

# Projet ODL 2014/2015

## Jeu de l'Halma

encadré par M. Gilles Lebrun

projet réalisé par Tendry Rakotondramasy et  
Quentin Geliot

## **Avant Propos :**

Compte tenu de la taille de notre projet et des nombreux fichiers .h et .c, la documentation est très lourde.

Pour le faire tenir dans la plateforme moodle, nous avons dû compresser tous les dossiers en raison de la capacité maximale de celle-ci.

En espérant que vous ne nous en tiendrez pas rigueur.

## **Objectif du projet :**

Le but du projet était de réaliser la conception du jeu de l'Halma(plateau, pions, joueurs...), des IA, un didacticiel permettant la compréhension du jeu pour un novice, ainsi que la possibilité d'annuler des coups et des tours.

### **Les règles du jeu :**

Les règles sont simples : Le joueur doit déplacer ses pions aux emplacements de départ des pions du joueur adverse.

Les déplacements se font de deux manières : le déplacement simple (le pion se déplace d'une case à côté de lui), et le déplacement par saut (le pion saute un autre pion, peu importe sa couleur). Si le pion a sauté un autre pion, il peut continuer à en sauter d'autres tant que cela reste possible.

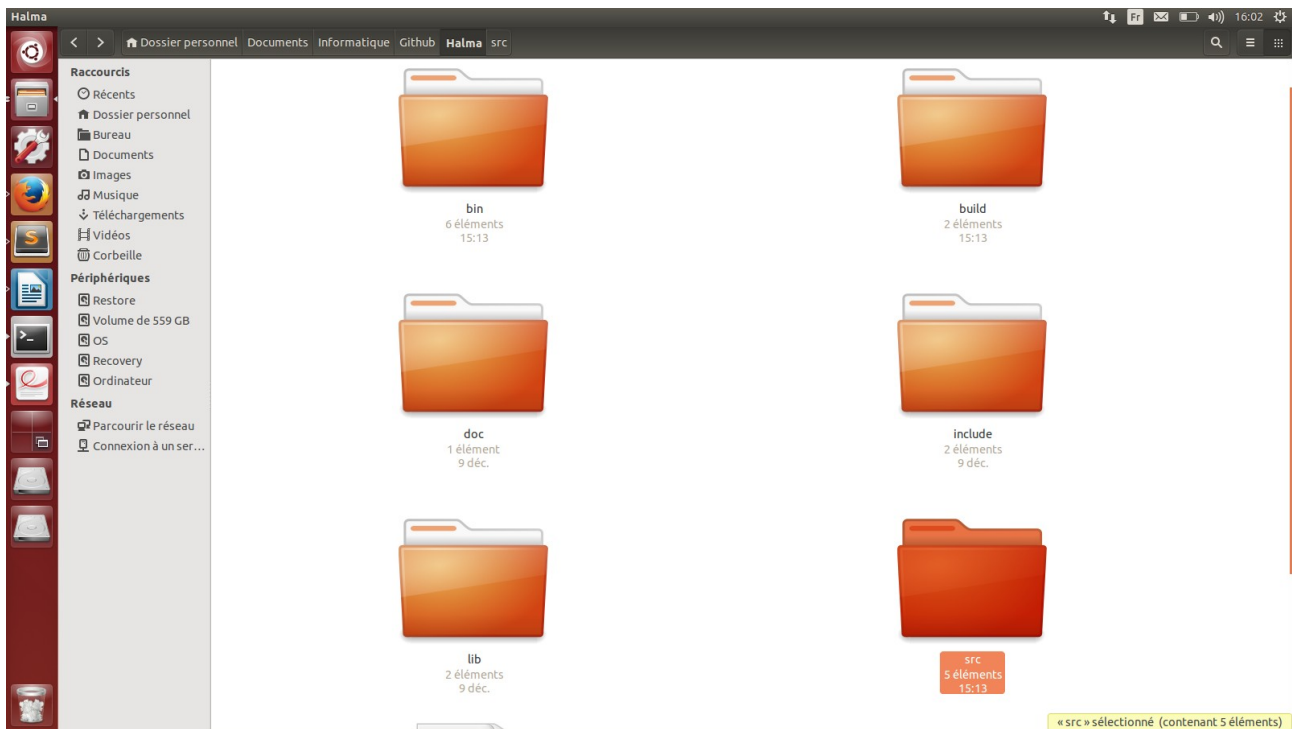
Dans l'exemple suivant, le joueur jaune a mis ses pions à l'endroit exact où étaient les pions du joueur bleu : il gagne la partie.

### **Réalisation du projet :**

#### **- Outils de développement logiciels :**

Des makefiles rékursifs ont été créés pour la compilation.

L'architecture classique d'un projet a également été mise en place :



La portabilité du code sur Windows a été gérée mais le soucis est que la gestion des affichages a été définie sur un environnement LINUX, cet environnement étant non compatible avec celui de WINDOWS, l'affichage du programme sous WINDOWS n'est pas fonctionnel, nous avons manqué de temps pour apporter les divers correctifs à ce sujet, mais la compilation et la création de l'exécutable est bel et bien fonctionnelle. Nous n'avons pas eu recours à l'utilisation d'un IDE et avons préféré un usage séparé des diverses fonctions du projet, un logiciel de gestion de version, un éditeur de texte, et un terminal couplés aux makefiles pour la compilation.

#### Le dossier bin :

Ce dossier contient les exécutables du jeu pour toutes les plateformes (la DLL pour windows), les deux fichiers de démarrage (pour 2 joueurs ou 4), le dossier où sont sauvegardées les parties, et un dossier didacticiel contenant les fichiers pour le didacticiel.

#### Le dossier build :

Ce dossier contient tous les .o dans le dossier source ainsi qu'un makefile

#### Le dossier doc :

Ce dossier contient toutes les documentations des fonctions (mise en place par le logiciel doxygen).

#### Le dossier include :

Ce dossier contient le .h de la bibliothèque libMatrice (ce .h se situe dans le dossier libMatrice)

### Le dossier lib :

Ce dossier contient la bibliothèque dynamique libMatrice

### Le dossier src :

Le projet a une architecture MVC : Modele, Vue, Controleur :

#### - Le modele :

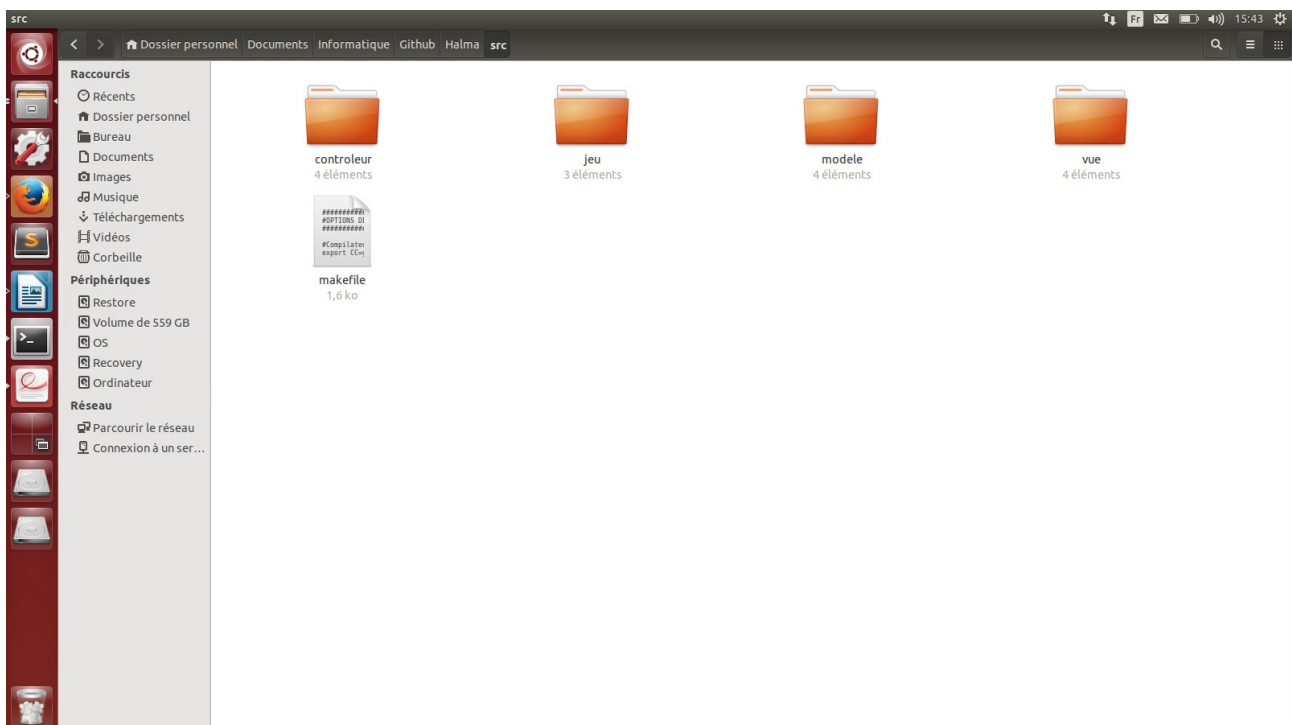
Ce dossier contient toutes les fonctions destinées à la mise en place du jeu de l'Halma : les définitions des pions, positions, joueurs, plateau, modele... mais également les fonctions permettant de choisir un pion sur le plateau, de déplacer un pion selon les règles, la vérification de victoire etc

#### - La vue :

Ce dossier contient toutes les fonctions relatives à l'affichage du jeu : l'affichage du plateau et de ses pions, mais aussi l'affichage des textes permettant la compréhension du jeu (menu, consignes pour jouer etc).

#### - Le controleur :

Ce dossier contient toutes les fonctions permettant à l'utilisateur de jouer au jeu. Ces fonctions exploitent les fonctions des dossiers précédents.



## **Le modèle**

Le modele est composé de 3 dossiers et d'un makefile.

Les 3 dossiers sont :

- header : contient tous les .h du modele
- souce : contient tous les .c du modele
- test : ce dossier est particulier : il contient un main indépendant ainsi qu'un makefile lui aussi indépendant. Ces deux fichiers nous ont permis de tester nos fonctions, de les corriger si nécessaires afin de créer un modèle solide.

Dans l'image qui suit, vous verrez d'autres fichiers que nous n'avons pas mentionné (des fichiers .txt), ceux-ci n'ayant pas d'importances notoires.

## **La vue**

La vue est également composée de 3 dossiers et d'un makefile comme le modèle.

Ces 3 dossiers ont les mêmes caractéristiques que les dossiers définies précédemment.

## **Le controleur**

Comme pour la vue et le modele, l'architecture du dossier controleur est le même que précédemment.

## **Jeu**

Ce dossier est particulier. En effet, il n'est composé que de deux dossiers ainsi que d'un makefile.

Le dossier source ne contient que le main principal relatif au jeu !

En ce qui concerne le dossier test et du makefile, leur définition n'a pas changé.

Nota Bene : tous les makefiles situés précédemment sont appelés par le makefile principal. Seul ceux situés dans les dossiers test sont indépendants !

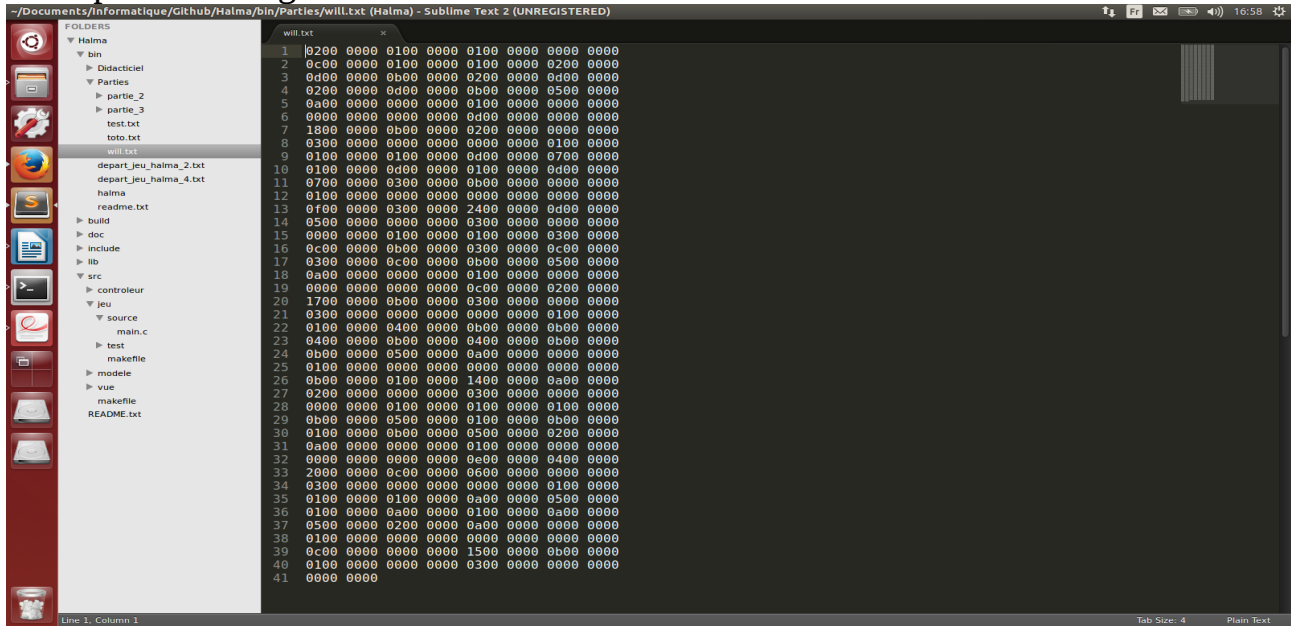
# A propos des consignes détaillées

L'annulation des coups ainsi que des tours est faite par des piles.

La sauvegarde prend en compte tous les coups ainsi que tous les tours. L'utilisateur reprendra sa partie là où il l'avait laissé.

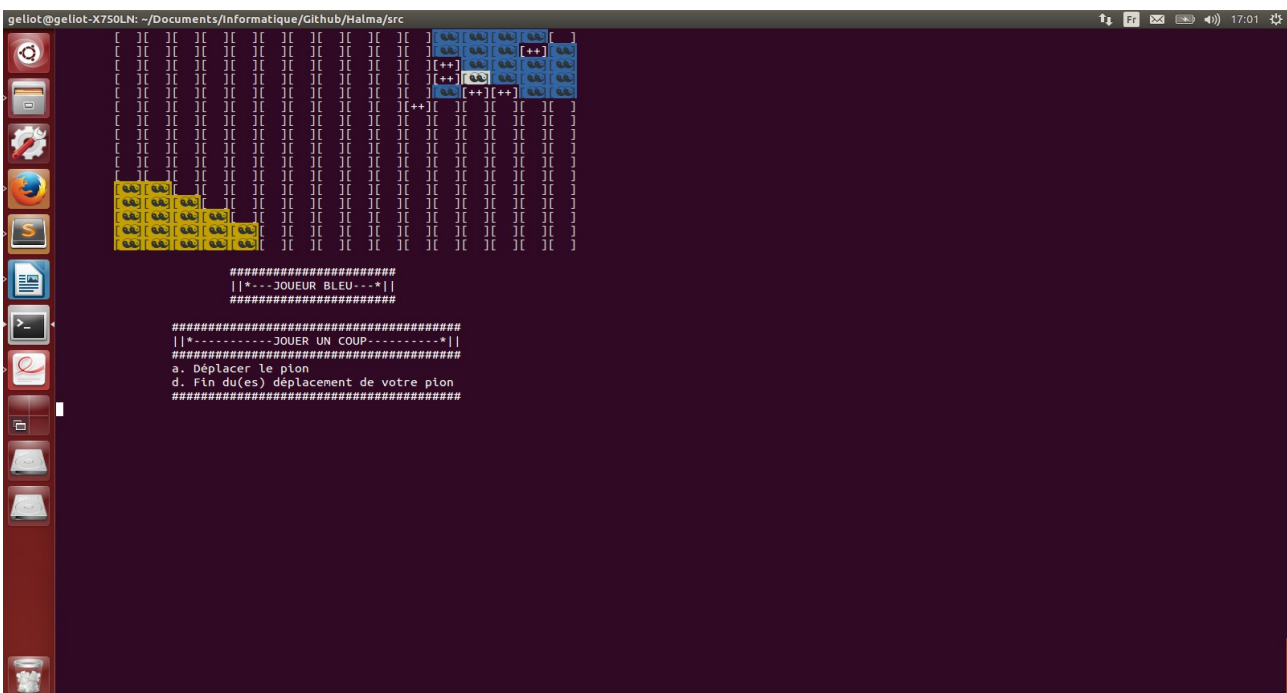
De plus, le fichier de sauvegarde est écrit en binaire.

Exemple de sauvegarde :



```
1 0200 0000 0100 0000 0100 0000 0000 0000
2 0c00 0000 0100 0000 0100 0000 0200 0000
3 0d00 0000 0b00 0000 0200 0000 0d00 0000
4 0200 0000 0d00 0000 0b00 0000 0500 0000
5 0a00 0000 0000 0000 0100 0000 0000 0000
6 0000 0000 0000 0000 0d00 0000 0000 0000
7 1800 0000 0b00 0000 0200 0000 0000 0000
8 0300 0000 0000 0000 0000 0000 0100 0000
9 0100 0000 0100 0000 0d00 0000 0700 0000
10 0100 0000 0d00 0000 0100 0000 0d00 0000
11 0700 0000 0300 0000 0b00 0000 0000 0000
12 0100 0000 0000 0000 0000 0000 0000 0000
13 0f00 0000 0300 0000 2400 0000 0d00 0000
14 0500 0000 0000 0000 0300 0000 0000 0000
15 0000 0000 0100 0000 0100 0000 0300 0000
16 0c00 0000 0b00 0000 0300 0000 0c00 0000
17 0300 0000 0c00 0000 0b00 0000 0500 0000
18 0a00 0000 0000 0000 0100 0000 0000 0000
19 0000 0000 0000 0000 0c00 0000 0200 0000
20 1700 0000 0b00 0000 0300 0000 0000 0000
21 0300 0000 0000 0000 0000 0000 0100 0000
22 0100 0000 0400 0000 0b00 0000 0b00 0000
23 0400 0000 0b00 0000 0400 0000 0b00 0000
24 0b00 0000 0500 0000 0a00 0000 0000 0000
25 0100 0000 0000 0000 0000 0000 0000 0000
26 0b00 0000 0100 0000 1400 0000 0a00 0000
27 0200 0000 0000 0000 0300 0000 0000 0000
28 0000 0000 0100 0000 0100 0000 0100 0000
29 0b00 0000 0500 0000 0100 0000 0b00 0000
30 0100 0000 0b00 0000 0500 0000 0200 0000
31 0a00 0000 0000 0000 0100 0000 0000 0000
32 0000 0000 0000 0000 0a00 0000 0400 0000
33 2000 0000 0c00 0000 0600 0000 0000 0000
34 0300 0000 0000 0000 0000 0000 0100 0000
35 0100 0000 0100 0000 0a00 0000 0500 0000
36 0100 0000 0a00 0000 0100 0000 0a00 0000
37 0500 0000 0200 0000 0a00 0000 0000 0000
38 0100 0000 0000 0000 0000 0000 0000 0000
39 0c00 0000 0000 0000 1500 0000 0b00 0000
40 0100 0000 0000 0000 0300 0000 0000 0000
41 0000 0000
```

Les coups possibles sont également disponibles. Ceux-ci sont indiqués par des croix blanches.



```
#####
||*---JOUEUR BLEU---*||
#####

#####
||*-----JOUER UN COUP-----*||
#####

a. Déplacer le pion
d. Fin du(es) déplacement de votre pion
#####
```

Les IA ont été créés. Cependant ils ne jouent pas de façon aléatoire.

L'ordinateur calcule pour chaque pion son importance. Plus celle-ci est grande, plus la probabilité qu'il le déplace augmente.

Il prend ensuite celui dans l'importance est la plus grande et le déplace en suivant les règles du jeu.

## **Bilan des Contributeurs :**

### Geliot Quentin :

Ce fut mon premier projet informatique. J'ai eu des difficultés avec les pointeurs, allocations dynamiques, makefiles, chaînes de caractères, listes, piles...

L'architecture MVC m'a permis de comprendre tout de suite comment aller s'organiser le projet : construction du modèle, de la vue puis du contrôleur qui utilise le modèle et la vue. J'ai rapidement compris qu'avec la dissociation de ces 3 objets permet une plus grande maniabilité de réalisation

Pour le bon fonctionnement du jeu, les pointeurs sont beaucoup utilisés. J'ai pu bien comprendre leur principe et les manipuler correctement.

De même pour les listes. Nous en avons créé quelques-unes. Leur création et leur manipulation m'a permis de comprendre leur fonctionnement.

Le cours d'algorithmie m'a aussi permis de bien saisir les listes et les piles.

Mes recherches sur internet m'ont également permis d'en savoir plus sur les chaînes de caractères, et la difficultés de travailler avec (saisie par l'utilisateur, manipulation dans le code...)

La lecture des makefiles (réalisé par mon binôme et expliqué par celui-ci) m'a permis de comprendre comment les lire et les écrire.

### Tendry Rakotondramasy :

Développer un jeu n'est pas une nouveauté pour moi, en revanche c'était l'occasion de pouvoir utiliser un langage permissif, mais nécessitant beaucoup de rigueur, au sein d'un projet plus conséquent que ceux réalisés dans mon précédent cursus.

J'ai été beaucoup ralenti par la gestion de la mémoire propre au langage C (segmentation faults et allocations dynamiques), le temps de travail a été de plus de 50 % dépensé à la correction, réparation et mise en place de structures dynamiques correcte dans ce langage. Je trouve personnellement que cela nous a même beaucoup trop ralenti, aboutissant au fait que nous n'avons par exemple pas pu gérer la portabilité complète sous Windows.



La mise en place du MVC nous a par contre permis d'avoir un projet relativement solide et stable, cela nous a permis de mettre en place la sauvegarde binaire ou encore la gestion complète des tours et des coups.

Je suis plutôt satisfait de l'ensemble du projet et ai pu confier des tâches importantes et complexes à mon binôme sans trop avoir à m'en soucier.