

LANGAGE AVANCÉ DE PROGRAMMATION

---

## Projet Java GénialBall

---

*Auteur :*  
Nicolas Sias 2TL1  
Youri Mouton 2TL2

*Année académique :*  
2014-2015



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Qu'est-ce que GénialBall ? . . . . .	1
1.2	Comment y jouer ? . . . . .	1
1.3	Interface du jeu . . . . .	2
1.4	Les contrôles du jeu . . . . .	2
<b>2</b>	<b>Extrait UML</b>	<b>3</b>
<b>3</b>	<b>Extrait du code source (SpaceBalls.java)</b>	<b>4</b>
<b>4</b>	<b>Documentation JavaDoc HTML</b>	<b>5</b>
<b>5</b>	<b>Description de votre stratégie de validation</b>	<b>6</b>
5.1	Utilisation d'un package test . . . . .	6
5.2	JUnit Testing : . . . . .	6
5.3	System.out.println . . . . .	6
<b>6</b>	<b>Conclusion</b>	<b>7</b>



# Chapitre 1

## Introduction

### 1.1 Qu'est-ce que GénialBall ?

GénialBall est un jeu de sport, dans un mélange de football et d'air-hockey, où deux joueurs s'affrontent en réseau local, ayant comme but de pousser le ballon dans le but adverse.

### 1.2 Comment y jouer ?

Lancez l'application, choisissez d'être le client ou le serveur ( un joueur doit être le client, l'autre doit être le serveur), attendez que la recherche d'IP local se termine, puis sélectionnez l'IP local de votre adversaire.

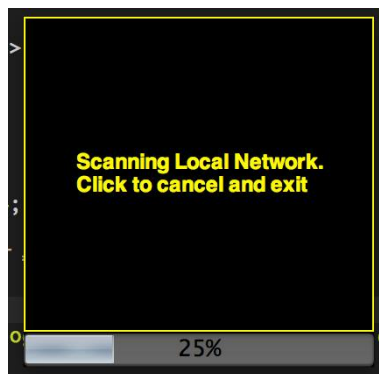


FIGURE 1.1 – Le jeu va d'abord chercher les autres machines connectées dans le réseau local.

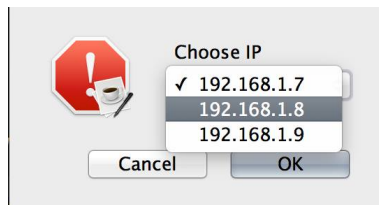
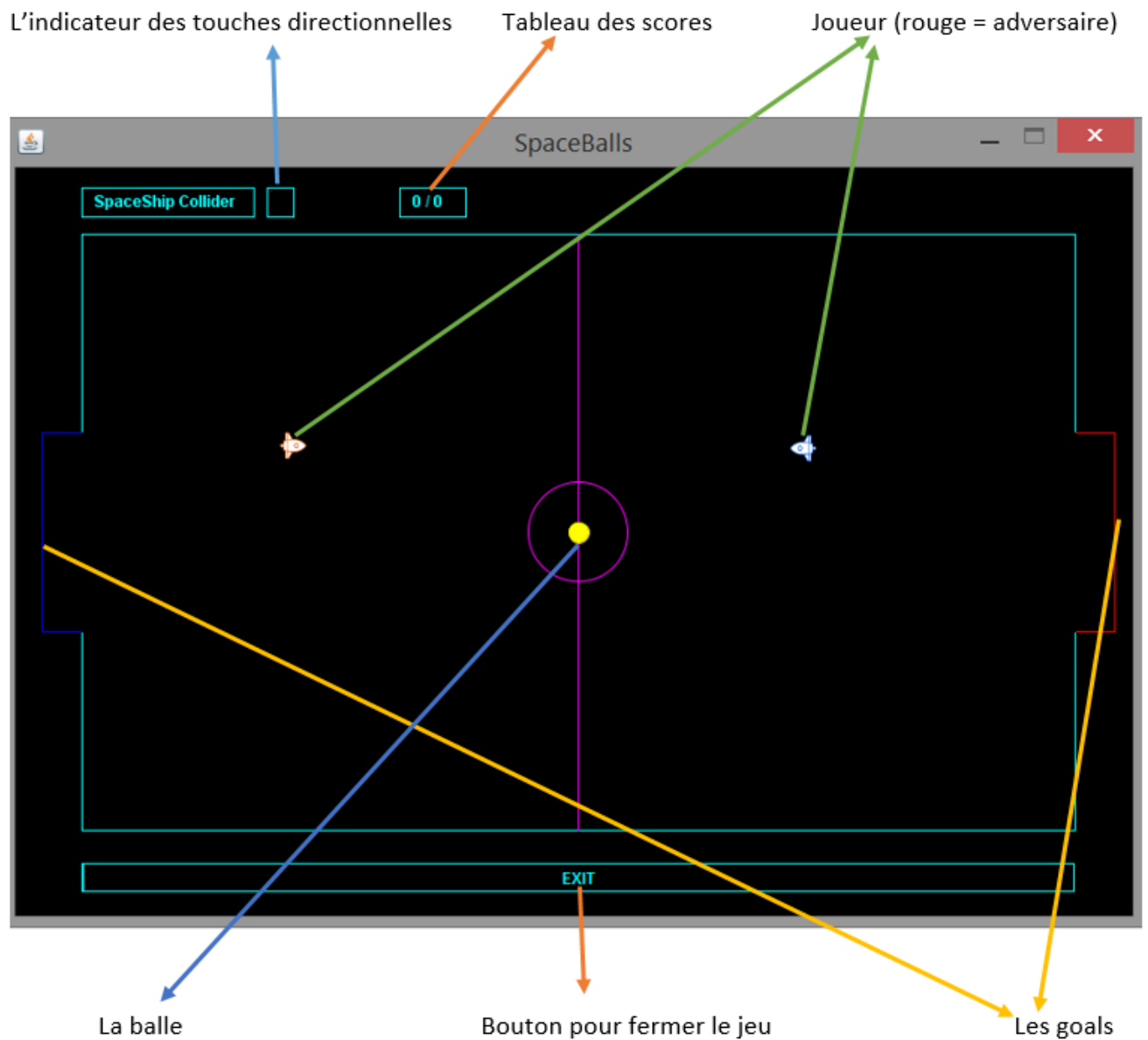


FIGURE 1.2 – Sélectionnez ensuite l'IP local de votre adversaire

### 1.3 Interface du jeu



### 1.4 Les contrôles du jeu

- Utilisez les touches directionnelles pour déplacer votre vaisseau.
- Appuyez sur la barre d'espace si vous désirez faire un tir avec la balle.
- Si vous désirez quitter, cliquez sur la touche "Exit" ou appuyez sur Esc.

## Chapitre 2

### Extrait UML

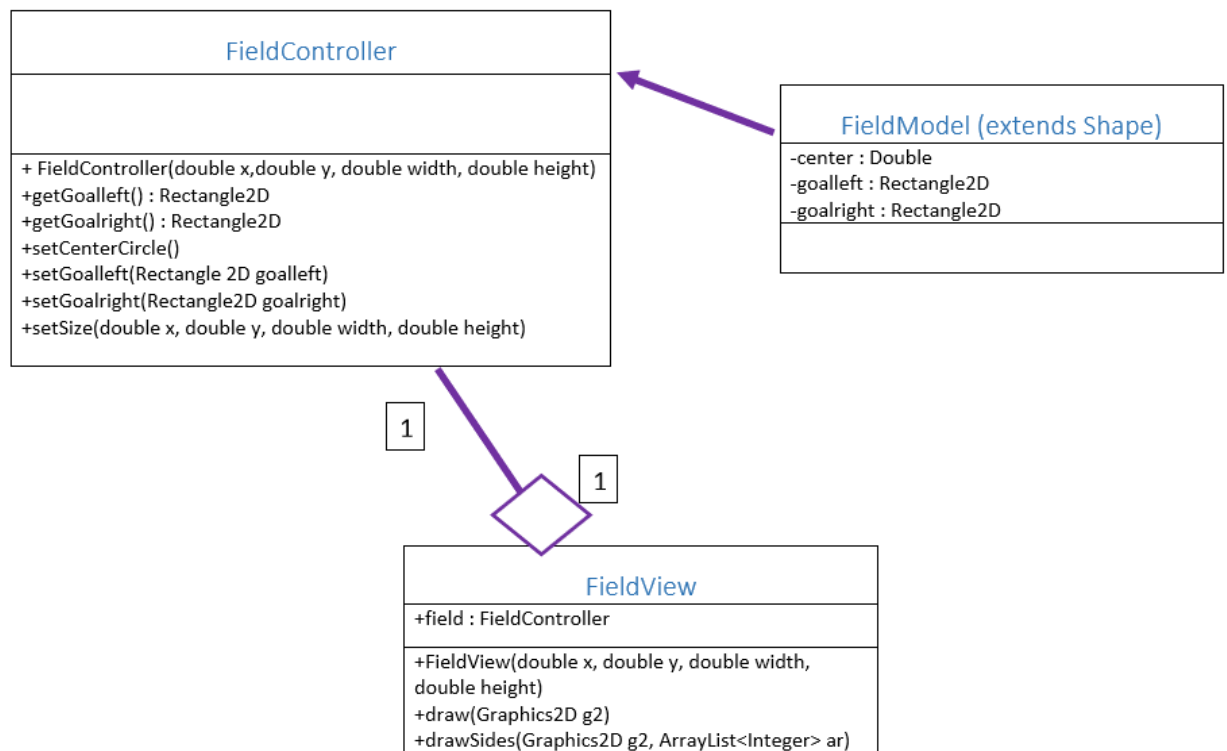


FIGURE 2.1 – Extrait UML des classes terrain

## Chapitre 3

# Extrait du code source (SpaceBalls.java)

```
package main;

import gui.Board;

import java.awt.EventQueue;

import javax.swing.JFrame;

@SuppressWarnings("serial")
public class SpaceBalls extends JFrame {

    Board mainBoard;

    public SpaceBalls() throws Exception {
        // our main window is 850x600 for now.
        this.setSize(850, 600);

        // initialize a board which takes the whole screen
        mainBoard = new Board(this.getSize());

        // add the jpanel in the jframe
        add(mainBoard);

        // jframe settings
        this.setTitle("SpaceBalls");
        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);
    }





















    public static void main(String[] args) {
        // main loop
        EventQueue.invokeLater(new Runnable() {
            // main application
            @Override
            public void run() {
                try {
                    SpaceBalls ex = new SpaceBalls();
                    ex.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```



## Chapitre 4

# Documentation JavaDoc HTML

Toute la javaDoc est dans un fichier annexe "doc". Si vous désirez lire notre javaDoc, veuillez selectionner dans ce fichier "index.html"

Nom	Modifié le	Type	Taille
 geo	15-12-14 16:30	Dossier de fichiers	
 gui	15-12-14 16:30	Dossier de fichiers	
 index-files	15-12-14 16:30	Dossier de fichiers	
 main	15-12-14 16:30	Dossier de fichiers	
 net	15-12-14 16:30	Dossier de fichiers	
 test	15-12-14 16:30	Dossier de fichiers	
 .DS_Store	15-12-14 16:30	Fichier DS_STORE	9 Ko
 allclasses-frame.html	15-12-14 16:30	Fichier HTML	4 Ko
 allclasses-noframe.html	15-12-14 16:30	Fichier HTML	3 Ko
 constant-values.html	15-12-14 16:30	Fichier HTML	24 Ko
 deprecated-list.html	15-12-14 16:30	Fichier HTML	4 Ko
 help-doc.html	15-12-14 16:30	Fichier HTML	9 Ko
 index.html	15-12-14 16:30	Fichier HTML	3 Ko
 overview-frame.html	15-12-14 16:30	Fichier HTML	2 Ko
 overview-summary.html	15-12-14 16:30	Fichier HTML	5 Ko
 overview-tree.html	15-12-14 16:30	Fichier HTML	10 Ko
 package-list	15-12-14 16:30	Fichier	1 Ko
 script.js	15-12-14 16:30	Fichier de JavaScri...	1 Ko
 serialized-form.html	15-12-14 16:30	Fichier HTML	14 Ko
 stylesheet.css	15-12-14 16:30	Document de feuil...	14 Ko

## Chapitre 5

# Description de votre stratégie de validation

### 5.1 Utilisation d'un package test

Pour tester les points importants de notre jeu, nous avons créé plusieurs classes "test" dans un package mis à l'écart. Voici le contenu du package "test" :

**ClientTest.java** : Cette classe sert à détecter les erreurs de réception et/ou d'envoi de chaînes de caractères au serveur.

**ExitButtonTest.java** : Elle sert à vérifier si le bouton "Exit" lorsque le joueur 2 quitte la partie fonctionne.

**FieldTest.java** : Elle sert à détecter des erreurs lors de l'initialisation du Field, en liant le contrôle au visuel, utilisant du JUnit Testing

**InetTest.java** : Elle vérifie la détection de l'IP local par java (et savoir, par la même occasion, l'IP local du joueur lançant cette application).

**SwingWorkerTest** : Elle teste la recherche d'IP local (donc la classe DiscoverLocal.java).

**TextFieldTest** : Cette classe, utilisant le JUnit Testing, vérifie si l'affichage textuelle sur le terrain fonctionne.

### 5.2 JUnit Testing :

On a utilisé le JUnit Testing principalement pour l'affichage textuelle sur le terrain ( via la classe TextFieldTest) ainsi que pour l'initialisation du terrain, en liant le contrôle au visuel (via la classe FieldTest)

### 5.3 System.out.println

Cette méthode a été utilisé très fréquemment pour détecter les collisions entre les objets (balle, terrain et joueur) .

## Chapitre 6

# Conclusion

De manière générale, la conception de ce jeu nous a permis d'apprendre beaucoup sur l'Orienté Object appliqué à un projet avoisinant les 3000 lignes de code et les design patterns tels que le MVC.

Les gestions de collisions avec la balle a posé beaucoup de problèmes, la classe Ball.java a été réécrite plusieurs fois.

Le projet a permis une bonne introduction à la programmation événementielle, aux éléments d'interface graphique et au threading, qui a permis de lancer nos processus jeu, serveur et client de manière concurrente sans bloquer les éléments d'interface graphique.

La manière dont le programme envoie les informations pourrait être fort améliorée, utiliser UDP permettrait de pouvoir jouer sans que le déplacement des joueurs et de la balle soit saccadé.

Nous utilisons une boîte de dialogue avec une barre de progrès lorsque le programme essaye de trouver des ordinateurs connectés sur le réseau local, nous pourrions lancer une requête UDP en broadcast pour ne trouver que les ordinateurs qui ont un ServerSocket écoutant sur un port qui nous intéresse.

L'utilisation d'une librairie graphique aurait pu nous faciliter la tâche et nous permettre de nous concentrer sur d'autres aspects et détails du jeu, mais ce fut un apprentissage intéressant.

Le programme manque d'un panneau de configuration graphique que l'on aurait bien voulu coder, mais le temps donné ne nous l'a pas permis.