Interface Homme-Machine
TP S02

Date du travail :	21.04.2015	Etudiants :	Benoît Repond
Classe :	T-1a		Gilles Waeber

Application avec interface graphique

Table des matières

MineHunt

Notre application

Un aperçu de notre application se trouve à la page suivante mais un fichier jar de l'application est disponible si besoin à l'adresse :

http://gilles.waeber.home.hefr.ch/ihm/s02/MineHunt.jar

Extension de l'application

Etant donné que nous avons réalisé les trois exercices facultatifs (11 à 13), nous n'avons pas apporté d'énormes extensions à notre application. Nous avons toutefois apporté les 2 extensions suivantes :

- Lorsque l'utilisateur clique sur une bombe un son (bruit d'explosion) est joué en plus du changement visuel de la cellule. La fonction de son peut être désactivée au moyen d'un « CheckMenuItem » dans le menu des réglages.
- Pour les cellules ouvertes qui n'ont pas de mines voisines, l'affichage du nombre de mines voisines n'affiche rien mais change quand même de couleur (jaune).
  Cela ressemble plus au comportement du démineur classique et offre une meilleure visibilité de l'état du jeu (surtout s'il y a un grand nombre de cellules sans mines autour).

Difficultés rencontrées

La plus grosse difficulté a été le début du TP, surtout la création de l'interface (exercice 4). Etant donné qu'au début nous n'avions rien, il était assez difficile d'imaginer toute la structure interne du modèle, de la vue, du contrôleur et de leurs interactions.

Une difficulté mineure a été de faire fonctionne notre application en dehors de l'environnement de développement. Nous avons eu une difficulté avec le son (fichier .mp3 à l'intérieur du fichier jar).

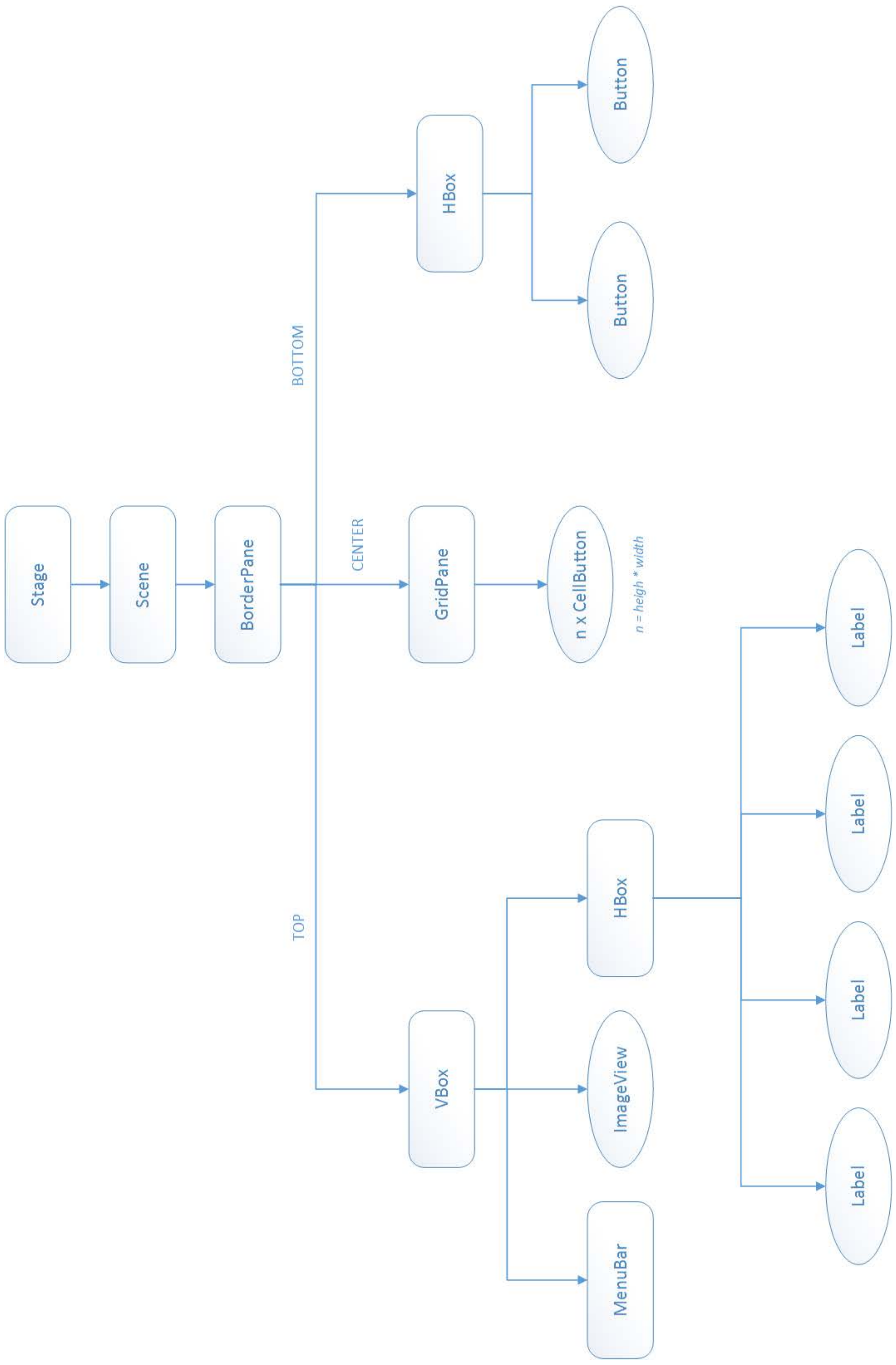
Problème : Nous avons constaté que sur Mac, la création d'une nouvelle partie prenait extrêmement beaucoup de temps alors que sous Windows elle est quasiment instantanée. Nous avons effectué des tests sur différentes machines avec différents OS, mais malheureusement nous n'avons pas réussi à trouver la cause de ce problème.

Points positifs et négatifs

Positif : Le projet est ludique mais il est surtout concret. Il permet de mettre en pratique tous les chapitres vus en cours sur JavaFX et également le cours de programmation.

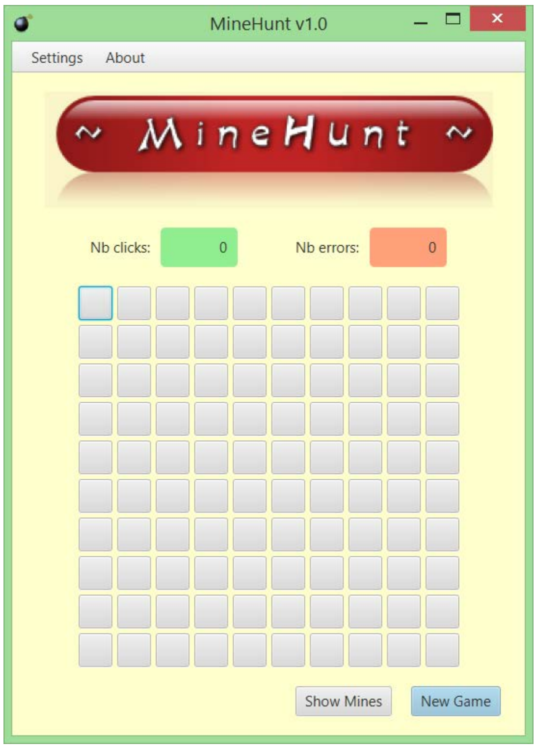
Négatif : Bien que la période de réalisation du TP soit assez grand, il nous a quand même fallu beaucoup de temps pour le réaliser.

Graphe de scène



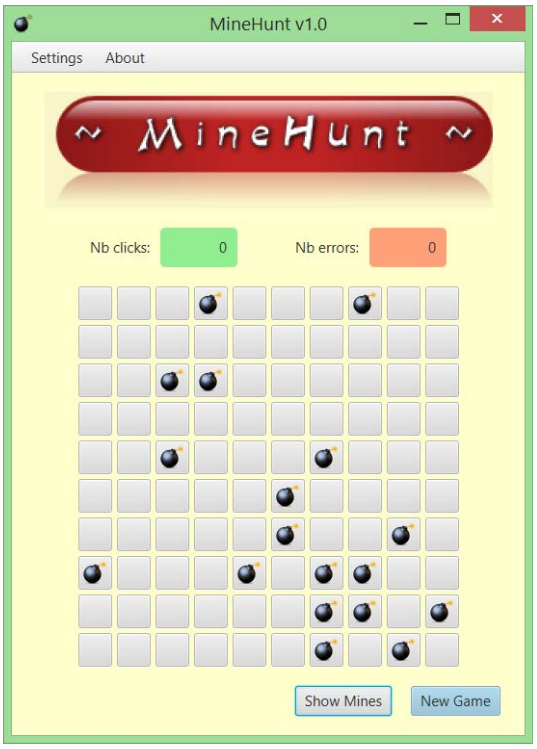
Aperçu de l'application

Aperçu de l'application au lancement



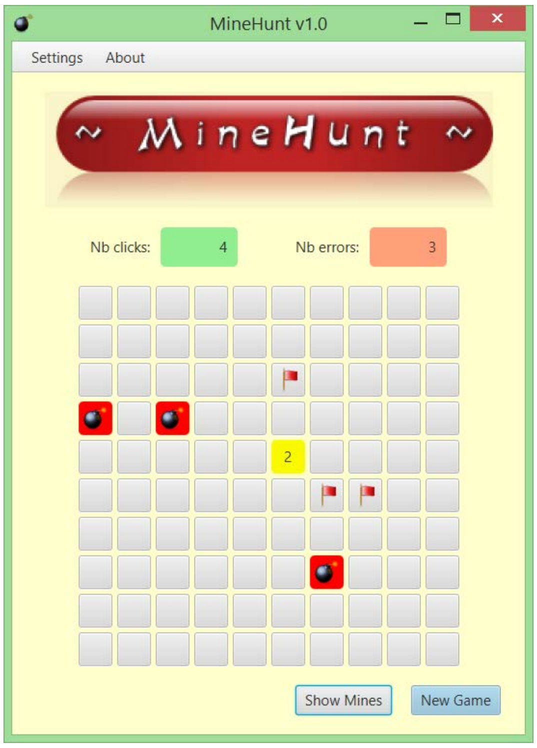
Nb clicks: 0 Nb errors: 0

Affichage des mines (mode tricheur)



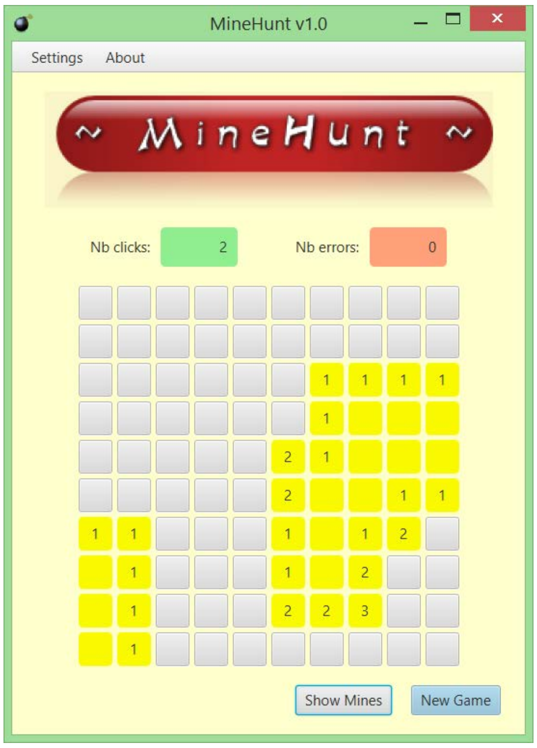
Nb clicks: 0 Nb errors: 0

Affichage des mines « explosées » et des drapeaux



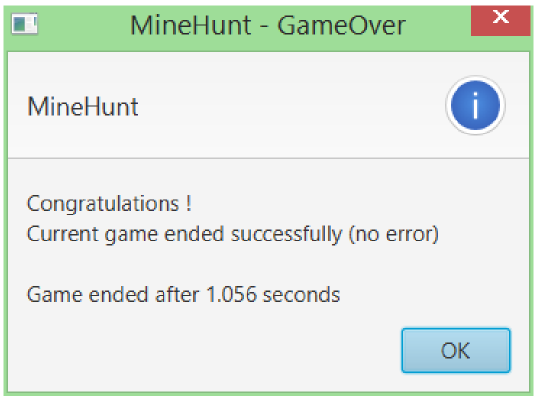
Nb clicks: 4 Nb errors: 3

Découvrement automatique des cellules voisines



Nb clicks: 2 Nb errors: 0

Fin de partie sans erreurs

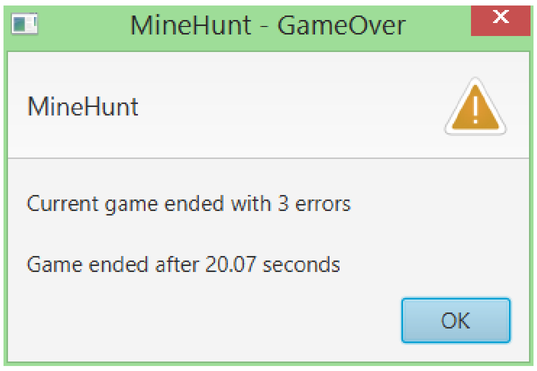


MineHunt

Congratulations !
Current game ended successfully (no error)

Game ended after 1.056 seconds

Fin de partie avec erreurs

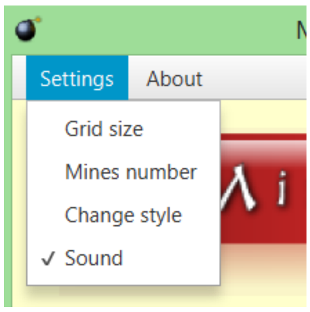


MineHunt

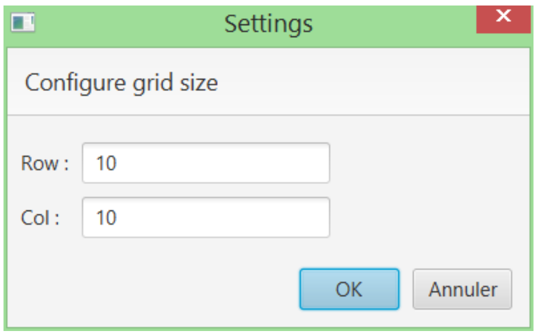
Current game ended with 3 errors

Game ended after 20.07 seconds

Menu des réglages



Configuration de la taille de la grille

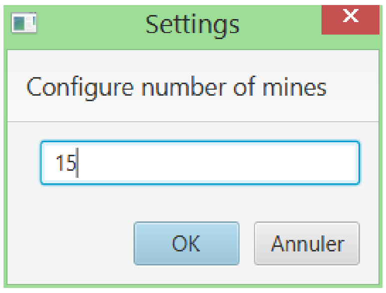


Configure grid size

Row : 10

Col : 10

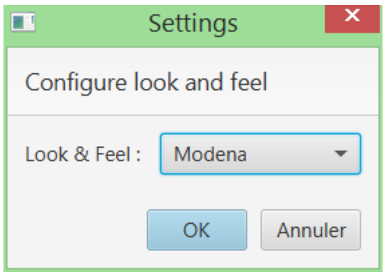
Configuration du nombre de mines



Configure number of mines

15

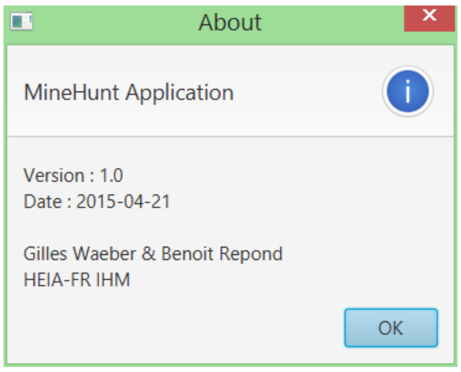
Configuration du style



Configure look and feel

Look & Feel : Modena

« A propos »



MineHunt Application

Version : 1.0
Date : 2015-04-21

Gilles Waeber & Benoit Repond
HEIA-FR IHM

Code Java

MineHunt

```java
package s02;

import javafx.application.Application;

public class MineHunt {

  public static void main(String[] args) {

    Application.launch(MineHuntView.class, args);

  }

}
```

MineHuntModel

```java
package s02;

import java.util.Random;

public class MineHuntModel implements IMineHuntModel{

  private int minesNb;
  private int clickNb;
  private int errorsNb;
  private Cell[][] grid;

  private boolean sound = true;
  private boolean firstRound = true;

  private int newMinesNb = 0;
  private int newGridCol = 0;
  private int newGridRow = 0;

  public MineHuntModel(int gridSizeCol, int gridSizeRow, int minesNb){
    this.minesNb = minesNb;
    buildGrid(gridSizeCol, gridSizeRow);
    initNewGame(minesNb);
  }

  public MineHuntModel(){
    this(10, 10, 18);
  }


  // Build model grid
  private void buildGrid(int gridSizeCol, int gridSizeRow){

    grid = new Cell[gridSizeCol][gridSizeRow];
    for(int i = 0; i < gridSizeCol; i++){
      for(int j = 0; j < gridSizeRow; j++){
        grid[i][j] = new Cell();
      }
    }

  }
```

```
@Override
public void initNewGame(int minesNb) {

  // Random mines placement in the grid
  randomMinesPlacement(minesNb);

  // Counters init
  clickNb = 0;
  errorsNb = 0;

  firstRound = true;

  setNewMinesNb(0);
  setNewGridCol(0);
  setNewGridRow(0);

}


void randomMinesPlacement(int minesNb){

  Random r = new Random();
  for(int i = 0; i < minesNb; i++){

    int col = r.nextInt(gridWidth());
    int row = r.nextInt(gridHeight());

    if(!grid[col][row].isMined()){
      grid[col][row].setMined(true);
    }else{
      i--;    // mine already exist here
    }

  }

}


@Override
public int gridWidth() {
  return grid.length;
}


@Override
public int gridHeight() {
  return grid[0].length;
}


@Override
public int mines() {
  return minesNb;
}


@Override
public int errors() {
  return errorsNb;
}

@Override
public int clicks(){
  return clickNb;
}
```

```
@Override
public int neighborMines(int row, int col) {

  int neighborMines = 0;

  for(int i = row-1; i <= row+1; i++){
    for(int j = col-1; j <= col+1; j++){

      if((i == row) && (j == col)){
        // no test in the cell itself
      }
      else if((i < 0) || (i > gridHeight()-1) || (j < 0) || (j > gridWidth()-1)){
        // no test in the borders
      }else{
        if(grid[j][i].isMined()){
          neighborMines++;
        }

      }

    }
  }

  return neighborMines;

}


@Override
public boolean isOpen(int row, int col) {
  return grid[col][row].isOpen();
}

@Override
public boolean isFlagged(int row, int col) {
  return grid[col][row].isFlaged();
}

@Override
public boolean isMined(int row, int col) {
  return grid[col][row].isMined();
}

public void setMined(int row, int col){
  grid[col][row].setMined(false);
}


@Override
public boolean isGameOver() {

  for(int j = 0; j < gridHeight(); j++){
    for(int i = 0; i < gridWidth(); i++){

      if(!isMined(j, i)){
        if(!isOpen(j, i)){
          return false;
        }
      }

    }
  }

  return true;
}
```

```java
@Override
public boolean open(int row, int col) {

  Cell cell = grid[col][row];

  if(!cell.isOpen()){

    cell.setOpen(true);
    cell.setFlaged(false);
    clickNb++;

    if(cell.isMined()){
      if(!firstRound){
        errorsNb++;
      }
      return true;
    }else{
      return false;
    }

  }

  return false;

}


@Override
public void setFlagState(int row, int col, boolean state) {
  grid[col][row].setFlaged(state);
}


@Override
public boolean openProximity(int row, int col) {

  Cell cell = grid[col][row];

  if(!cell.isOpen() && !cell.isMined()){

    cell.setOpen(true);
    cell.setFlaged(false);
    return true;

  }

  return false;

}


@Override
public void newModel(int gridSizeCol, int gridSizeRow, int minesNb){

  this.minesNb = minesNb;
  grid = null;
  buildGrid(gridSizeCol, gridSizeRow);
  initNewGame(minesNb);

}


boolean isSound() {
  return sound;
}
```

```
  void setSound(boolean sound) {
    this.sound = sound;
  }

  int getNewMinesNb() {
    return newMinesNb;
  }

  void setNewMinesNb(int newMinesNb) {
    this.newMinesNb = newMinesNb;
  }

  int getNewGridCol() {
    return newGridCol;
  }

  void setNewGridCol(int newGridCol) {
    this.newGridCol = newGridCol;
  }

  int getNewGridRow() {
    return newGridRow;
  }

  void setNewGridRow(int newGridRow) {
    this.newGridRow = newGridRow;
  }

  boolean isFirstRound(){
    return firstRound;
  }

  void setFirstRound(boolean firstRound){
    this.firstRound = firstRound;
  }


}
```

MineHuntView

```
package s02;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseButton;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


public class MineHuntView extends Application{

  private static final String TITLE  = "/resources/MineHunt_Title.png";
  private static final String ICON   = "/resources/MineHunt_Bomb.png";
```

```java
  private BorderPane    root            = new BorderPane();
  private VBox          title           = new VBox(20);
  private HBox          counters        = new HBox(10);
  private GridPane      grid            = new GridPane();
  private HBox          controls        = new HBox(20);
  private MenuBar       menu            = new MenuBar();

  private ImageView     titleImage      = new ImageView(TITLE);
  private Label         clicksLabel     = new Label("Nb clicks:");
  private Label         clicks          = new Label();
  private Label         errorsLabel     = new Label("Nb errors:");
  private Label         errors          = new Label();

  private Button        btnShowMines    = new Button("Show Mines");
  private Button        btnNewGame      = new Button("New Game");

  private MineHuntModel      model      = new MineHuntModel();
  private MineHuntController controller = new MineHuntController(model, this);
  private CellButton[][]     gridCellButton;

  private boolean       showMines;
  private double        startTime;


  @Override
  public void start(Stage mainStage) throws Exception {

    // Stage
    mainStage.setTitle("MineHunt v1.0");
    mainStage.setMinWidth(550);
    mainStage.setResizable(true);
    mainStage.getIcons().add(new Image(ICON));

    // BorderPane
    root.setTop(title);
    root.setCenter(grid);
    root.setBottom(controls);

    // Background
    Background background = new Background(new BackgroundFill(Color.rgb(255, 255, 204),
CornerRadii.EMPTY, null));
    root.setBackground(background);



    // -------------------------------------------------------




    // Menu about
    Menu menuAbout = new Menu();
    Label lblMenuAbout = new Label("About");
    menuAbout.setGraphic(lblMenuAbout);

    lblMenuAbout.setOnMouseClicked(event-> {

      Alert dialog= new Alert(AlertType.INFORMATION);
      dialog.setTitle("About");
      dialog.setHeaderText("MineHunt Application");
      dialog.setContentText("Version : 1.0\n" + "Date : 2015-04-21\n\n" + "Gilles Waeber &
Benoit Repond\n" + "HEIA-FR IHM");
      dialog.showAndWait();

    });
    // Menu settings
    Menu menuSettings = new Menu("Settings");
    MenuItem setGridSize = new MenuItem("Grid size");
    MenuItem setMinesNb = new MenuItem("Mines number");
    MenuItem setStyle = new MenuItem("Change style");
    CheckMenuItem setSound = new CheckMenuItem("Sound");
    setSound.setSelected(true);
    menuSettings.getItems().addAll(setGridSize, setMinesNb, setStyle, setSound);

    setGridSize.setOnAction(event-> {
      controller.settingsGridSize();
    });

    setMinesNb.setOnAction(event-> {
      controller.settingsMines();
    });

    setStyle.setOnAction(event -> {
      controller.settingsStyle();
    });

    setSound.setOnAction(event-> {
      model.setSound(!model.isSound());
    });


    // Menu
    menu.getMenus().addAll(menuSettings, menuAbout);
    title.getChildren().add(menu);


    // -------------------------------------------------------


    // Title image
    title.setAlignment(Pos.CENTER);
    title.getChildren().add(titleImage);

    // Number clicks
    clicks.setAlignment(Pos.BASELINE_RIGHT);
    clicks.setPrefWidth(80);
    clicks.setPadding(new Insets(10, 10, 10, 10));
    clicks.setBackground(new Background(new BackgroundFill(Color.LIGHTGREEN, new
CornerRadii(5), null)));

    // Number errors
    errors.setAlignment(Pos.BASELINE_RIGHT);
    errors.setPrefWidth(80);
    errors.setPadding(new Insets(10, 10, 10, 10));
    errors.setBackground(new Background(new BackgroundFill(Color.LIGHTSALMON, new
CornerRadii(5), null)));
    errorsLabel.setPadding(new Insets(0, 0, 0, 50));

    // Title
    counters.setAlignment(Pos.CENTER);
    counters.getChildren().add(clicksLabel);
    counters.getChildren().add(clicks);
    counters.getChildren().add(errorsLabel);
    counters.getChildren().add(errors);
    title.getChildren().add(counters);
    title.setPadding(new Insets(0, 0, 20, 0));


    // -------------------------------------------------------
    // Grid
    grid.setAlignment(Pos.CENTER);
    grid.setHgap(5);
    grid.setVgap(5);

    // Button grid creation
    gridButtonCreation();

    // -------------------------------------------------------


    // Controls
    btnNewGame.setDefaultButton(true);
    controls.setAlignment(Pos.BASELINE_RIGHT);
    controls.getChildren().add(btnShowMines);
    controls.getChildren().add(btnNewGame);
    controls.setPadding(new Insets(20, 25, 20, 10));

    // NewGame event handler
    btnNewGame.setOnAction(event -> {
      controller.newGame();
    });

    // ShowMines event handler
    btnShowMines.setOnAction(event -> {
      controller.showMines();
    });


    // -------------------------------------------------------

    // Misc
    setupVariables();

    // App launch
    mainStage.setScene(new Scene(root));
    mainStage.show();

  }


  public void gridButtonCreation() {

    gridCellButton = new CellButton[model.gridWidth()][model.gridHeight()];

    for(int j = 0; j < model.gridHeight(); j++){
      for(int i = 0; i < model.gridWidth(); i++){

        CellButton btn = new CellButton(j, i);
        btn.setMinSize(35, 35);

        gridCellButton[i][j] = btn;
        grid.add(gridCellButton[i][j], i, j);

        // Button event handler
        btn.setOnMouseClicked(event -> {

          // Check if game is not already ended
          if(!model.isGameOver()){

            // Handle left or right mouse click
            if(event.getButton() == MouseButton.PRIMARY){
              controller.leftClickAction(btn);
            }else if(event.getButton() == MouseButton.SECONDARY){
              controller.rightClickAction(btn);
            }
            // Check if game is over
            if(model.isGameOver()){
              controller.gameOver();
            }

          }

        });

      }
    }



  }



  public void setupVariables() {
    errors.setText("0");
    clicks.setText("0");
    startTime = System.currentTimeMillis();
    showMines = false;
  }


  // Change Look and Feel
  public void setLF(String style){

    switch (style) {
      case "Modena" :
        setUserAgentStylesheet(STYLESHEET_MODENA);
        break;
      case "Caspian" :
        setUserAgentStylesheet(STYLESHEET_CASPIAN);
        break;
      default :
        setUserAgentStylesheet(STYLESHEET_MODENA);
        break;
    }

  }


  // Getters and setters
  boolean isShowMines() {
    return showMines;
  }

  void setShowMines(boolean showMines) {
    this.showMines = showMines;
  }

  double getStartTime() {
    return startTime;
  }

  Label getClicks() {
    return clicks;
  }

  Label getErrors(){
    return errors;
  }
```

```
  GridPane getGrid(){
    return grid;
  }

  CellButton[][] getGridCellButton(){
    return gridCellButton;
  }


  /*
  @Override
  public void init() throws Exception{

  }*/


}
```

MineHuntController

```
package s02;

import java.util.Optional;

import javafx.geometry.Insets;
import javafx.scene.control.*;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonBar.ButtonData;
import javafx.scene.layout.GridPane;

public class MineHuntController {

  private MineHuntModel model;
  private MineHuntView   view;

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


  public void leftClickAction(CellButton btn) {

    // si rien ou flagged, on ouvre
    if(!model.isOpen(btn.getRow(), btn.getCol()) || model.isFlagged(btn.getRow(),
btn.getCol())){

      // ouvrir la cell
      if(model.open(btn.getRow(), btn.getCol())){

        // bombe

        // test if user is playing his first round
        if(!model.isFirstRound()){
          btn.presentationOpenedOnMine(model.isSound());
        }else{

          // remove mine from model
          model.setMined(btn.getRow(), btn.getCol());

          // insert 1 mine in another position in the model
          model.randomMinesPlacement(1);
          // open the button
          int neighborMines = model.neighborMines(btn.getRow(), btn.getCol());
          btn.presentationOpened(neighborMines);
          if(neighborMines == 0){
            openProximity(btn.getRow(), btn.getCol());
          }

          // if showMines hide and display mines again
          if(view.isShowMines()){
            showMines(); showMines();
          }

          // set first round to false
          model.setFirstRound(false);

        }

      }else{

        // test if user is playing his first round
        if(model.isFirstRound()){
          model.setFirstRound(false);
        }

        // pas de bombe, affiche neighborMines
        int neighborMines = model.neighborMines(btn.getRow(), btn.getCol());
        btn.presentationOpened(neighborMines);

        // test si pas de neighborMines, découvre les cases à zéro à proximité
        if(neighborMines == 0){
          openProximity(btn.getRow(), btn.getCol());
        }

      }
    }
    else if(model.isOpen(btn.getRow(), btn.getCol())){
      // la cellule est deja ouverte on ne fait rien
    }

    // Update menu counters
    view.getClicks().setText(Integer.toString(model.clicks()));
    view.getErrors().setText(Integer.toString(model.errors()));

  }




  // Ouvre les mines à proximité d'une mine
  private void openProximity(int row, int col) {


    for(int i = row-1; i <= row+1; i++){
      for(int j = col-1; j <= col+1; j++){

        if((i == row) && (j == col)){
          // pas de test de la case elle-même
        }
        else if((i < 0) || (i > model.gridHeight()-1) || (j < 0) || (j > model.gridWidth()-
1)){
          // pas de test dans les bordures
        }else{

          // si la mine à proximité a neighborMines à 0
          if(model.openProximity(i, j)){
            // ouverture de la case
            int neighborMines = model.neighborMines(i, j);

            // open button
            view.getGridCellButton()[j][i].presentationOpened(neighborMines);

            // appel récursif
            if(neighborMines == 0){
              openProximity(i, j);
            }

          }

        }

      }
    }

  }



  public void rightClickAction(CellButton btn) {

    // flag
    if(!model.isOpen(btn.getRow(), btn.getCol()) && !model.isFlagged(btn.getRow(),
btn.getCol())){

      btn.presentationFlagged();
      model.setFlagState(btn.getRow(), btn.getCol(), true);

    }else if(model.isFlagged(btn.getRow(), btn.getCol())){
      btn.presentationClosed();
      model.setFlagState(btn.getRow(), btn.getCol(), false);

    }

  }



  public void showMines(){

    for(int j = 0; j < model.gridHeight(); j++){
      for(int i = 0; i < model.gridWidth(); i++){
        if(model.isMined(j, i)){

          // get CellButton from position in CellButton grid
          CellButton minedButton = view.getGridCellButton()[i][j];

          if(!view.isShowMines()){
            // modify button (display mines)
            minedButton.presentationShowMine();

          }else{
            // modify button (hide unopened mines only if not opened)
            if(!model.isOpen(j, i)){

              if(model.isFlagged(j, i)){
                minedButton.presentationFlagged();             // button was flagged before
              }else{
                minedButton.presentationClosed();              // button was opened before
              }
            }

          }

        }
      }
    }

    view.setShowMines(!view.isShowMines());

  }

  public void gameOver(){

    int errors = model.errors();

    if(errors > 0){
      gameOverAlert(AlertType.WARNING, false, errors);                           // Game
lost
    }else{
      gameOverAlert(AlertType.INFORMATION, true, 0);                             // Game win
    }

  }

  private void gameOverAlert(AlertType alertType, boolean win, int errors){

    double endTime = System.currentTimeMillis();
    double gameTime = (endTime - view.getStartTime()) / 1000.0;
    String text = "";

    Alert dialog= new Alert(alertType);
    dialog.setTitle("MineHunt - GameOver");
    dialog.setHeaderText("MineHunt");

    if(win){
      text = "Congratulations !\nCurrent game ended successfully (no error)";
    }else{
      text = "Current game ended with " + errors + " errors";
    }
    text += "\n\nGame ended after " + Double.toString(gameTime) + " seconds";

    dialog.setContentText(text);
    dialog.showAndWait();

  }


  public void newGame() {

    // model
    int col = model.gridWidth();
    int row = model.gridHeight();
    int mines = model.mines();

    if(model.getNewGridCol() > 0) col = model.getNewGridCol();
    if(model.getNewGridRow() > 0) row = model.getNewGridRow();
    if(model.getNewMinesNb() > 0) mines = model.getNewMinesNb();
    model.newModel(col, row, mines);

    // view
    view.getGrid().getChildren().clear();
    view.gridButtonCreation();
    view.setupVariables();

  }
  public void settingsGridSize(){

    Dialog<String[]> dialog = new Dialog<String[]>();
    dialog.setTitle("Settings");
    dialog.setHeaderText("Configure grid size");

    GridPane grid = new GridPane();
    grid.setHgap(10);
    grid.setVgap(10);
    grid.setPadding(new Insets(20, 150, 10, 10));

    TextField row = new TextField();
    row.setText(String.valueOf(model.gridHeight()));
    TextField col = new TextField();
    col.setText(String.valueOf(model.gridWidth()));

    grid.add(new Label("Row :"), 0, 0);
    grid.add(row, 1, 0);
    grid.add(new Label("Col :"), 0, 1);
    grid.add(col, 1, 1);

    ButtonType okButtonType = new ButtonType("OK", ButtonData.OK_DONE);
    dialog.getDialogPane().getButtonTypes().addAll(okButtonType, ButtonType.CANCEL);
    dialog.getDialogPane().setContent(grid);

    dialog.setResultConverter(dialogButton -> {
      if (dialogButton == okButtonType) {
        return new String[]{row.getText(), col.getText()};
      }
      return null;
    });

    Optional<String[]> input = dialog.showAndWait();

    if(input.isPresent()){
      String newRowS = input.get()[0].toString();
      String newColS = input.get()[1].toString();
      if(isInteger(newRowS) && isInteger(newColS)){

        int newRow = Integer.valueOf(newRowS);
        int newCol = Integer.valueOf(newColS);
        int newSize = newRow * newCol;
        int mines = model.mines();
        if(model.getNewMinesNb() > 0) mines = model.getNewMinesNb();

        if(newRow > 0 && newCol > 0 && newSize >= mines ){

          model.setNewGridRow(Integer.valueOf(newRow));
          model.setNewGridCol(Integer.valueOf(newCol));

        }else{

          Alert alert = new Alert(AlertType.ERROR);
          alert.setTitle("Settings");
          alert.setHeaderText(null);
          alert.setContentText("Grid size parameters should be bigger than 0 and grid size 
should not be smaller than mines number !");
          alert.showAndWait();

        }

      }
    }

  }
  public void settingsMines(){

    TextInputDialog dialog = new TextInputDialog(Integer.toString(model.mines()));
    dialog.setTitle("Settings");
    dialog.setHeaderText("Configure number of mines");
    dialog.setGraphic(null);
    Optional<String> input = dialog.showAndWait();

    if(input.isPresent()){
      if(isInteger(input.get())){

        int row = model.gridHeight();
        int col = model.gridWidth();
        if(model.getNewGridCol() > 0) col = model.getNewGridCol();
        if(model.getNewGridRow() > 0) row = model.getNewGridRow();

        if(Integer.valueOf(input.get()) > 0 && Integer.valueOf(input.get()) <= (row *
col)){
          model.setNewMinesNb(Integer.valueOf(input.get()));
        }else{

          Alert alert = new Alert(AlertType.ERROR);
          alert.setTitle("Settings");
          alert.setHeaderText(null);
          alert.setContentText("Mines number should be bigger than 0 and smaller than grid
size !");
          alert.showAndWait();

        }

      }
    }


  }

  public void settingsStyle() {

    String[] choices = {"Modena", "Caspian"};
    ChoiceDialog<String> cDialog= new ChoiceDialog<>(choices[0], choices);
    cDialog.setTitle("Settings");
    cDialog.setHeaderText("Configure look and feel");
    cDialog.setGraphic(null);
    cDialog.setContentText("Look & Feel : ");
    Optional<String> selection = cDialog.showAndWait();

    if(selection.isPresent()){
      view.setLF(selection.get());
    }

  }

  private static boolean isInteger(String s) {
    try {
      Integer.parseInt(s);
    }catch(NumberFormatException e) {
      return false;
    }catch(NullPointerException e) {
      return false;
    }
    return true;
  }


}
```

Cell

```
package s02;

public class Cell {

  private boolean open;
  private boolean flaged;
  private boolean mined;

  public Cell() {
    open = false;
    flaged = false;
    mined = false;
  }

  public boolean isOpen() {
    return open;
  }

  public void setOpen(boolean open) {
    this.open = open;
  }

  public boolean isFlaged() {
    return flaged;
  }

  public void setFlaged(boolean flaged) {
    this.flaged = flaged;
  }

  public boolean isMined() {
    return mined;
  }

  public void setMined(boolean mined) {
    this.mined = mined;
  }

}
```

CellButton

```
package s02;

import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.image.ImageView;
import javafx.scene.layout.*;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.paint.Color;

public class CellButton extends Button{

  private static final String FLAGG      = "/resources/MineHunt_Flagg.png";
  private static final String BOMB       = "/resources/MineHunt_Bomb.png";
  private static final String BOMB_SOUND = "/resources/MineHunt_Bomb_Explosion.mp3";

  private int row;
  private int col;
  private Media sound = new Media(getClass().getResource(BOMB_SOUND).toString());
  private MediaPlayer mediaPlayer = new MediaPlayer(sound);

  public CellButton(int row, int col) {
    this.row = row;
    this.col = col;
  }

  public int getRow() {
    return row;
  }

  public int getCol() {
    return col;
  }

  public void presentationClosed(){
    setGraphic(null);
  }

  // Open a cell
  public void presentationOpened(int nbrBombs){
    setGraphic(null);
    setBackground(new Background(new BackgroundFill(Color.rgb(250, 250, 0), new
CornerRadii(5), null)));
    if(nbrBombs != 0){
      setText(Integer.toString(nbrBombs));
    }
  }

  // Mark cell as flagged
  public void presentationFlagged(){
    setGraphic(new ImageView(FLAGG));
    setPadding(new Insets(0, 0, 0, 0));
  }

  // Open a mined cell
  public void presentationOpenedOnMine(boolean sound){
    setGraphic(new ImageView(BOMB));
    setPadding(new Insets(0, 0, 0, 0));
    setBackground(new Background(new BackgroundFill(Color.RED, new CornerRadii(5), null)));
    if(sound){
      mediaPlayer.play();
    }

  }

  // Show mines
  public void presentationShowMine(){
    setGraphic(new ImageView(BOMB));
    setPadding(new Insets(0, 0, 0, 0));
  }

}
```

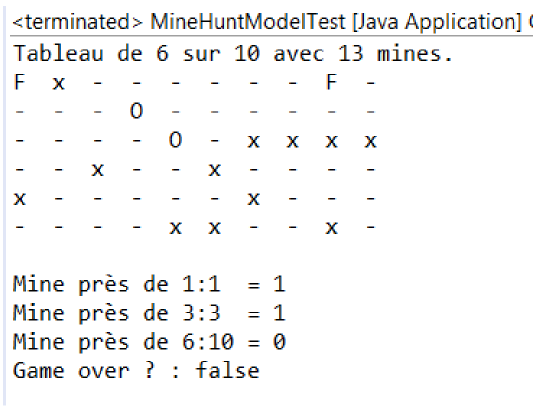
IMineHuntModel

```java
package s02;

public interface IMineHuntModel {

  /**
   * Initialize a new game
   * Reset game settings (counters and new values)
   * Random mines placement
   *
   * @param minesNb : Number of mines
   */
  void initNewGame(int minesNb);

  /**
   * Number of columns in grid
   */
  int gridWidth();

  /**
   * Number of rows in grid
   */
  int gridHeight();

  /**
   * Number of mines in grid
   */
  int mines();

  /**
   * Error counter
   */
  int errors();

  /**
   * Clicks counter
   */
  int clicks();

  /**
   * Number of mines in the neighborhood of a cell
   *
   * @param row Cell row index
   * @param col Cell column index
   */
  int neighborMines(int row, int col);

  /**
   * Return true if cell is open
   *
   * @param row : Cell row index
   * @param col : Cell column index
   */
  boolean isOpen(int row, int col);

  /**
   * Return true if cell is flagged
   *
   * @param row : Cell row index
   * @param col : Cell column index
   */
  boolean isFlagged(int row, int col);
  /**
   * Return true if cell is mined
   *
   * @param row : Cell row index
   * @param col : Cell column index
   */
  boolean isMined(int row, int col);

  /**
   * Check if game is over
   * Return true only if all non-mined cells have been opened
   *
   * @return true if game over
   */
  boolean isGameOver();

  /**
   * Open a cell
   * No effect if cell is already open.
   *
   * @param  row   Cell row index
   * @param  col   Cell column index
   * @return true if a mine was in the cell
   */
  boolean open(int row, int col);

  /**
   * Set cell to flagged
   *
   * @param row   : Cell row index
   * @param col   : Cell column index
   * @param state : flag state
   */
  void setFlagState(int row, int col, boolean state);


  /**
   * Open a cell (used when user click on cell with no neighbor mines)
   * No effect if cell is already open or if there is a bomb
   *
   * @param row   : Cell row index
   * @param col   : Cell column index
   */
  boolean openProximity(int row, int col);


  /**
   * Generate a new model (used to start a new game)
   *
   * @param gridSizeCol : col
   * @param gridSizeRow : row
   * @param minesNb     : Number of mines
   */
  void newModel(int gridSizeCol, int gridSizeRow, int minesNb);

}
```

MineHuntModelTest

```java
package s02.test;

import s02.MineHuntModel;

public class MineHuntModelTest {

  public static void main(String[] args) {

    MineHuntModel mhm = new MineHuntModel(10, 6, 13);

    mhm.setFlagState(0, 0, true);
    mhm.grid[8][0].setFlaged(true);
    mhm.open(2, 4);
    mhm.grid[3][1].setOpen(true);

    // affichage grille
    System.out.println("Tableau de " + mhm.gridHeight() + " sur " + mhm.gridWidth() + "
avec " + mhm.mines() + " mines.");
    for(int j = 0; j < mhm.gridHeight(); j++){
      for(int i = 0; i < mhm.gridWidth(); i++){

        if(mhm.grid[i][j].isOpen()){
          System.out.print("O  ");
        }else if(mhm.grid[i][j].isFlaged()){
          System.out.print("F  ");
        }else if(mhm.grid[i][j].isMined()){
          System.out.print("x  ");
        }else{
          System.out.print("-  ");
        }

      }
      System.out.println("");
    }


    System.out.println();
    System.out.println("Mine près de 1:1  = " + mhm.neighborMines(1, 1));
    System.out.println("Mine près de 3:3  = " + mhm.neighborMines(3, 3));
    System.out.println("Mine près de 6:10 = " + mhm.neighborMines(6, 10));
    System.out.println("Game over ? : " + mhm.isGameOver());

  }

}
```

Affichage dans la console d'un test du modèle :



```
<terminated> MineHuntModelTest [Java Application] (
Tableau de 6 sur 10 avec 13 mines.
F  x  -  -  -  -  -  -  F  -
-  -  -  O  -  -  -  -  -  -
-  -  -  -  O  -  x  x  x  x
-  -  x  -  -  x  -  -  -  -
x  -  -  -  -  -  x  -  -  -
-  -  -  -  x  x  -  -  x  -

Mine près de 1:1  = 1
Mine près de 3:3  = 1
Mine près de 6:10 = 0
Game over ? : false
```