POLYTECH SORBONNE - MAIN4

Projet Yellow

Une partie d'échecs

Anass MELLOUKI
Constance JACQUES

Encadrants : Cécile Braunstein et
Jean-Baptiste Brejon

Janvier 2019

Table des matières

1 Pourquoi des échecs ?

"*Les Échecs vous apprennent à vous concentrer, à améliorer votre logique. Ils vous enseignent à jouer selon les règles, à prendre la responsabilité de vos actions et comment résoudre un problème dans un environnement incertain.*"
Garry Kasparov

"*Le jeu d'échecs possède cette remarquable propriété de ne pas fatiguer l'esprit et d'augmenter bien plutôt sa souplesse et sa vivacité.*"
Garry Kasparov

Toutes les études depuis ont confirmé que jouer aux échecs permettait :
- De développer une pensée logique
- D'améliorer la capacité à résoudre des problèmes
- D'améliorer l'attention et la concentration
- De favoriser la mémoire...

2 Les contraintes

- Thème : **Yellow**
- Langage de programmation : **c++**
- Nombre de classes minimal : **8**
- Niveaux de hiérarchie : **3**
- Fonctions virtuelles : **2**
- Surcharges d'opérateurs : **2**
- Conteneurs : **2**
- Codes commentés
- Absence d'erreur avec Valgrind
- Utilisation d'un Makefile

3 Structure

3.1 Règles

Si vous voulez jouer à notre jeu, il faut savoir que nous n'avons pas programmé la technique du roque et que pour finir une partie (déclarer échec et mat), vous devez vous emparer du roi avec une de vos pièces. De plus, à cause de la contrainte des 3 niveaux de hiérarchie, nous avons été obligés de mettre deux types de pions dans le jeu. Les pions bleus fonctionnent comme des pions ordinaires tandis que les pions rouges ont la particularité de capturer les pièces adverses (se trouvant devant eux) en se déplaçant verticalement contrairement aux pions ordinaires qui capturent en diagonale.

Ces inconvénients mis à part, le jeu que nous avons programmé fonctionne comme un jeu d'échecs classiques !

3.2 Composition

Nombre de classes	Niveaux de hiérarchie	Fonctions virtuelles	Surcharges d'opérateurs	Conteneurs
10	3	2	3	5

3.3 Réprésentation hiérarchique

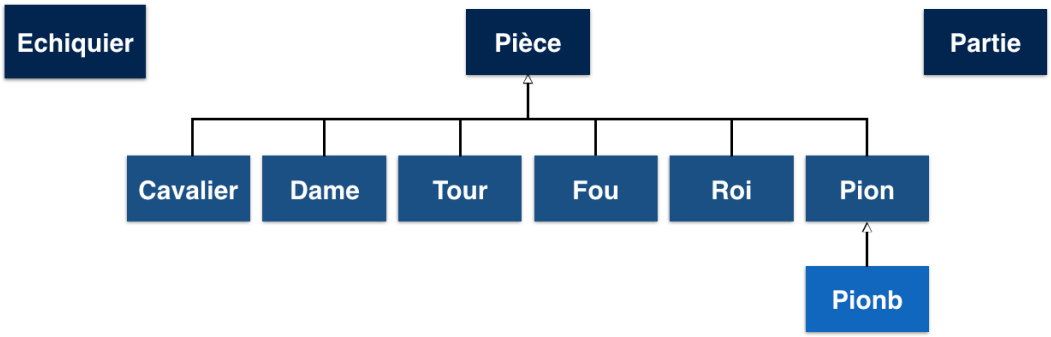

Figure 1 - Hiérarchie des classes *

4 Programmation du jeu

4.1 Les classes

Dans ce programme, nous avons créé 10 classes :

1. *Echiquier* : Terrain d'évolution de la partie
2. *Partie* : Déroulement du jeu
3. *Pièce* : Pièce d'échecs quelconque
4. *Roi* : Code du Roi
5. *Dame* : Code de la Dame
6. *Fou* : Code du Fou
7. *Cavalier* : code du Cavalier
8. *Tour* : Code de la Tour
9. *Pion* : Code du Pion bleu
10. *Pionb* : Code du Pion rouge

Ces 10 classes nous permettent de respecter les contraintes de classe et hiérarchisation.

4.2 Les fonctions virtuelles

Nous avons utilisé 2 fonctions virtuelles dans notre hierarchisation des pièces :

— *virtual bool validation(int x_vise, int y_vise, Echiquier E)* : Fonction qui permet de voir si le mouvement que veux faire le joueur est possible. Cette fonction dépendant du type de la pièce, il paraît donc judicieux d'utiliser "virtual" ici.
— *virtual void deplacement(int x_vise, int y_vise, Echiquier E)* : Fonction qui effectue le déplacement. Cette fonction dépendant de la validation, elle est donc virtuelle.

∗. Vrai diagramme UML fourni dans le fichier envoyé avec les codes

4.3 Les surcharges d'opérateur

Concernant les surcharges d'opérateurs, nous avons dû en faire 3. La première est la surcharge de l'opérateur "<<" de la classe Echiquier. Cette surcharge permet d'afficher de manière optimisée l'échiquier sur le terminal.
La deuxième est celle de l'opérateur de "<<" mais de la classe Piece. Cette surcharge permet d'afficher la position actuelle de la pièce.
La dernière surcharge est l'opérateur d'affectation "=" de la classe Piece.

4.4 Les conteneurs

Trois vecteurs ont été utilisés pour la fonction "validation" des classes Dame, Fou et Tour. Nous expliquerons plus tard dans le rapport comment nous avons utilisé ces conteneurs (partie 4.7).
Nous avons aussi utilisé deux listes dans la classe Partie. La première contient toutes les pièces du joueur 1 et la deuxième celles du joueur 2. Ces listes sont utiles pour voir si les pièces sont encore disponibles ou si elles ont été capturées par le joueur adverse.

4.5 Interface graphique

Afin de rendre cette partie d'échecs plus attractive et plus divertissante, nous avons fait le choix d'utiliser Simple DirectMedia Layer (SDL), une bibliothèque logicielle libre utilisée pour créer des applications multimédias en deux dimensions.
Dans notre *main*, nous incluons les bibliothèques $SDL2/SDL.h$ et $SDL2/SDL\_image.h$, bibliothèque de chargement de fichier image (en tant que surfaces et textures SDL).

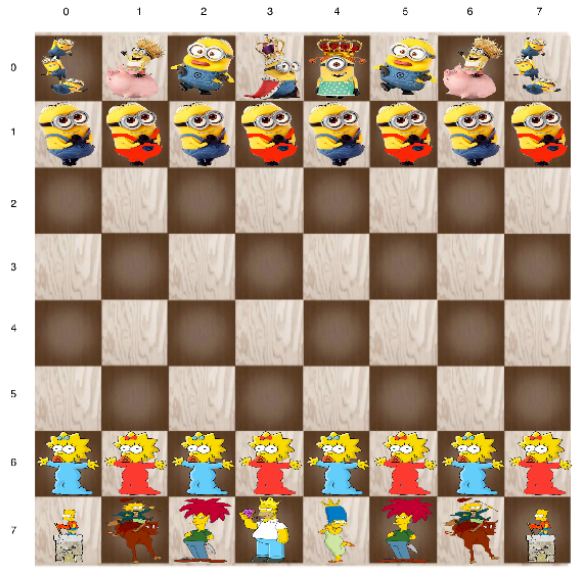

Figure 2 - Interface après exécution

4.6 Partie de l'implémentation dont nous sommes les plus fiers

Une partie du code dont nous sommes fiers est la fonction "validation" de la classe Dame.
Le but de cette fonction est de voir si le déplacement que veux faire un joueur est autorisé. On commence d'abord par chercher les cases sur lesquelles la dame peut être déplacée, sachant qu'une dame peut aller dans toutes les directions.
La solution que nous avons trouvée pour répondre à ces critères est la suivante : créer un vecteur

d'entiers dans lequel nous stockons les cases jouables.
Pour convertir une case en entier, on met le numéro de la ligne en dizaine et le numéro de la colonne en unité. Par exemple, si on veut aller sur la case (3,7), l'entier sera 37.
Une fois que cela est fait, on parcourt le vecteur d'entiers en regardant afin de trouver une valeur correspondante à la case sur laquelle le joueur veut se rendre. Si c'est le cas, la fonction retourne "true". Sinon, elle retourne "false".

4.7 Exécution du jeu

Pour lancer la partie, il suffit de "make" sur le terminal et de lancer l'exécutable "./main".
Ceci étant fait, la partie peut commencer. Pour jouer, il faut utiliser le terminal, l'affichage SDL permet uniquement de mieux visualiser la partie. En effet, nous n'avons pas réussi à donner aux joueurs la possibilité d'utiliser la souris (pour le moment).
Il faut donc entrer l'identifiant de la pièce (les identifiants de toutes les pièces sont affichés sur le terminal), puis entrer la ligne et la colonne de la case visée.

5 Bibliographie

Ci-dessous se trouvent les recherches que nous avons effectués pour écrire nos codes et ce rapport.

1. Bienfaits des échecs
   `http://tpgbesancon.com/de-linteret-de-jouer-aux-echecs/`

2. Bibliothèque SDL
   `https://fr.wikipedia.org/wiki/Simple_DirectMedia_Layer`
   `https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/17117-installatio`
   `https://alexandre-laurent.developpez.com/tutoriels/sdl-2/`