Projet Scala : Puissance-4

Pierre Cerantola, Réda Eloualladi, Clara Nguyen

ING2 GSI-02

Programmation Fonctionnelle

Pr. Bartholomew George

12 avr. 2020

Introduction

L’objectif de ce projet est d’écrire un programme en Scala permettant à deux joueurs de s'opposer lors d'une partie de Puissance 4. Chaque joueur peut être soit un humain soit une IA et la taille du plateau est variable.

Nous étions 3 sur ce projet : Pierre Cerantola, Réda Eloualladi et Clara Nguyen. Nous nous sommes départagés le travail de façon dynamique. Nous avons Partagé le projet en 3 étapes et nous nous sommes partagés les fonctions à coder à chaque étape, puis nous nous sommes entraidés en cas de difficultés. La première étape a consisté à créer l’interface de jeu pour un joueur humain contre un ordinateur qui joue aléatoirement. La deuxième étape a consisté à ajouteur la partie de l’IA consistant à évaluer chaque coup possible. Enfin, la dernière étape a consisté à implémenter l’IA finale, avec l’algorithme MinMax.

1. Mode d'Emploi

La version de Scala que nous avons utilisé est : **2.11.12**. Afin de construire notre projet, nous avons utilisé l'éditeur Visual Studio code. Pour compiler et créer dans le même temps un fichier exécutable, nous exécutons la commande **scalac Puissance4.scala -d Puissance4.jar**, et pour exécuter cet exécutable .jar, nous exécutons la commande **scala Puissance4.jar**. Pour simplement compiler, nous exécutons la commande **scalac Puissance4.jar**.

Après avoir exécuté le programme, il suffit de suivre les indications à la console. Le premier choix est celui de la création ou de l'importation de la partie. En choisissant l'option **1**, l'utilisateur devra spécifier la taille du nouveau plateau à créer, et en choisissant l'option **2**, l'utilisateur devra rentrer sur une seule ligne dans la console le plateau à importer, en espaçant chaque nouvelle ligne du plateau par un **espace**, et en nommant chaque jeton par **R** pour "rouge", **J** pour "jaune", et **.** pour une case vide. Par exemple, pour importer le plateau de la page 2 du sujet, il suffit de rentrer "**RJJJRJR JRJRRJR JRJRJJ. .R.R... ....**" dans la console.

Après la création ou l'importation du plateau, l'utilisateur est amené à choisir les 2 types d'adversaires qui s'affronteront au cours de la partie : pour le joueur Jaune et le joueur Rouge, l'utilisateur peut choisir entre un humain, une IA de niveau 1 (une IA qui joue aléatoirement), une IA de niveau 2 (une IA qui joue en regardant le meilleur coup prochain), et une IA de niveau 3 (une IA qui utilise l'algorithme MinMax). Note : si l'utilisateur sélectionne pour un des joueurs l'IA de niveau 3 (i.e. MinMax), la profondeur de la recherche du score des coups lui sera demandée à la suite. Plus la profondeur est importante, et plus la qualité de jeu de l'IA sera haute, au détriment du temps que prendra chaque coup pour être joué par l'ordinateur (note : la profondeur doit être un nombre strictement positif).

Après avoir configuré les joueurs qui s'affronteront au cours de la partie, la partie peut enfin commencer. Le choix du premier joueur à commencer est déterminé aléatoirement. A chaque coup, le programme demande au joueur humain (s'il y a un joueur humain) la colonne à jouer, puis affiche le nouveau plateau. Lorsque les 2 joueurs sont des IA, le programme déroule

automatiquement la suite des plateaux à chaque coup. Dès qu'un coup est gagnant, la partie s'arrête, le plateau gagnant est affiché, ainsi que la couleur du joueur gagnant.

2. Choix de Conception et Design

Nous avons décidé d'implémenter ultimement une IA avec MinMax. Cependant, avant cela, nous avons créé les 3 types d'IA suivants.

a. Aléatoire

Aléatoire joue seulement un coup au hasard à son tour. Elle n'est pas performante, il s'agissait seulement d'une IA d'essai.

b. Meilleur Coup

Meilleur coup à cet instant: une grande partie de cette IA est utilisée dans la fonction Minmax. Elle calcule le score de chaque coup possible dans un tour. Pour ce faire, elle simule un coup et y attribue des bonus et des malus en fonction de l'organisation des pièces dans le plateau. Il coupe alors le tableau en fenêtres, qui sont des suites de 4 cases consécutives formant une ligne, une colonne ou une diagonale.

Si une fenêtre comporte 4 jetons du joueur, un bonus maximum est attribué. Si une fenêtre comporte 3 jetons du joueur et une case vide, un bonus de rang 2 est attribué. Si elle comporte 2 jetons du joueur et 2 cases vides, un bonus de rang 3 est attribué. Enfin, si elle comporte 1 jetons du joueur et 3 cases vides, un bonus de rang 4 est attribué.

Inversement, le même système de fenêtres permet d'évaluer les malus en fonction de l'organisation des pièces de l'adversaire. Si une fenêtre comporte 3 jetons de l'adversaire et une case vide, un bonus de rang 2 est attribué. Si elle comporte 2 jetons de l'adversaire et 2 cases vides, un bonus de rang 3 est attribué. Enfin, si elle comporte 1 jetons de l'adversaire et 3 cases vides, un bonus de rang 4 est attribué.

c. MinMax

Il s'agit de l'implémentation de l'algorithme Minmax avec différents types de difficultés. Minmax est l'algorithme que nous connaissons le mieux pour résoudre un jeu à 1 contre 1. Cette IA construit (conceptuellement) l'arbre de tous les plateaux possibles à partir d'une configuration de plateau donnée, avec une profondeur passée en argument. On associe ensuite à chaque plateau

aux extrémités de notre arbre (i.e. ses feuilles) un score, en fonction de l'alignement de jetons du joueur à maximiser et du joueur à minimiser. Ce score est ensuite remonté jusqu'au niveau de l'arbre qui vient juste après la racine, de façon à déterminer le coup qui est le plus rentable à jouer. Plus la profondeur choisie sera importante, plus le coup aura de chance d'être gagnant sur le long terme (en effet, avec une profondeur plus élevée, on explore davantage de choix possibles et donc de scores possibles).

3. Avantages et Limites du Programme

a. Avantages

Notre IA MinMax est performante dans la mesure où nous avons joué plusieurs parties contre elle et n'avons jamais réussi à la battre. De plus, nous pouvons créer et importer/exporter des tableaux de tailles généralisées n*m. Il est aussi possible de se faire s'affronter 2 IA. Enfin, nous avons créé différents difficultés pour l'algorithme Minmax en sélectionnant différents degrés de profondeur de l'arbre parcouru par l'algorithme.

b. Pistes d'amélioration

Nous aurions pu implémenter une interface graphique à notre programme plutôt que de conserver la console comme moyen d'affichage. Nous aurions pu aussi ajouter l'élagage Alpha-Beta à notre algorithmes Minimax afin d'optimiser l'efficacité du programme en terme de temps d'exécution et de ressources utilisées sur de grands tableaux ou à une grande profondeur. De plus, nous aurions pu implémentant d'autres types d'algorithmes connus d'IA pour les jeux de 1 contre 1 comme Monte Carlo Tree Search. A ce moment, nous aurions été en mesure de comparer d'efficacité des différents algorithmes en faisant jouer un type d'IA contre un autre type d'IA.

Conclusion

L’implémentation du Puissance-4 avec une IA a été un succès. Notre programme permet que chaque joueur soit une IA ou un humain et permet d’importer un plateau. Il permet aussi de jouer avec 3 IA différentes, la dernière, MinMax, ayant une difficulté paramétrable.