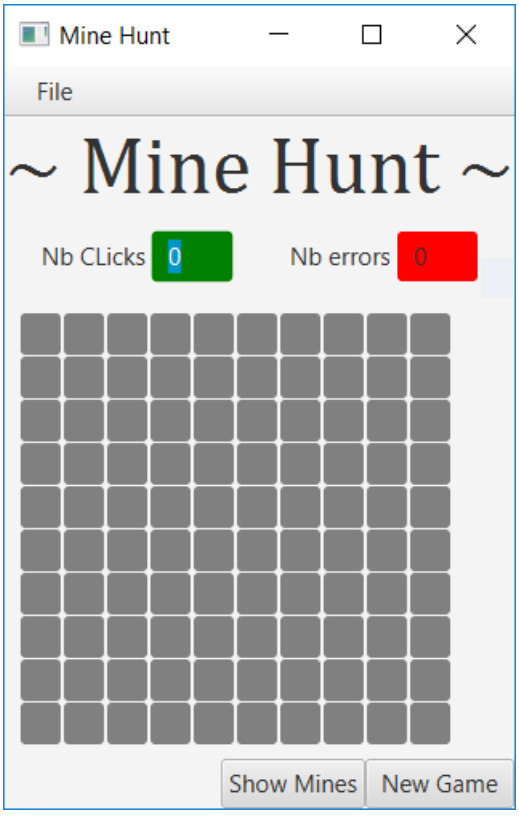

FIGURE 1 – Vue principale

1. Model-View-Controller

2.1.2 Settings

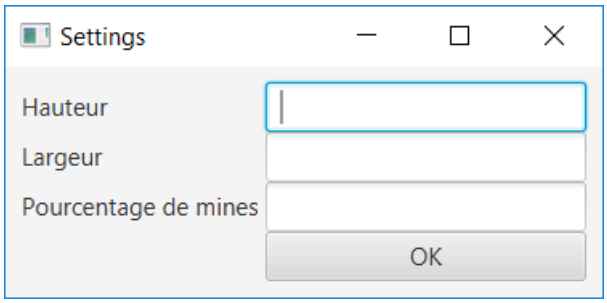

FIGURE 2 – Vue *Settings*

2.1.3 Vue principale avec un terrain plus grand en cours de partie

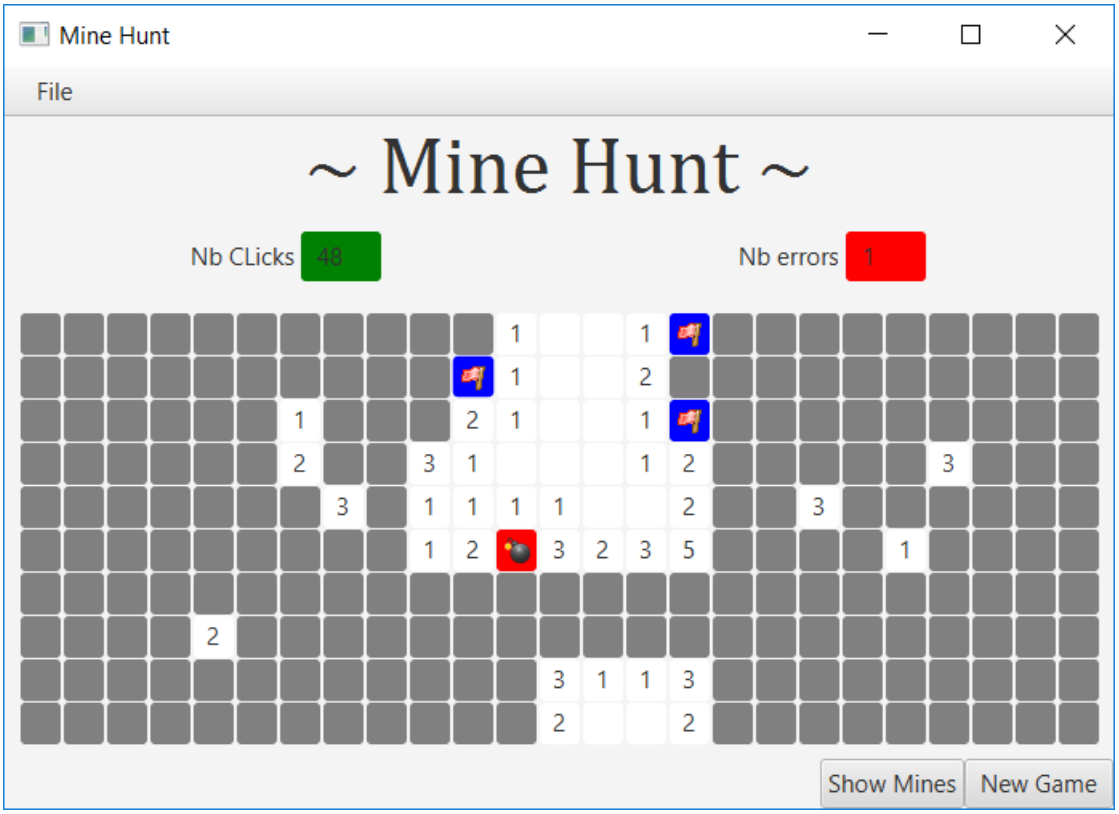

FIGURE 3 – Partie en cours

2.2 Graphes de scène

2.2.1 Vue principale

Voici le graphe de scène de la vue principale :

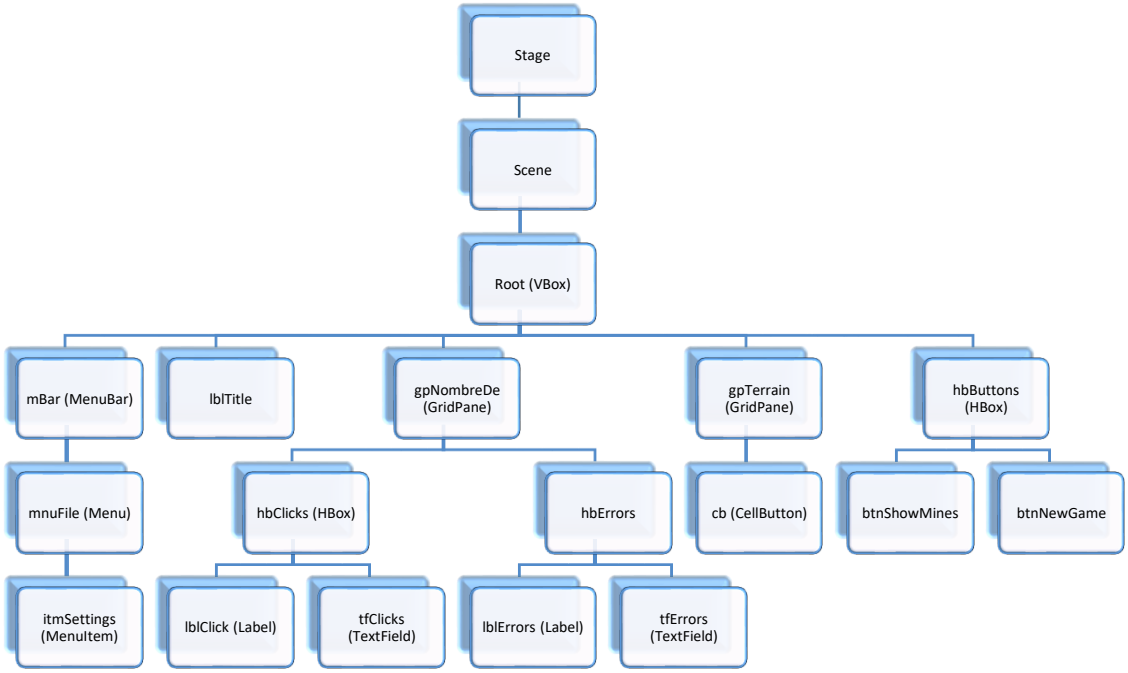

FIGURE 4 – Graphe de scène de la vue principale

Remarque : sur le schéma il n'y a qu'un seul *cb (CellButon*. Dans la réalité, il y a *hauteur* × *largeur* nombre de *Cellbuttons*.

2.2.2 Settings

Voici le graphe de scène de *Settings*

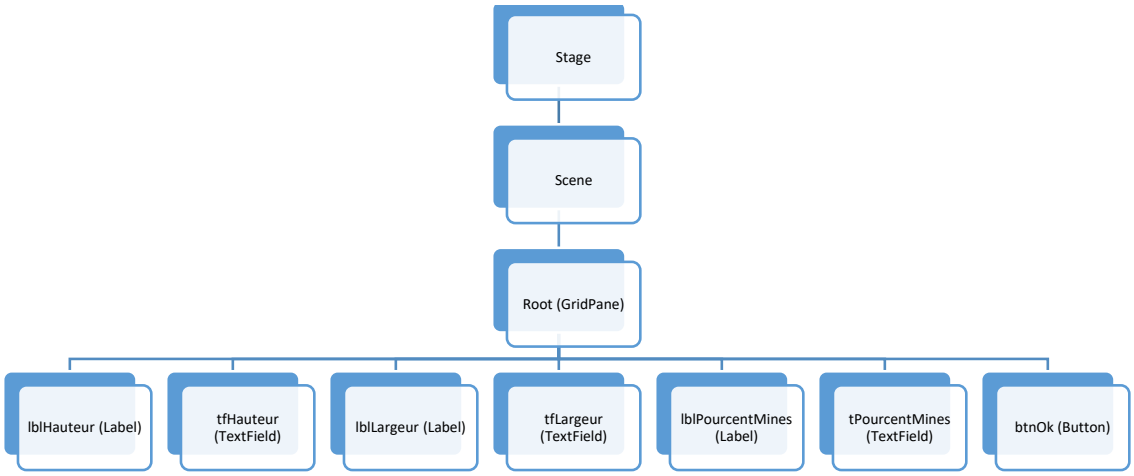

FIGURE 5 – Graphe de scène de *Settings*

2.3 Description des classes

- Cellbutton
- IMineHuntModel
- MineHuntController
- MineHuntModel

— MineHuntView
— NewGameController
— SettingController
— SettingView
— ShowMinesController
— TerrainController

Et la description de chacune d'entre elles :

CellButton C'est la classe que nous avons utilisée pour les mines, elle hérite de la classe *Button*. Nous lui avons ajouté les attributs *ligIndex* et *colIndex*, afin de pouvoir savoir sur quel bouton nous cliquons (parmi tous les boutons présent dans notre "terrain" de mines

IMineHuntModel C'est l'interface de notre modèle. Elle contient les méthodes que nous pensions utiliser pour notre modèle. Finalement, notre interface ne contenait pas toutes les méthodes nécessaires, nous avons donc du en rajouter dans notre modèle

MineHuntController Cette classe permet de faire le lien entre entre la vue principale et le modèle

MineHuntModel C'est le modèle, elle contient toute l'intelligence de notre programme

MineHuntView La fenêtre principale

NewGameController Le controller faisant le lien entre le bouton *New Game* et le modèle

SettingController Le controller faisant le lien entre la fenêtre *Setting* et le modèle

SettingView La fenêtre secondaire permettant d'afficher les paramètre pour la partie suivante

ShowMinesController Le controller faisant le lien entre le bouton *ShowMines* et le modèle

TerrainController Le controller faisant le lien entre les *CellButtons* et le modèle

2.4 Fonctionnalité implémentées

Voici la liste des éléments que nous avons implémenter dans notre application :
— Possibilité de définir la hauteur, la largeur et le pourcentage de mines du terrain par l'utilisateur
— Nouvelle partie
— Montrer les mines
— Placer des drapeaux
— Click automatique sur toutes les cases autour d'une case n'ayant pas de mine(s) autour
— Compteur de clicks
— Compteur d'erreurs

2.5 Code (fichiers *.java*)

Ici, nous verrons le code de chacune des classes et expliquerons plus en détail leurs méthodes.

2.5.1 Cellbutton.java

```
package s02;

import javafx.scene.control.Button;

public class CellButton extends Button {
  private int ligIndex;
  private int colIndex;

  public CellButton(int ligIndex, int colIndex) {
    super();
    this.ligIndex = ligIndex;
    this.colIndex = colIndex;
  }
```

```java
  /**
   * Retourne l'index de la ligne
   * @return
   */
  public int getLigIndex() {
    return ligIndex;
  }

  /**
   * Retourne l'index de la colonne
   * @return
   */
  public int getColIndex() {
    return colIndex;
  }
}
```

../CellButton.java

2.5.2 IMineHuntModel.java

```java
package s02;

public interface IMineHuntModel {
  int nbMinesAutour(int i, int j);
  boolean estCoinHautGauche(int i, int j);
  boolean estCoinHautDroite(int i, int j);
  boolean estCoinBasDroite(int i, int j);
  boolean estCoinBasGauche(int i, int j);
  boolean estBordHaut(int i, int j);
  boolean estBordDroite(int i, int j);
  boolean estBordBas(int i, int j);
  boolean estBordGauche(int i, int j);
  boolean estUneMine(int i, int j);
  void afficherTableau();
  void initialiser();
}
```

../IMineHuntModel.java

2.5.3 MineHuntController.java

```java
package s02;

import s02.MineHuntModel;
import s02.MineHuntView;

public class MineHuntController {
  private MineHuntModel model;
  private MineHuntView view;

  // CONSTRUCTEUR
  // -----------------------------------------------
  public MineHuntController(MineHuntModel model, MineHuntView view) {
    this.view = view;
    this.model = model;
  }

  public void initialize() {
    try {
      int hauteur = model.getTerrain().length;
      int largeur = model.getTerrain()[0].length;
      int pourcentMines = model.getPourcentMines();
      model.setTerrain(hauteur, largeur);
```

```
      model.setDejaClique(hauteur, largeur);
      model.setFlagged(hauteur, largeur);
      model.setPourcentMines(pourcentMines);
      model.setNbClicks(0);
      model.setNbErreurs(0);
      model.initialiser();
    } catch (IllegalArgumentException e) {
      e.printStackTrace();
    }

    view.creerTerrain(model.getNbLines(), model.getNbCol());

    System.out.println("Initialisation avec un terrain d'une largeur de " + model.
    getTerrain()[0].length
        + " * une hauteur de " + model.getTerrain().length + " mines " + "et avec
    un pourcentage de mines de "
        + model.getPourcentMines());
  }

  /**
   * Cette méthode sert à définir un terrain lors de la première initialisation
   * ensuite, l'utilisateur pourra définir cela lui-même
   */
  public void firstInit() {
    int hauteur = 10;
    int largeur = 10;
    int pourcentMines = 10;
    model.setTerrain(hauteur, largeur);
    model.setPourcentMines(pourcentMines);
    model.initialiser();
  }

}
```

../MineHuntController.java

2.5.4 MineHuntModel.java

Le modèle contient trois attributs principaux :

terrain Tableau 2D de booléens représentant toutes les cases. *true* si une bombe est présente, sinon *false*

dejaClique Tableau 2D de booléens représentant l'état actuel des cases. *true* si la case à déjà été cliquée, sinon *false*

flagged Tableau 2D de booléens représentant les drapeaux sur le terrain. *true* si un drapeau est présent sur la case, sinon *false*

```
package s02;

import java.util.Random;

public class MineHuntModel implements IMineHuntModel {
  private boolean[][] terrain;
  private boolean[][] dejaClique;
  private boolean[][] flagged;
  private int nbClicks;
  private int nbErreurs;
  private int pourcentDeMines;
  private int nbMines;

  public MineHuntModel() {
    terrain = null;
    dejaClique = null;
    flagged = null;
    nbClicks = 0;
    nbErreurs = 0;
```

```
20      this.pourcentDeMines = 0;
    }

    /**
     * Initialise le tableau de mines
25   */
    @Override
    public void initialiser() {
      nbMines = 0;
      // On initialise que si le tableau existe
30    if (terrain != null) {
        int ligIndex = getNbLines();
        int colIndex = getNbCol();
        // On parcourt toutes les cases une par une
        for (int i = 0; i < ligIndex; i++) {
35        for (int j = 0; j < colIndex; j++) {
            placerMineAleatoirement(i, j);
          }
        }
        afficherTableau();
40      System.out.println("Terrain initialisé");
      } else {
        System.out.println("Terrain non-initialisé, terrain = null");
      }
      System.out.println(nbMines + " mines présentes sur le terrain");
45  }

    /**
     * Crée un terrain
     * @param ligIndex
50   * @param colIndex
     */
    public void setTerrain(int ligIndex, int colIndex) {
      if (colIndex <= 0 || ligIndex <= 0)
        throw new IllegalArgumentException("Le tableau doit contenir au moins une
      case !");
55    terrain = new boolean[ligIndex][colIndex];
      System.out.print("Nouveau terrain créé");
    }

    /**
60   * Crée le tableau de cases dejaClique
     * @param ligIndex
     * @param colIndex
     */
    public void setDejaClique(int ligIndex, int colIndex) {
65    if (colIndex <= 0 || ligIndex <= 0)
        throw new IllegalArgumentException("Le tableau doit contenir au moins une
      case !");
      dejaClique = new boolean[ligIndex][colIndex];
    }

70  /**
     * Change la valeur d'une case dans le tableau dejaClique
     * @param ligIndex
     * @param colIndex
     * @param dejaClique true/false
75   */
    public void setUneCaseDejaClique(int ligIndex, int colIndex, boolean dejaClique)
      {
      this.dejaClique[ligIndex][colIndex] = dejaClique;
    }

80  /**
     * Crée le tableau de cases avec drapeau
     * @param ligIndex
     * @param colIndex
     */
85  public void setFlagged(int ligIndex, int colIndex) {
```

```
    if (ligIndex <= 0 || colIndex <= 0)
      throw new IllegalArgumentException("Le tableau doit contenir au moins une
    case !");
    flagged = new boolean[ligIndex][colIndex];
  }

  /**
   * Modifie une case dans le tableau flagged
   * @param ligIndex
   * @param colIndex
   * @param flagged true/false
   */
  public void setUneCaseFlagged(int ligIndex, int colIndex, boolean flagged) {
    this.flagged[ligIndex][colIndex] = flagged;
  }

  /**
   * Modifie la variable nbClick
   * @param nbClicks
   */
  public void setNbClicks(int nbClicks) {
    if (nbClicks < 0)
      throw new IllegalArgumentException("Le nombre de clicks ne peut pas être
    négatif");
    this.nbClicks = nbClicks;
  }

  /**
   * Modiife la variable nbErreurs
   * @param nbErreurs
   */
  public void setNbErreurs(int nbErreurs) {
    if (nbErreurs < 0)
      throw new IllegalArgumentException("Le nombre d'erreurs ne peut pas être
    négatif");
    this.nbErreurs = nbErreurs;
  }

  /**
   * Modifie la variable pourcentMines
   * @param pourcentage
   */
  public void setPourcentMines(int pourcentage) {
    if (pourcentage < 0 || pourcentage > 100)
      throw new IllegalArgumentException("Le pourcentage doit être compris entre 1
    et 100");
    pourcentDeMines = pourcentage;
  }

  /**
   * Retourne le terrain de mines
   * @return
   */
  public boolean[][] getTerrain() {
    return terrain;
  }

  /**
   * Retourne le nombre de lignes du terrain
   * @return
   */
  public int getNbLines() {
    return terrain.length;
  }

  /**
   * Retourne le nombre de colonnes du terrain
   * @return
   */
```

```
  public int getNbCol() {
    return terrain[0].length;
  }

  /**
   * Retourne le tableau de cases dejaClique
   * @return
   */
  public boolean[][] getDejaClique() {
    return dejaClique;
  }

  /**
   * Retourne la valeur d'une case dans le tableau dejaClique
   * @param ligIndex
   * @param colIndex
   * @return
   */
  public boolean getUneCaseDejaClique(int ligIndex, int colIndex) {
    return dejaClique[ligIndex][colIndex];
  }

  /**
   * Retourne la valeur d'une case dans le tableau flagged
   * @param ligIndex
   * @param colIndex
   * @return
   */
  public boolean getUneCaseFlagged(int ligIndex, int colIndex) {
    return flagged[ligIndex][colIndex];
  }

  /**
   * Retourne le tableau 2D flagged
   * @return
   */
  public boolean[][] getFlagged() {
    return flagged;
  }

  /**
   * Retourne le pourcentage de mines
   * @return
   */
  public int getPourcentMines() {
    return pourcentDeMines;
  }

  /**
   * Retourne le nombre de clicks
   * @return
   */
  public int getNbClicks() {
    return nbClicks;
  }

  /**
   * Retourne le nombre d'erreurs
   * @return
   */
  public int getNbErrors() {
    return nbErreurs;
  }

  /**
   * Place (peut-être) une mine avev pourcentMine pourcent de chance
   * @param i
   * @param j
   */
```

```
220   private void placerMineAleatoirement(int i, int j) {
        Random r = new Random();
        int rand = r.nextInt(100);
        if (rand < pourcentDeMines) {
          terrain[i][j] = true;
225       System.out.print("une mine en plus");
          nbMines++;
        }
      }

230   /**
       * Retourne true si la partie est terminée, sinon false
       * @return true/false
       */
      public boolean estTermine() {
235     // Si le nombre de clicks = nombre de cases sans bombes
        // alors la partie est terminée
        return (nbClicks >= (terrain.length * terrain[0].length) - nbMines);

      }
240
      @Override
      public boolean estBordHaut(int i, int j) {
        return i == 0;
      }
245
      @Override
      public boolean estBordDroite(int i, int j) {
        return j == terrain[0].length - 1;
      }
250
      @Override
      public boolean estBordBas(int i, int j) {
        return i == terrain.length - 1;
      }
255
      @Override
      public boolean estBordGauche(int i, int j) {
        return j == 0;
      }
260
      @Override
      public boolean estCoinHautGauche(int i, int j) {
        return i == 0 && j == 0;
      }
265
      @Override
      public boolean estCoinHautDroite(int i, int j) {
        return i == 0 && j == terrain[0].length - 1;
      }
270
      @Override
      public boolean estCoinBasDroite(int i, int j) {
        return i == terrain.length - 1 && j == terrain[0].length - 1;
      }
275
      @Override
      public boolean estCoinBasGauche(int i, int j) {
        return i == terrain.length - 1 && j == 0;
      }
280
      /**
       * Retourne un tableau contenant les index des cases autour de la case dont
       * les index sont donnés en paramètre
       * @param i
285    * @param j
       * @return
       */
      public int[][] casesAutour(int i, int j) {
```

```
    int[][] r = new int[8][2];
    // Initialisation avec des valeurs qu'on ne peut
    // pas trouver dans le champs de mine
    for (int k = 0; k < r.length; k++) {
      for (int l = 0; l < r[0].length; l++) {
        r[k][l] = -1;
      }
    }
    try { // haut gauche
      r[0][0] = i - 1;
      r[0][1] = j - 1;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // haut
      r[1][0] = i - 1;
      r[1][1] = j;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // haut droite
      r[2][0] = i - 1;
      r[2][1] = j + 1;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // droite
      r[3][0] = i;
      r[3][1] = j + 1;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // bas droite
      r[4][0] = i + 1;
      r[4][1] = j + 1;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // bas
      r[5][0] = i + 1;
      r[5][1] = j;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // bas gauche
      r[6][0] = i + 1;
      r[6][1] = j - 1;
    } catch (Exception e) {
      // Ne rien faire
    }
    try { // gauche
      r[7][0] = i;
      r[7][1] = j - 1;
    } catch (Exception e) {
      // Ne rien faire
    }
    return r;
  }

  @Override
  public boolean estUneMine(int i, int j) {
    return terrain[i][j];
  }

  /**
   * Compte le nombre de mines autour de la mine dont l'index est donné en
    paramètre
   */
  @Override
```

```
  public int nbMinesAutour(int i, int j) {
    int nbMine = 0;
    if (estCoinHautGauche(i, j)) {
      nbMine = estUneMine(i, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j) ? nbMine + 1 : nbMine;
    } else if (estCoinHautDroite(i, j)) {
      nbMine = estUneMine(i, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j) ? nbMine + 1 : nbMine;
    } else if (estCoinBasDroite(i, j)) {
      nbMine = estUneMine(i - 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j - 1) ? nbMine + 1 : nbMine;
    } else if (estCoinBasGauche(i, j)) {
      nbMine = estUneMine(i - 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j + 1) ? nbMine + 1 : nbMine;
    } else if (estBordHaut(i, j)) {
      nbMine = estUneMine(i, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j - 1) ? nbMine + 1 : nbMine;
    } else if (estBordDroite(i, j)) {
      nbMine = estUneMine(i, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j) ? nbMine + 1 : nbMine;
    } else if (estBordBas(i, j)) {
      nbMine = estUneMine(i - 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j + 1) ? nbMine + 1 : nbMine;
    } else if (estBordGauche(i, j)) {
      nbMine = estUneMine(i - 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j + 1) ? nbMine + 1 : nbMine;
    } else {
      nbMine = estUneMine(i - 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j + 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i, j - 1) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i + 1, j) ? nbMine + 1 : nbMine;
      nbMine = estUneMine(i - 1, j - 1) ? nbMine + 1 : nbMine;
    }
    return nbMine;
  }

  /**
   * Affiche dans la console le terrain de mines
   */
  @Override
  public void afficherTableau() {
    String c;
    for (int i = 0; i < terrain.length; i++) {
      for (int j = 0; j < terrain[0].length; j++) {
        c = terrain[i][j] == false ? Integer.toString(nbMinesAutour(i, j)) : " ";
        System.out.print("[" + c + "]");
      }
      System.out.println();
    }
  }
```

```
  /**
   * Affiche le tableau de cases dejaClique
   */
  public void afficherDejaClique() {
    String c;
    for (int i = 0; i < dejaClique.length; i++) {
      for (int j = 0; j < dejaClique[0].length; j++) {
        c = dejaClique[i][j] == false ? " " : "D";
        // c = dejaClique[i][j];
        System.out.print("[" + c + "]");
      }
      System.out.println();
    }
  }
}
```

../MineHuntModel.java

2.5.5 MineHuntView.java

```
package s02;

import s02.MineHuntController;
import s02.CellButton;
import s02.SettingsView;
import javafx.geometry.*;
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.ImageView;
import javafx.scene.layout.*;
import javafx.scene.text.Font;


public class MineHuntView extends Application {
  private SettingsView sView;
  private SettingController sController;
  private MineHuntModel mhModel;
  private MineHuntController mhController;
  private NewGameController ngController;
  private TerrainController tController;
  private ShowMinesController smController;
  private Scene scene;
  private VBox root;
  private Menu mnuFile;
  private TextField tfClicks, tfErrors;
  private Button btnShowMines, btnNewGame;
  private static final String IMG_BOMB = "file:src/s02/resources/bomb.png";
  private static final String IMG_FLAG = "file:src/s02/resources/flag.png";
  private GridPane gpTerrain = new GridPane();
  private CellButton[][] cellButtons;

  public void init() {
    mhModel = new MineHuntModel();
    gpTerrain = new GridPane();
    tController = new TerrainController(mhModel, this);
    mhController = new MineHuntController(mhModel, this);
    mhController.firstInit(); // pour générer un terrain quand on lance le
                  // programme
    mhController.initialize();
    sView = new SettingsView(mhModel, sController);
    sController = new SettingController(mhModel, sView, this);
    sView.setController(sController);
    System.out.println("Fin initialisation");
```

```
}

// MÉTHODE PRINCIPALE
// ------------------------------------------------
@Override
public void start(Stage primaryStage) {
  try {
    // création du container principal
    root = root();
    Label lblTitle = lblTitle();
    MenuBar mBar = new MenuBar();
    mnuFile = new Menu("File");
    MenuItem itmSettings = new MenuItem("Settings");
    itmSettings.setOnAction(sController);
    mnuFile.getItems().add(itmSettings);
    mBar.getMenus().add(mnuFile);
    tfClicks = new TextField();
    tfErrors = new TextField();
    GridPane gpNombreDe = gpNombreDe();
    smController = new ShowMinesController(mhModel, this);
    HBox hbButtons = hbButtons();
    createControllerNewGame();
    root.getChildren().addAll(mBar, lblTitle, gpNombreDe, gpTerrain, hbButtons);

    scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.setResizable(true);
    primaryStage.setTitle("Mine Hunt");
    primaryStage.show();

  } catch (Exception e) {
    e.printStackTrace();
  }

}

/**
 * Crée le controleur du bouton New Game
 */
public void createControllerNewGame() {
  ngController = new NewGameController(mhModel, this, mhController, smController)
  ;
  btnNewGame.setOnAction(ngController);
}

/**
 * Met à jour le TextField nbClicks
 * @param nbClicks
 */
public void updateNbClicks(int nbClicks) {
  tfClicks.setText(Integer.toString(nbClicks));
}

/**
 * Met à jour le TextField nbErrors
 * @param nbErrors
 */
public void updateNbErrors(int nbErrors) {
  tfErrors.setText(Integer.toString(nbErrors));
}

/**
 * Place tous les CellButtons en fonction de la hauteur et
 * et de la largeur en paramètre
 * @param ligIndex
 * @param colIndex
 */
public void creerTerrain(int ligIndex, int colIndex) {
  // Tableau contenant les boutons
```

```
115     cellButtons = new CellButton[ligIndex][colIndex];
        // Controleur contrôlant l'appui sur les boutons
        gpTerrain.setHgap(2);
        gpTerrain.setVgap(2);
        gpTerrain.setPadding(new Insets(10, 10, 10, 10));
120     // Placement de tous les boutons
        // hauteur -> ligIndex
        // largeur -> colIndex
        for (int i = 0; i < ligIndex; i++) {
          for (int j = 0; j < colIndex; j++) {
125         CellButton cb = new CellButton(i, j);
            cb.setMinSize(25, 25);
            cb.setMaxSize(25, 25);
            cb.setStyle("-fx-font-size: 12px; -fx-background-color: grey;" + "-fx-
      border-radius: null;");
            cb.setPadding(new Insets(1, 1, 1, 1));
130         cb.setOnMouseClicked(event -> {
              tController.gererClick(event);
            });

            // Ajout du bouton dans le tableau
135         cellButtons[cb.getLigIndex()][cb.getColIndex()] = cb;
            // Ajout dans la gridPane (Affichage graphique)
            gpTerrain.add(cb, j, i); // Cette méthode fonctionne à l'inverse
                        // de ma logique
          }
140     }
    }

    private VBox root() {
      root = new VBox();
145   root.setAlignment(Pos.TOP_CENTER);
      // root.setPadding(new Insets(10, 10, 10, 10));
      return root;
    }

150 private Label lblTitle() {
      Label lblTitle = new Label("~ Mine Hunt ~");
      lblTitle.setFont(Font.font("Cambria", 50));
      return lblTitle;
    }
155
    private GridPane gpNombreDe() {
      // GridPane Numbers of clicks / errors
      GridPane gpNombreDe = new GridPane();
      ColumnConstraints column1 = new ColumnConstraints();
160   ColumnConstraints column2 = new ColumnConstraints();
      column1.setPercentWidth(50);
      column2.setPercentWidth(50);
      gpNombreDe.setHgap(10);
      gpNombreDe.setPadding(new Insets(10, 10, 10, 10));
165   HBox hbClicks = new HBox();
      hbClicks.setAlignment(Pos.CENTER);
      Label lblClicks = new Label("Nb CLicks ");
      tfClicks.setEditable(false);
      tfClicks.setText("0");
170   tfClicks.setMaxSize(50, 25);
      tfClicks.setStyle("-fx-background-color: green;");
      hbClicks.getChildren().addAll(lblClicks, tfClicks);

      HBox hbErrors = new HBox();
175   hbErrors.setAlignment(Pos.CENTER);
      Label lblErrors = new Label("Nb errors ");
      tfErrors = new TextField();
      tfErrors.setEditable(false);

180   tfErrors.setText("0");
      tfErrors.setMaxSize(50, 10);
      tfErrors.setStyle("-fx-background-color: red;");
```

```
    hbErrors.getChildren().addAll(lblErrors, tfErrors);

    gpNombreDe.getColumnConstraints().addAll(column1, column2);
    gpNombreDe.add(hbClicks, 0, 0);
    gpNombreDe.add(hbErrors, 1, 0);

    return gpNombreDe;
  }

  private HBox hbButtons() {
    HBox hbButtons = new HBox();
    hbButtons.setAlignment(Pos.BOTTOM_RIGHT);
    btnShowMines = new Button("Show Mines");
    btnShowMines.setStyle("-fx-padding: 5px;");
    btnShowMines.setOnAction(smController);
    btnNewGame = new Button("New Game");

    hbButtons.getChildren().addAll(btnShowMines, btnNewGame);
    return hbButtons;
  }

  /**
   * Met le backgrount du CellButton en rouge (bombe)
   * @param btn
   */
  public void colorierEnRouge(CellButton btn) {
    btn.setStyle("-fx-background-color: red;");
    btn.setGraphic(new ImageView(IMG_BOMB));
  }

  /**
   * Affiche le nombre de mines autour sur le bouton
   * @param btn
   * @param nbMinesAutour
   */
  public void afficherNbMinesAutour(CellButton btn, int nbMinesAutour) {
    btn.setStyle("-fx-background-color: white;");
    if (nbMinesAutour > 0)
      btn.setText(Integer.toString(nbMinesAutour));
  }

  /**
   * Place un drapeau sur le bouton
   * @param btn
   */
  public void mettreUnDrapeau(CellButton btn) {
    btn.setStyle("-fx-background-color: blue;");
    btn.setGraphic(new ImageView(IMG_FLAG));
  }

  /**
   * Enlève un drapeau sur le bouton
   * @param btn
   */
  public void enleverDrapeau(CellButton btn) {
    btn.setStyle("-fx-background-color: grey;");
    btn.setGraphic(null);
  }

  /**
   * Ouvre toutes les cases présentent dans la tableau casesAutour
   * @param terrain
   * @param casesAutour
   */
  public void ouvrirToutesLesCasesAutour(boolean[][] terrain, int[][] casesAutour)
    {
    for (int i = 0; i < casesAutour.length; i++) {
      if ((casesAutour[i][0] >= 0 && casesAutour[i][0] < terrain.length)
          && (casesAutour[i][1] >= 0 && casesAutour[i][1] < terrain[0].length)) {
```

```
        // Simuler un clic sur les cases autour
        tController.gererBouton(cellButtons[casesAutour[i][0]][casesAutour[i][1]]);
      }
    }
  }

  /**
   * Montre toutes les mines du terrain
   * @param mines
   * @param dejaClique
   */
  public void montrerMines(boolean[][] mines, boolean[][] dejaClique) {
    // tableau mines et cellButtons ont la même forme
    for (int i = 0; i < mines.length; i++) {
      for (int j = 0; j < mines[0].length; j++) {
        if (mines[i][j] == true && dejaClique[i][j] == false) {
          cellButtons[i][j].setStyle("-fx-background-color: darkred");
        }
      }
    }
  }

  /**
   * Cache toutes les mines du terrain
   * @param mines
   * @param dejaClique
   */
  public void cacherMines(boolean[][] mines, boolean[][] dejaClique) {
    for (int i = 0; i < mines.length; i++) {
      for (int j = 0; j < mines[0].length; j++) {
        if (mines[i][j] == true && dejaClique[i][j] == false) {
          cellButtons[i][j].setStyle("-fx-background-color: grey");
        }
      }
    }
  }

  /**
   * Change le text du bouton ShowMines
   */
  public void switchLabelBtnShowMines() {
    if (btnShowMines.getText() == "Show Mines")
      btnShowMines.setText("Hide Mines");
    else
      btnShowMines.setText("Show Mines");
  }

  /**
   * Termine la partie et affiche le nombre d'erreurs
   * @param nbErrors
   */
  public void terminerPartie(int nbErrors) {
    Alert dialog;
    if (nbErrors == 0) {
      dialog = new Alert(AlertType.INFORMATION);
      dialog.setContentText("Congratulations !\nCurrent game ended successfully (no
     error)");
    } else {
      dialog = new Alert(AlertType.WARNING);
      dialog.setContentText("Current game ended with " + nbErrors + " errors");
    }
    dialog.setTitle("Mine Hunt - GameOver");
    dialog.setHeaderText("MineHunt");
    dialog.showAndWait();
  }

  /**
   * Met le texte du bouton ShowMine dans son état initial
   */
```

```
  public void reinitLabelBtnShowMines() {
320   btnShowMines.setText("Show Mines");
  }

  /**
   * Supprime les éléments du gridpane Terrain
325  */
  public void resetTerrain() {
    gpTerrain.getChildren().clear();
  }

330 }
```

../MineHuntView.java

2.5.6 NewGameController.java

```
package s02;

import s02.MineHuntController;
import javafx.event.ActionEvent;
5 import javafx.event.EventHandler;
import javafx.scene.control.Button;

public class NewGameController implements EventHandler<ActionEvent> {
  private MineHuntModel model;
10 private MineHuntView view;
  private MineHuntController mhController;
  private ShowMinesController smController;

  public NewGameController(MineHuntModel model, MineHuntView view,
    MineHuntController mhController,
15    ShowMinesController smController) {
    this.model = model;
    this.view = view;
    this.mhController = mhController;
    this.smController = smController;
20 }

  @Override
  /**
   * Méthode appelée lorsque "New Game" est pressé
25  */
  public void handle(ActionEvent event) {
    view.resetTerrain();
    mhController.initialize();
    view.updateNbClicks(model.getNbClicks());
30  view.updateNbErrors(model.getNbErrors());
    smController.setMineShowed(false);
    view.reinitLabelBtnShowMines();
  }
}
```

../NewGameController.java

2.5.7 SettingController.java

```
package s02;

import javafx.event.ActionEvent;
import javafx.event.EventHandler;
5

public class SettingController implements EventHandler<ActionEvent> {
  private MineHuntModel model;
  private SettingsView view;
```

```
  public SettingController(MineHuntModel model, SettingsView view, MineHuntView
    mainView) {
    this.model = model;
    this.view = view;
  }

  @Override
  /**
   * Lorsque le menu Setting est pressé
   */
  public void handle(ActionEvent event) {
    System.out.println("JEAN");
    view.CreateSettingsView();
  }

  /**
   * Lorsque le bouton "Ok" de la vue Seting est pressé
   */
  public void btnOk() {
    int hauteur = view.getHauteur();
    int largeur = view.getLargeur();
    int pourcentMines = view.getPourcentMines();
    model.setTerrain(hauteur, largeur);
    model.setPourcentMines(pourcentMines);
    view.close();
  }


}
```

../SettingController.java

2.5.8 SettingsView.java

```
package s02;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class SettingsView {
  private MineHuntModel model;
  private SettingController controller;
  private TextField tfHauteur;
  private TextField tfLargeur;
  private TextField tfPourcentMines;
  private Stage subStage;

  public SettingsView(MineHuntModel model, SettingController controller) {
    this.model = model;
    this.controller = controller;
  }

  public void CreateSettingsView()
    {
        subStage = new Stage();
        subStage.setTitle("Settings");

        GridPane root = new GridPane();
```

```
        root.setAlignment(Pos.CENTER);
        root.setPadding(new Insets(10,10,10,10));
        ColumnConstraints cc = new ColumnConstraints();
        cc.setMinWidth(100);
        Scene scene = new Scene(root);
        Label lblHauteur = new Label("Hauteur ");
        Label lblLargeur = new Label("Largeur ");
        Label lblPourcentMines = new Label("Pourcentage de mines ");

        tfHauteur = new TextField();
        tfLargeur = new TextField();
        tfPourcentMines = new TextField();
        Button btnOk = new Button("OK");
        btnOk.setMinWidth(200);
        btnOk.setOnAction((e)->{controller.btnOk();});
        root.getColumnConstraints().addAll(cc);
        root.add(lblHauteur, 0,0);
        root.add(lblLargeur, 0,1);
        root.add(lblPourcentMines, 0,2);
        root.add(tfHauteur, 1, 0);
        root.add(tfLargeur, 1, 1);
        root.add(tfPourcentMines, 1, 2);
        root.add(btnOk, 1, 3);
        subStage.setScene(scene);
        subStage.show();
    }

  public void setController(SettingController controller) {
    this.controller = controller;
  }

  /**
   * Retourne le contenu du champ texte "hauteur"
   * @return
   */
  public int getHauteur() {
    return Integer.parseInt((tfHauteur.getText()));
  }

  /**
   * Retourne le contenu du champ texte "largeur"
   * @return
   */
  public int getLargeur() {
    return Integer.parseInt((tfLargeur.getText()));
  }

  /**
   * Retourne le contenu du champ texte "pourcentage de mines"
   * @return
   */
  public int getPourcentMines() {
    return Integer.parseInt((tfPourcentMines.getText()));
  }

  /**
   * Ferme la fenêtre
   */
  public void close() {
    subStage.close();
  }
}
```

../SettingsView.java

2.5.9 ShowMinesController.java

```java
package s02;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import s02.MineHuntModel;
import s02.MineHuntView;

public class ShowMinesController implements EventHandler<ActionEvent> {
  private MineHuntModel model;
  private MineHuntView view;
  private boolean minesShowed;

  public ShowMinesController(MineHuntModel model, MineHuntView view) {
    this.model = model;
    this.view = view;
    minesShowed = false;
  }

  @Override
  /**
   * Lorsque l'on clique sur "Show Mines"
   */
  public void handle(ActionEvent MouseEvent) {
    System.out.println("Clic sur show mines");
    // Si les mines ne sont pas montrées
    if (!minesShowed) {
      boolean[][] mines = model.getTerrain();
      boolean[][] dejaClique = model.getDejaClique();
      view.montrerMines(mines, dejaClique);
      view.switchLabelBtnShowMines();
      minesShowed = true;
    } else {
      // cacher les mines sur lesquelles on a pas encore cliqué
      boolean[][] mines = model.getTerrain();
      boolean[][] dejaClique = model.getDejaClique();
      view.cacherMines(mines, dejaClique);
      view.switchLabelBtnShowMines();
      minesShowed = false;
    }
  }

  /**
   * Réinitialise l'attribut "minesShowed" lors d'une nouvelle partie
   * @param val
   */
  public void setMineShowed(boolean val) {
    minesShowed = val;
  }
}
```

../ShowMinesController.java

2.5.10 TerrainController.java

```java
package s02;

import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;

public class TerrainController {
  private MineHuntModel model;
  private MineHuntView view;

  public TerrainController(MineHuntModel model, MineHuntView view) {
    this.view = view;
    this.model = model;
  }
```

```
15  /**
     * Lorsque l'on clique sur un bouton du champs de mines
     *
     * @param event
     */
20  public void gererClick(MouseEvent event) {
      try {
        CellButton btn = (CellButton) event.getSource();
        // Click gauche
        if (event.getButton() == MouseButton.PRIMARY) {
25        gererBouton(btn);
          if (model.estTermine()) {
            int nbErrors = model.getNbErrors();
            view.terminerPartie(nbErrors);
          }
30        // Click droit
        } else if (event.getButton() == MouseButton.SECONDARY) {
          int ligIndex = btn.getLigIndex();
          int colIndex = btn.getColIndex();
          if (model.getUneCaseDejaClique(ligIndex, colIndex) == false) {
35          if (model.getUneCaseFlagged(ligIndex, colIndex) == false) {
              model.setUneCaseFlagged(ligIndex, colIndex, true);
              view.mettreUnDrapeau(btn);
              System.out.println(model.getUneCaseFlagged(ligIndex, colIndex));
            } else {
40            model.setUneCaseFlagged(ligIndex, colIndex, false);
              view.enleverDrapeau(btn);
            }
            System.out.println("Click droit");
          }
45      }
      } catch (IllegalArgumentException e) {
        e.printStackTrace();
      }
    }
50
    /**
     * Gère le comportement de la case cliquée
     *
     * @param btn
55   */
    public void gererBouton(CellButton btn) {
      int ligIndex = btn.getLigIndex();
      int colIndex = btn.getColIndex();
      // si la case n'est pas flagged
60    if (model.getUneCaseFlagged(ligIndex, colIndex) == false) {
        if (model.getUneCaseDejaClique(ligIndex, colIndex) == true) {}
          // si la case n'a pas été cliquée
        else if (model.estUneMine(ligIndex, colIndex)) {
          model.setUneCaseDejaClique(ligIndex, colIndex, true);
65        model.setNbErreurs(model.getNbErrors() + 1);
          System.out.println("clic sur une bombe");
          view.colorierEnRouge(btn);
          view.updateNbErrors((model.getNbErrors()));
        } else {
70        model.setUneCaseDejaClique(ligIndex, colIndex, true);
          model.setNbClicks(model.getNbClicks() + 1);
          view.afficherNbMinesAutour(btn, model.nbMinesAutour(ligIndex, colIndex));
          view.updateNbClicks((model.getNbClicks()));
          // Ouvre les cases autour si pas de mines autour
75        if (model.nbMinesAutour(ligIndex, colIndex) == 0) {
            view.ouvrirToutesLesCasesAutour(model.getTerrain(), model.casesAutour(
    ligIndex, colIndex));
          }
        }
      } else {
80      System.out.println("Clic gauche sur une case avec drapeau, aucun effet");
      }
```

```
    }
85 }
```

../TerrainController.java

3 Conclusion

Le développement de cette application fut très enrichissant. D'une part, cela nous a permis de nous familiariser avec beaucoup de composants *JavaFX* et d'autre part avec l'architecture MVC et la gestion des événements.

3.1 Difficultés rencontrées

Voici les éléments qui nous ont posé problème :

— Au départ, c'était difficile de coder toutes les fonctionnalités en MVC. Nous ne pensions pas qu'il était nécessaire de mémoriser les drapeau dans le modèle par exemple. Nous étions parti sur l'idée que puisqu'il ne s'agissait que de l'affichage d'un drapeau, cela aurait pu être placé dans la vue

— Nous sommes restés bloqués un long moment à cause des tableaux en deux dimensions. Nous en manipulons tellement, que nous avions des méthodes avec lesquelles nous prenions la hauteur à la place de la largeur et vis-versa, et cela menait à des problèmes.
Le simple fait que nous avons codés tout nos tableaux en mettant *[ligIndex][colIndex]* nous a donné des difficultés avec l'ajout des *CellButtons* dans le *GridPane* parce que la méthode permettant l'ajout d'éléments dans le *GridPane* prend d'abord en paramètre l'index de la colonne et ensuite l'index de la ligne.

3.2 Idées d'amélioration

— Faire en sorte de limiter le nombre de mines pour que le programme de soit pas plus grand que l'écran

— Dans la fenêtre *Settings*, n'autoriser que les numéros dans les champs texte, ou remplacer les champs texte par des *Spinners*

— Sauvegarder la partie en cours