Projet : Mastermind

CROCE Baptiste , MARZOOK Cyril

18 Janvier 2019

Table des matières

I Introduction

Le principe de ce projet était de créer un Mastermind, et une ia capable d'y jouer également. Dans ce jeu, le joueur a pour but de deviner un code de 4 pions avec 6 couleurs possibles en 10 coups maximum. Nous connaissions déjà le jeu avant ce projet, ce qui nous a permis de nous plonger dans le code plus rapidement.

Nos motivations étaient de pouvoir nous améliorer en programmation fonctionnelle car c'était tout nouveau pour nous. Ainsi que de travailler en autonomie efficacement.
Nos objectifs principaux étaient de réussir a coder les ias et les différents modules sans récursivité apparente mais seulement avec les fonctions du module List. De plus nous tenions a faire un mode joueur contre joueur pour pouvoir faire découvrir le jeu a nos proches et amis.

Mastermind

Nous tenons à remercier M. George d'avoir répondu à toutes nos questions et de nous avoir aiguillé dans nos choix tout au long de ce projet.

II Structure du projet

Notre projet a une structure plutôt compartimenté car nous avons fait plusieurs choix.
En effet, premièrement nous avons découpé les différentes intelligences artificielles en plusieurs modules afin de pouvoir bien les distinguer et comprendre leur fonctionnement en détail. Toutes ces ia sont regroupées dans le module ia et leur utilisation est gérée également à travers ce module. De plus, l'interface graphique fait office de programme principale, on lance le jeu en l'appelant à travers un Makefile.
Pour jouer à notre mastermind, il faut donc lancer la commande make puis l'exécutable de cette façon :

Listing 1 : Compilation

```
make
./mastermind string -> int -> int -> bool
```

Légende :
le paramètre 1 correspond au nom du joueur
le paramètre 2 correspond au nombre de tentatives maximum
le paramètre 3 correspond au nombre de parties
le paramètre 4 correspond à la reponse voulue : automatique = false et manuelle = true

Dans notre jeu il y a deux modes de jeu, Joueur vs IA et Joueur vs Joueur. Cela veut dire que si vous voulez faire 4 parties vous allez tout d'abord choisir le mode de jeu pour deux parties et pour les deux suivantes vous aurez à nouveau le choix entre les différents modes. Cela permet donc de tester tout les niveaux sans forcément faire un grand nombre de parties et sans relancer le jeu.

III Explication des choix algorithmiques

III.1 Partie IA

Notre objectif principal pour l'ia était de pouvoir avoir plusieurs niveaux de difficultés de jeu. Pour cela nous avons mis en place quatre niveaux différents.
De plus, un de nos objectifs était de réaliser toutes ces ia sans aucune récursivité apparente si ce n'est celle des fonctions du module List. Cela nous a donc pris beaucoup de temps mais nous avons finalement réussi et nous en sommes plutôt fier. Afin de bien séparer tout ces programmes, nous avons créé un module différent pour chaque ia , un pour l'algorithme de Knuth, un autre pour l'ia naïve, pour l'ia de niveau 2 et un dernier pour l'ia de niveau 3.
Désormais intéressons nous à chacune des ia que nous avons pu coder.

L'algorithme de Knuth

L'algorithme de knuth est basé sur une fonction filtre et un Minmax. Cependant dans notre projet, nous avons appliqué un Maxmin pour traduire cet algorithme mais cela correspond au Minmax car nous avons traité les listes d'une autre manière pour diminuer le temps d'exécution du programme. En effet Knuth propose de prendre pour chaque code possible son pire cas en prenant le plus petit nombre d'éléments qu'il supprime dans la liste des codes possibles. Et bien nous avons pris non pas le nombre d'éléments qu'il supprime mais le nombre d'élements qu'il reste. Donc s'il fallait prendre le plus petit nombre d'éléments qu'il supprime dans la liste des codes possibles nous avons pris le plus grand nombre d'éléments restant dans cette liste de codes possibles. Cela revient donc exactement au même nous obtenons les mêmes résultats mais de cette manière l'algorithme tourne beaucoup plus vite.

Finalement, nous avons établie une moyenne du nombre de coups pour deviner le code secret et le temps associé avec plus de 500 codes secrets aléatoires et nous avons obtenu une moyenne de 4,399 coups par jeu pour 3,5 secondes d'exécution.
Voici un extrait de code traduisant l'élement le plus important de cette algorithme. c'est la fonction implémentée à laquelle lui est appliquée un List.fold_left. Le principe est de ne prendre que les codes de la liste des possibles p qui ont la même réponse que le dernier code essayé. C'est donc la fonction qui permettra de filtrer la liste des possibles et de calculer les poids.

Listing 2 : Filtre

```
fun a t -> if (reponse t essai) = Some(x,y) then t :: a else a
```

L'ia naïve

Cette ia comme son nom l'indique est naïve. En effet elle ne fait que choisir un code aléatoire parmi la liste de tout les codes possibles.

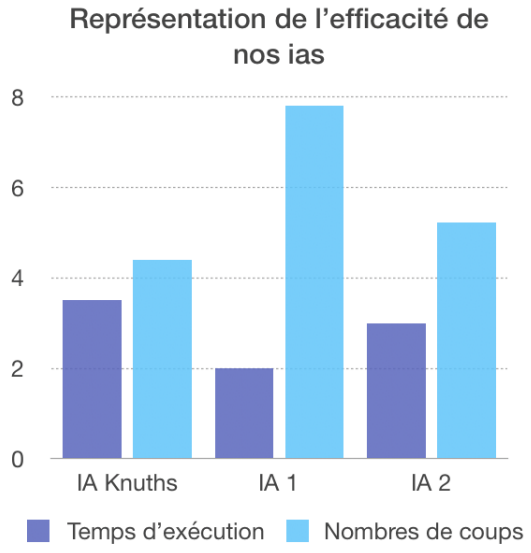
L'ia de niveau 2

L'ia de niveau 2 fût la plus dure a réaliser dans la mesure où coder une ia qui gagne toujours en un faible nombre de coups est plus simple et est déjà fait. Il nous fallait une ia pas trop forte pour avoir une réelle gradation des niveaux. C'est-à-dire une ia qui découvre le code quasiment tout le temps mais contre laquelle il serait possible de gagner. En effet il sera très difficile de battre l'algorithme de Knuth ou encore notre ia de niveau 3.
Le principe de cette ia est le suivant, prendre dans la liste des possible non pas tout les codes donnant la même réponse que l'essais précédent mais de prendre que le nombre de pions bien placés en considération. Ensuite on obtient une nouvelle liste de code possible. Désormais pour choisir un code, on compare le nombre de couleur qu'il a en commun avec l'essai, celui avec le plus grand nombre de couleur en commun et choisi et ainsi de suite.
Il nous a donc fallu jauger expérimentalement, a tâtons avant de voir qu'il fallait prendre ces paramètres la en considération. Et finalement nous avons obtenu les résultats escomptés, avec une moyenne de 7,8 coups pour deviner le code secret (avec 500 codes secrets testés) en environ 2 secondes.

L'ia de niveau 3

Enfin nous avons l'ia de niveau 3. Cette ia trouve le code secret en moins de coups que l'ia précédente. Elle revient a appliquer la fonction filtre de l'algorithme de Knuth puis ensuite au lieu d'appliquer le fameux Minmax, choisir un code au hasard parmi la nouvelle liste des codes possibles obtenue. Ses résultats sont plutôt concluant, une moyenne de 5,22 coups sur 500 codes secrets choisis au hasard avec un temps d'exécution d'environ 2,5 secondes.

Pour mieux visualiser les résultats de ces ias voici un graphique traduisant leur nombre moyen de coups et leur temps d'exécution :



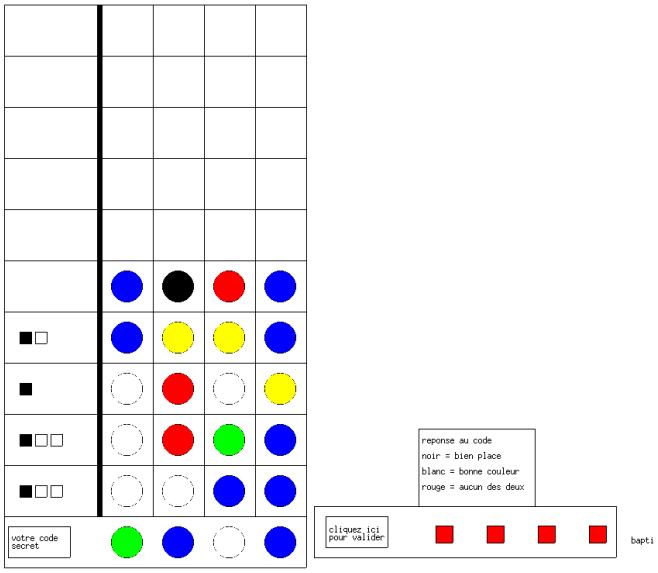
Graphique

III.2 Partie graphique

Pour accompagner ces ias et pouvoir jouer, nous avons décidé d'utiliser le module Graphics. En effet un affichage avec le terminal était envisageable, mais peu joli et pas du tout ergonomique. C'est à partir de ce dernier point que nous avons choisi d'utiliser ce module. Ce dernier a été assez compliqué à mettre en place mais au final se montre assez efficace, pour favoriser l'ergonomie. Par la suite nous allons expliquer nos choix par rapport à notre interface.

Grille

Pour cela, nous avons choisi de rester extrêmement proche de la version physique. En effet on créer une grille, avec le nombre de tentative et le nombre de pions choisi. Les anciennes tentatives et reponses sont toujours visible en dessous du coup à effectuer. Les réponses sont donc logiquement placées à gauche avec, un carré noir pour signifier qu'un pion est bien placé, un blanc si seule la couleur est bonne. Pour le choix des couleurs, nous avions le choix entre un glisser-déposer ou un clic pour changer la couleur de chaque pion puis un bouton pour valider. Nous avons choisi la deuxième option, car nettement moins compliqué à faire et tout aussi agréable. La même méthode est utilisée pour rentrer les réponses à la main mais avec seulement noir (bien placé), blanc (bonne couleur seulement), rouge (aucun des deux).



Interface graphique

Menu

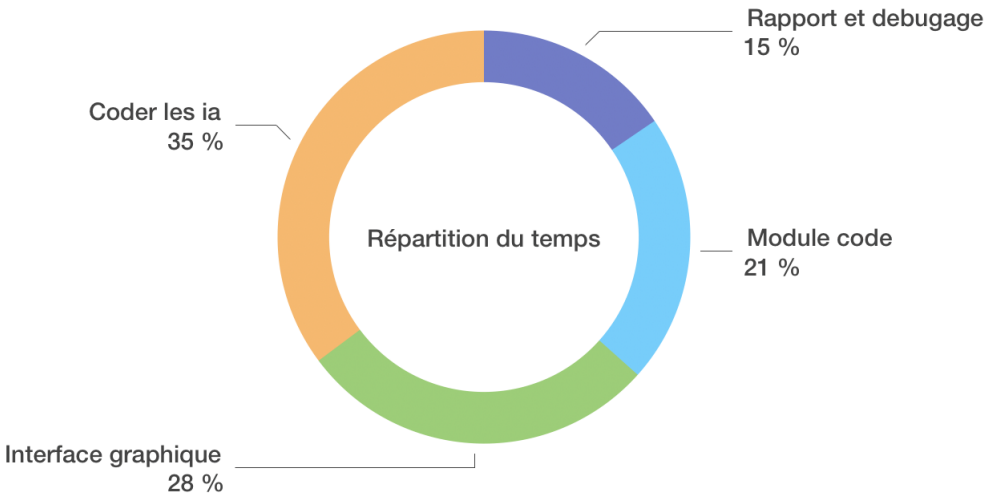
Pour les menus, on a décidé de tout faire avec des clics, que ce soit pour les choix de mode ou de niveau. Nous avons également pris l'initiative d'attendre un clic de l'utilisateur lorsque l'ia devine et que la reponse est automatique pour afficher son choix et la reponse correspondante, car nous n'avons pas réussi à mettre un temps d'attente entre chaque tentatives. Cela permet en plus à l'utilisateur de comprendre les réponses, et de choisir le rythme de l'avancée.

IV Répartition du travail

Dès le début de ce projet nous avions conscience de l'importance de bien répartir le travail à faire au sein de notre groupe. Pour cela nous avons codé le module Code à deux pour pouvoir bien comprendre comment fonctionnera tout ce qui en découlera. C'est-à-dire la réalisation de plusieurs intelligences artificielles et de l'interface graphique bien sûr.

Ensuite après avoir fini de coder le module Code nous avons coupé le travail en deux. Nous avons décidé que d'une part l'un s'occupera de l'interface graphique (d'ailleurs nous disposons d'une interface graphique mais aussi d'une interface terminale), et l'autre de l'intelligence artificielle et de l'algorithme de Knuths. De ce fait, nous nous sommes occupés ensemble de relier l'interface graphique aux différents modules de jeux car il fallait mettre en commun ce que nous avions fait.

Finalement, voici un graphique montrant le temps que nous avons mis pour chaque étape de ce projet :



Graphique

V Utilisation de Gitlab

Durant ce projet, nous devions utiliser le Gitlab de l'eisti. Cela ne nous a pas déranger ni pris plus de temps que d'habitude car dès l'année dernière nous réalisions nos projets en utlisant Github.
Nous n'étions donc pas dépaysés par cela. Nous avons procédé de la sorte. Nous avons créé une première branche "règle" dans lequel nous avons mis principalement le module code imposé. Ensuite nous avons créé deux dernières branche pour pouvoir travailler en simultané sans avoir à gérer les éventuelles erreurs de commits. Il y a donc la branche ia pour développer toutes les ia que nous avons pu et enfin la branche graphic dans laquelle nous avons développer l'interface graphique. Une fois les deux parties finies nous avons merge les branches dans la branche principale pour pouvoir commencer à relier l'interface graphique avec les différentes ia. Nous avons beaucoup de commits pour ce projet car habitant loin l'un de l'autre, nous nous montrions nos codes sur Gitlab.

VI Bilan

Pour résumer, au cours de ce projet nous avons réussi a faire fonctionner le Mastermind ainsi que 4 ias dont celle de Knuth. De plus nous avons mis sur pied une interface graphique parfaitement fonctionnelle en plus de celle sur le terminale. Et nous avons également réalisé deux modes de jeu, ia vs joueur et le joueur contre joueur.
Ce projet fut bénéfique pour plusieurs points, tout d'abord et le plus important il nous a permis de nous familiariser avec Ocaml, et plus généralement avec l'algorithmique fonctionnelle. L'utilisation du module List a la place de la récursivité était également difficile a mettre en place pour toutes les fonctions mais nous nous y sommes habitué et cela nous procuré un très bon entrainement.
La répartition du travail a été très importante puisque le projet n'aurait jamais pu etre fini sans nous pensons l'avoir répartie équitablement que ce soit au niveau de la difficulté et du temps.
L'utilisation du Git a aussi été très utile, malgré le fait que nous l'utilisions déja. Ce qui nous a semblé le plus intéressant a été le codage de l'ia, principalement le niveau 3 puisque nous avons du le créer de toute pièce contrairement a Knuth.
Le module Graphics c'est avéré plus puissant que nous l'espérions, mais assez difficile a prendre en main et pas du tout naturel. Le temps pour en comprendre les principes a été quasiment le même que le temps mis a créer la partie graphique elle même. Pour conclure ce que nous avons pensé de ce projet, il a été tr intéressant, pour tous les points cités précédemment, mais nous devons admettre que nous avons été beaucoup plus a l'aise que sur le Qwirkle de l'année dernière et c'est pour cela que nous avons décidé de faire tout les bonus. Malgré cela, ce fût très enrichissant.