

---

# Projet de Programmation 2 - Ksparov

---

Année 2016-2017, Semestre 2

# Table des matières

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>L'organisation du code</b>                                    | <b>2</b> |
| 1.1      | draw.scala ou l'interface utilisateur . . . . .                  | 2        |
| 1.2      | rules.scala ou l'implémentation des règles des échecs . . . . .  | 3        |
| 1.3      | game.scala, où alterner est important . . . . .                  | 3        |
| 1.4      | solve.scala ou la partie, pour le moment, sans intérêt . . . . . | 3        |
| <b>2</b> | <b>Les points clés de l'implémentation</b>                       | <b>3</b> |
| <b>3</b> | <b>Les défauts et erreurs connus</b>                             | <b>3</b> |
| <b>4</b> | <b>Pourquoi le design est magnifique</b>                         | <b>3</b> |
| <b>5</b> | <b>Pourquoi je suis beau</b>                                     | <b>3</b> |
| <b>6</b> | <b>Pourquoi pas ?</b>  | <b>3</b> |

De tout temps les Hommes ont cherché à jouer aux échecs ! Depuis le début de l'informatique moderne et le développement grand public des micro-processeurs de nombreuses applications ont vu le jour pour permettre à tout le monde de jouer aux échecs.

En effet qui ne rêve pas de pouvoir s'entraîner sans relâche pour atteindre le niveau d'un Bobby Fischer, ou de pouvoir revivre des parties de légende comme la finale de 1985 opposant Kasparov à Karpov.

Vous faire vivre cela, et bien plus encore, a été notre leitmotiv durant le développement de cette application. Voici la première version du meilleur jeu d'échec jamais créé, voici Ksarov !

*Le plus grand jeu de l'esprit jamais inventé, plus vous l'apprenez, plus vous y prenez du plaisir.*  
Garry Kasparov

## 1 L'organisation du code

Le code de notre projet a été divisé en quatre parties (chacune dans un fichier) qui correspondent chacune à une fonction particulière du code. On retrouve donc :

- draw.scala qui comprend tout ce qui concerne l'affichage de l'interface du jeu
- rules.scala qui implémente toutes les règles du jeu pour chaque pièce
- game.scala qui gère le lancement du jeu et qui fait l'interface entre les règles et l'application
- solve.scala qui définit ce qui a trait à la résolution par l'intelligence artificielle

Nous allons donc à présent étudier en détail chacun de ces fichiers.

### 1.1 draw.scala ou l'interface utilisateur

Le code a ici été séparé en plusieurs objets, chaque objet comprend tout ce qui concerne l'affichage d'une fenêtre particulière. Nous retrouvons donc un objet pour le menu initial, un pour le réglage des paramètres, un pour le plateau de jeu...

**Les fonctionnalités de l'interface :** L'interface a été créée de manière à pouvoir implémenter tout le projet, les trois parties, sans avoir à la modifier à chaque phase. Ainsi le jeu s'ouvre sur un menu général permettant pour le moment de lancer une partie, de modifier les paramètres et de quitter le jeu. Des boutons existent déjà pour le chargement d'une partie, les scores des joueurs... mais ne sont pas fonctionnels.

Le lancement d'une partie fait appel à un autre menu, celui qui permet de choisir le mode de jeu pour la partie : Humain/Humain, Humain/IA ou IA/IA.

**L'organisation générale d'un objet Draw... :** Chaque objet du fichier draw.scala a un nom construit comme DrawFenetre où Fenetre est l'élément affiché par cet objet (DrawParameters, DrawMenu, DrawBoard...). Tous les objets sont construits de la même manière et on y retrouve les mêmes classes : tous les éléments d'une fenêtre sont des extensions de GridPanel qui sont ensuite organisés par via un BorderLayout. Au total toute fenêtre est découpée en carré de 80 pixel.

Pour avoir un code plus propre une classe commune à tous les objets a été créée : BackgroundCase qui étend un GridPanel et qui permet selon ses paramètres de dessiner un certain nombre de cases de fond.

**Les paramètres :** Tous les visuels des textures des cases et des pièces de jeu peuvent être modifiées dans les paramètres. La plupart des textures ne sont pas très esthétiques mais ceci est un autre problème. Les paramètres sont enregistrés dans un fichier : src/main/resources/Parameters ce qui permet de conserver les choix pour chaque ouverture de l'application.

**Le plateau de jeu :** Le plateau de jeu est, heureusement, la fenêtre la plus complète, on y retrouve plusieurs éléments.

Tout d'abord le plateau, qui est constitué de case sur lesquelles les pièces se rajoutent via une icône. La particularité étant que le plateau dessine aussi les pièces lorsque l'on en crée une nouvelle instance.

Sur chaque côté du plateau on retrouve un compte des pièces mortes pour chaque joueur. Ceci est fait par le dessin des pièces en considérant que les positions négatives correspondent aux pièces mortes.

Finalement on retrouve en haut de la fenêtre un menu rapide qui permet pour le moment simplement de revenir au menu.

## 1.2 rules.scala ou l'implémentation des règles des échecs

## 1.3 game.scala, où alterner est important

Tout le soucis de cette partie est de bien alterner les différentes phases du jeu et ce de manière intuitive, comme l'exécution n'est pas linéaire, à cause de l'action des joueurs sur l'interface, il faut un système de "pare-feux" qui fonctionne en s'appuyant sur le principe des sémaphores. Un objet Switches régule donc les différentes parties du programme qui sont :

- (1) le premier clic de sélection d'une pièce
- (2) le deuxième clic qui peut être :
  - (a) un coup valide donc (3)
  - (b) un coup invalide donc retour au (1)
  - (c) la sélection d'une autre pièce donc (2)
- (3) le changement de joueur puis (1)

Pour cela trois variables binaires, la première se charge de stocker qui est le joueur courant, la deuxième si l'on doit attendre un clic de sélection de pièce, la troisième si l'on attend un clic de choix de mouvement.

Autre spécificité de cette partie, elle fait le lien entre l'interface et les règles c'est donc là que se trouvent les constantes utiles à l'ensemble du programme.

## 1.4 solve.scala ou la partie, pour le moment, sans intérêt

# 2 Les points clés de l'implémentation

Là et là !

# 3 Les défauts et erreurs connus

L'implémentation telle qu'elle est actuellement connaît certaines lourdeurs dans le code. En effet, pour les pièces le code utilise deux variables : une liste de 32 pièces et une liste de 64 cases qui sert à positionner chaque pièce de la première liste. Le code pourrait être modifier pour n'utiliser plus que la liste des 32 pièces.

A vous !

# 4 Pourquoi le design est magnifique

Parce que je l'ai réalisé !

# 5 Pourquoi je suis beau

Parce que !

# 6 Pourquoi pas ?

Et oui !