Puissance 4
Cahier des Charges

CHING Kahnvud BOREDON Anthony
RICO Quentin MANSARD Yann

15 mars 2016

Table des matières

1 Introduction **3**

2 Jeu du Puissance 4 **3**
2.1 Définition du Puissance 4 3
2.2 Motivation du projet 3
2.3 Contraintes 4
2.4 Exigences non fonctionnelles 4

3 Organigramme et fonctionnalités **4**
3.1 Organigramme 4
3.2 Liste des fonctionnalités du programme 5

4 Estimation du coût **6**

5 Conclusion **7**

1 Introduction

Les jeux de sociétés existent depuis la préhistoire [1] et, selon la culture des pratiquants et selon l'époque, se retrouvent sous diverses formes mais ont tous un même but : la prise de décisions autour de règles prédéfinies entre joueurs afin de gagner. Ce qui permet de départager les joueurs au cours de la partie est la réflexion, en effet chaque joueur doit "élaborer une stratégie d'actions" (Gonzàlez et Crête, 2006, p.1) et s'adapter selon les différentes stratégies de chacun des autres adversaires [2].

Avec l'explosion de l'informatique dans les années 80, de nombreux jeux de sociétés ont étés adaptés en programme et sont jouable sur ordinateur. L'utilisation de ce type de jeux informatique a permis la création d'un nouvel adversaire : l' "Intelligence artificielle". L' "Intelligence Artificielle" appelée "IA" dans les jeux vidéos "vise à créer ou simuler [...] une intelligence comparable à l'homme ou davantage spécialisée" [3]. C'est à dire que l'implémentation d'une "IA" est de simuler la stratégie d'un joueur ou d'une entité. Il existe de nombreux jeux stratégiques permettant d'implémenter une "IA" qui joue de manière parfaite, c'est le cas du Puissance 4.

2 Jeu du Puissance 4

2.1 Définition du Puissance 4

Le jeu du puissance 4 est un jeu inventé et commercialisé en 1974 par la Milton Bradley Company et fait partie de ces jeux dit de "stratégie combinatoire abstrait" [4]. Il se joue à deux joueurs, le but étant d'être le premier à aligner 4 jetons de la même couleur dans une grille de 6x7 positions en jouant au tour par tour (figure 1).

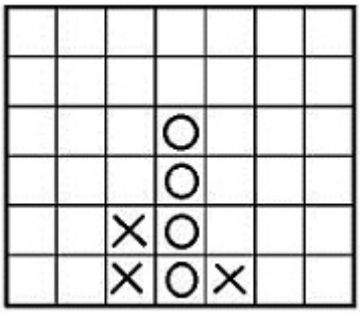

FIGURE 1 – Exemple d'une partie de Puissance 4

Pour aligner 4 jetons, il existe trois manières : verticalement, horizontalement et sur la diagonale. Ce jeu a été complétement résolu par James Allen et Victor Allis [5], indépedemment l'un de l'autre, en 1988. On sait désormais que le joueur qui commence la partie peut jouer de manière parfaite afin d'être sûr de gagner.

2.2 Motivation du projet

Le but de ce projet est de créer une "IA" permettant d'adapter la stratégie de la thèse de Victor Allis afin de gagner à tous les coups si celle-ci commence à jouer ou d'obtenir au moins un match nul lorsque l'adversaire commence (sauf dans le cas où l'adversaire joue de manière parfaite).

Ce projet s'adresse donc à un utilisateur souhaitant affronter un ami ou cette "IA" lors d'une partie de puissance 4.

2.3 Contraintes

Ce projet exige donc l'implémentation d'une interface permettant :

- de pouvoir jouer contre une autre personne ou contre l' "IA".
- à tous moments, de pouvoir consulter les statistiques des parties (victoires et défaites) que ce soit contre l' "IA" ou contre un autre joueur.
- au cours d'une partie, de pouvoir sauvegarder à tous moments, et par conséquent il devra également être possible de charger une partie.

L'intelligence artificielle implémentée doit pouvoir :

- gagner à tous les coups si elle commence la partie.
- s'adapter et obtenir le meilleur résultat possible si elle ne commence pas.

2.4 Exigences non fonctionnelles

L'interface graphique doit être intuitive et simple d'utilisation. Le temps de réponse de l' "IA" (le temps qu'elle met pour jouer un coup) doit être assez courte. Le programme doit également pouvoir fonctionner sous n'importe quel système d'exploitation.

3 Organigramme et fonctionnalités

3.1 Organigramme

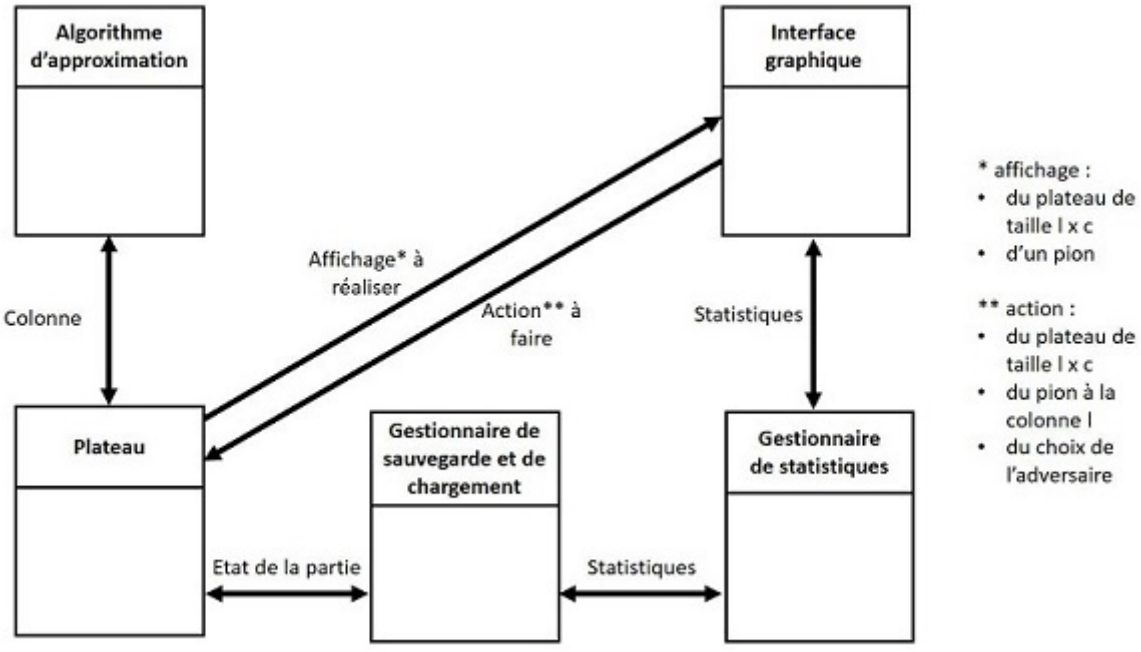

FIGURE 2 – Organigramme représentant les modules du logiciel

3.2 Liste des fonctionnalités du programme

Interface graphique :

- Afficher le menu() : affiche le menu avec des boutons permettant de choisir si on décide de jouer contre l' "IA", contre un autre joueur, de pouvoir charger une partie, d'accéder aux statistiques, de commencer la partie, de choisir la taille du plateau et de lancer une partie.
- Afficher le plateau() : affiche le plateau de jeux, un bouton de sauvegarde et un bouton pour revenir au menu.
- Afficher les statistiques() : affiche les statistiques de victoire et de défaite.

Gestionnaire des statistiques :

- Calcul du ratio homme contre machine() : Pourcentage de victoire de l'homme, qui commence la partie, contre la machine.
- Calcul du ratio machine contre homme() : Pourcentage de victoire de la machine, qui commence la partie, contre l'homme.
- Calcul du ratio homme contre homme() : Pourcentage de victoire du premier joueur contre le deuxième.
- Incrémentation des victoires et des défaites()

Plateau :

- Poser un pion(colonne)
- Teste si l'un des deux joueurs à gagner()
- Initialise le plateau()
- Détermine qui joue()

Gestionnaire de sauvegarde et de chargement :

- Sauvegarde la partie(plateau)
- Charge la partie()
- Sauvegarde les statistiques(statistiques)
- Charge les statistiques()

Algorithme d'approximation :

- Détermine le meilleur coup() : Fonction qui détermine le meilleur coup à jouer respectant les règles de bases expliquées dans dans le pseudo-algorithme suivant :
  - Si possibilité de gagner alors jouer la case gagnante
  - Sinon si possibilité de bloquer un coup gagnant alors jouer la case bloquante
  - Sinon jouer case determinée par Algorithme minimax-alphabeta()

- Algorithme minimax-alphabeta() : Fonction qui crée un arbre en fonction des poids des meilleurs coups possible tant élagué par la règle des alpha-bêtas et renvoyant un coup à jouer :
  - Pour chaque coup possible
  - Déterminer poids avec Algorithme des règles de bases()
  - Renvoyer coup avec poids le plus fort

— Algorithme des règles de bases() : Fonction qui utilise et fait des tests avec chacunes des règles (Claimeven, Aftereven, Baseinverse, Lowinverse, Vertical) pour déterminer un poids avec un ordre de priorités des règles :
  — Si Claimeven sur une case ayant le poids le plus fort alors renvoyer poids de cette case + valeur d'un Claimeven
  — Si Aftereven sur une case ayant le poids le plus fort alors renvoyer poids de cette case + valeur d'un Aftereven
  — Si Baseinverse sur une case ayant le poids le plus fort alors renvoyer poids de cette case + valeur d'un Baseinverse
  — Si Lowinverse sur une case ayant le poids le plus fort alors renvoyer poids de cette case + valeur d'un Lowinverse
  — Si Vertical sur une case ayant le poids le plus fort alors renvoyer poids de cette case + valeur d'un Vertical

Claimeven : Règle disant qu'on peut obtenir toutes cases paires au-dessus d'une case impaire qui serait jouable directement.
Aftereven : Règle disant que l'on peut obtenir une case si l'on a une menace d'aligner 4 pions sur une case paire.
Baseinverse : Règle disant que si il y a une menace pouvant être réalisée en plaçant un pion horizontalement à côté ou â une case de distance de ces deux pions, celle-ci peut être immédiatement bloquée par l'adversaire en jouant l'autre case correspondante.
Lowinverse : Règle disant que si l'on a deux colonnes avec chacune un nombre de cases vides impairs, l'addition de ce nombre de cases vides donne un nombre pair, et par conséquent l'avantage reste à celui qui a des menaces ou attaques sur les cases pairs.
Vertical : Règle disant que si deux pions de l'adversaire sont superposés et qu'il joue sur l'une de ces cases, alors on peut obtenir l'autre case.
Pour cet algorithme (Algorithme des règles de bases()), nous nous sommes restreint à l'utilisation de seulement 5 règles : Claimeven, Aftereven, Baseinverse, Lowinverse et Vertical, car comme il est prouvé par Victor Allis dans sa thèse [5], l'utilisation de seulement ces règles là permettent déjà un résultat quasi-parfait. Certaines de ces règles ne sont que des dérivés d'autres règles ou alors sont très coûteuses (pour le cas du Before par exemple). Il n'est donc pas recommandé si l'on veut avoir un programme avec un rythme de jeu assez fluide.

4 Estimation du coût

Interface Graphique (Personnes : Kahnvud, Quentin) (Heures estimées : 12h) (Lignes de codes total : 260)

— Afficher menu : 130 Lignes de codes
— Afficher plateau : 75 Lignes de codes
— Afficher statistiques : 50 Lignes de codes
— Afficher pion : 5 lignes de codes

Gestionnaire des statistiques (Personnes : Yann) (Heures estimées : 2h) (Lignes de codes total : 33)

— Calcul du ratio homme contre machine : 10 lignes de codes
— Calcul du ratio machine contre homme : 10 lignes de codes
— Calcul du ratio homme contre homme : 10 lignes de codes
— Incrémentation des victoires et des défaites : 3 lignes de codes

Plateau (Personnes : Quentin) (Heures estimées : 3h)(Lignes de codes total : 55)

- Poser un pion : 5 lignes de codes
- Teste si le jeu est gagné : 30 lignes de codes
- Détermine qui joue : 5 lignes de codes
- Initialisation du plateau : 15 lignes de codes

Gestion de la sauvegarde et du chargement (Personnes : Anthony, Yann) (Heures estimées : 12h) (Lignes de codes total : 120)

- Sauvegarde la partie : 40 lignes de codes
- Charge la partie : 40 lignes de codes
- Sauvegarde les statistiques : 20 lignes de codes
- Charge les statistiques : 20 lignes de codes

Algorithme d'approximation (Personnes : Anthony, Quentin, Yann, Kahnvud) (Heures estimées : 48h) (Lignes de codes total : 450)

- Détermine le meilleur coup : 100 lignes de codes
- Algorithme minimax-alphabeta : 150 lignes de codes
- Algorithme des règles de base : 200 lignes de codes

5 Conclusion

Dans ce cahier des charges, nous avons découvert et compris le fonctionnement des stratégies afin d'assurer la victoire à l' "IA" dès lors qu'il commence à jouer. Nous nous baserons donc sur ce cahier des charges pour le reste du projet. Pour ce projet nous estimons donc qu'il faudra environ 77 heures de travail et environ 918 lignes de codes au total. Nous pensons donc que le langage le plus approprié est le Java car les structures de classe nous permette d'implémenter le plateau, les pions et les joueurs et manière évidente que de réaliser des struct en langage C. De plus, nous n'avons pas d'appels systèmes à réaliser dans ce projet, par conséquent, nous ne voyons pas l'utilité du langage C++.

Références

[1] Jean-Marie Lhôte. *Histoire des jeux de sociétés*. Flammarion, 1993.

[2] Patrick Gonzàlez and Jean Crête. *Jeux de société : une initiation à la théorie des jeux en sciences sociales*. Presses De L'universite Laval, 2006.

[3] L'express. Intelligence artificielle. `http://www.lexpress.fr/actualite/sciences/intelligence-artificielle_1550708.html`. [consulté le 30 février 2016].

[4] Wikipédia. Jeu de stratégie combinatoire abstrait. `http://fr.wikipedia.org/w/index.php?title=Jeu_de_strat%C3%A9gie_combinatoire_abstrait&oldid=118087701`. [consulté le 30 février 2016].

[5] Victor Allis. *A knowledge-based approach of connect-four*. PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands, October 1988.