# Projet Puissance 4 : Compte Rendu

Yann Mansard Kahnvud Ching
Anthony Boredon Quentin Rico

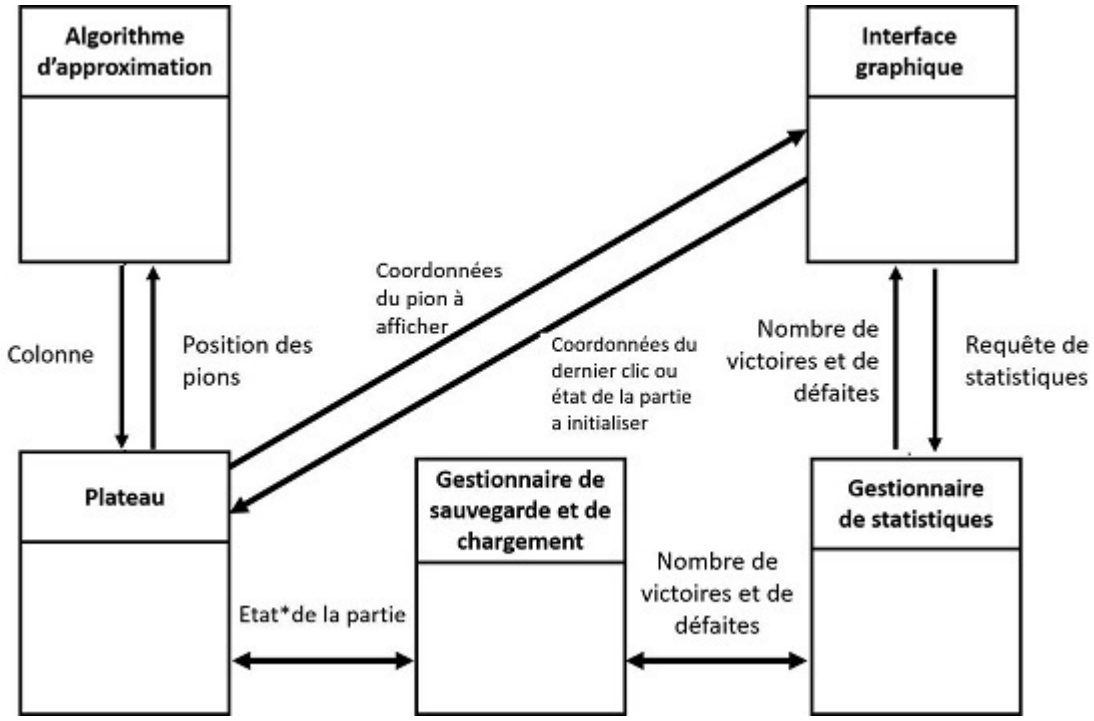
20 mai 2016

# Table des matières

# 1 Introduction

Pour la réalisation d'un projet informatique tel que le puissance 4, nous avons tout d'abord établi un cahier des charges qui nous a permis de mettre en évidence les modules participant à l'élaboration des spécifications. Dans nos spécifications, nous avons donc 5 parties comprenant un module Statistiques, un module Plateau représentant notre plateau de jeu, un module Gestionnaire de Sauvegarde et Chargement permettant de sauvegarder/charger des statistiques et des plateaux, un module interface Graphique permettant de repésenter notre jeu et un dernier module représentant une intelligence artificielle. Tous ces modules sont représentés dans l'organigramme ci-dessous. Pour ces 5 modules nous avons représenté leurs classes, leurs attributs et leurs méthodes. Nous avons donc, à partir de ces informations, pu mettre en place la réalisation du programme représentant l'aboutissement de ce projet.

# 2 Organigramme




# 3 Explication de l'architecture

L'organigramme nous montre 5 modules qui intéragissent les unes avec les autres. Nous allons expliquer dans cette partie, comment les modules, à travers leurs méthodes, intéragissent entre eux.

## 3.1 Plateau

Tout d'abord, dans la partie Plateau, nous faisons appel à la méthode premiereCaseVide et partieTerminee dans la méthode poserPion afin de pouvoir déterminer où et si on peut placer le pion.
Si le pion est posé et qu'il termine une partie, nous avons besoin de pouvoir mettre à jour les statistiques (victoire, défaite, match nul du premier joueur). Par conséquent dans poserPion nous importons le gestionnaire de sauvegarde et chargement pour envoyer des données.

### 3.2 Gestionnaire de statistiques

Cette partie ne réalise que des calculs potentiels sur des entiers et des double et va les renvoyer pour l'utilisation dans d'autres modules.

### 3.3 Interface graphique

Dans ce module, pour les classes FenetreMenu, FenetreJeu, FenetreStatistiques, dessinPlateau, nous importons les packages de java permettant de manipuler une interface graphique : java.awt et javax.swing.
Ce module nécessite toutes ces importations pour la création de boutons et de leurs intéractions.
Nous avons également besoin de gDC.GslPlateau, gDC.GslStatistiques, plateauDeJeu.Plateau, algoApproximation.Solution et gestionnaireStats.Statistiques.
Dans ActionPerformed de FenetreMenu, nous avons besoin de créer et d'appeler le constructeur et les méthodes de Plateau pour pouvoir le représenter graphiquement.
Nous faisons également appel au constructeur et aux méthodes du Gestionnaire de sauvegarde et chargement pour charger une partie sauvegardée et pour pouvoir afficher des statistiques.
Dans le constructeur de FenetreJeu, nous avons un plateau en entrée et nous le manipulons dans cette méthode.
Dans ActionPerformed de FenetreJeu, nous faisons appel à FenetreMenu et nous utilisons GsLPlateau pour sauvegarder le plateau.
Dans FenetreStatistiques nous utilisons les statistiques afin de pouvoir faire des calculs et afficher ces stats.
Pour le constructeur de dessinPlateau et la méthode mouseClicked, nous avons besoin du constructeur et des méthodes du plateau afin de pouvoir créer et afficher notre jeu.
Nous avons également besoin de la classe Solution d'algoApproximation pour créer une Solution et utiliser ses méthodes afin de pouvoir affronter l'intelligence artificielle.

### 3.4 Gestionnaire de sauvegarde et chargement

Dans ce module, nous importons le package comprenant les classes java.io.File et ses classes fils.
Dans Gsl nous avons besoin du package gestionnaireStatiques afin de pouvoir avoir trois attributs statistiques utilisés dans la classe fils GslStatistiques.
Nous avons eu besoin également de la classe plateaudeJeu.plateau pour manipuler un Plateau dans la classe fils GslPlateau.
Dans GslStatistiques nous importons le package gestionnaireStatistiques afin de pouvoir créer des statistiques ou les manipuler dans le constructeur et dans les méthodes ecriture et lecture. On utilise également les ouvertures, fermetures, écritures et lectures à l'aide du package java.io.File dans ces méthodes.
Dans GslPlateau nous utilisons java.io.File pour les mêmes raisons que dans GslStatistiques. Nous utilisons également le constructeur et les méthodes de la classe Plateau dans les méthodes lecture et écriture.

### 3.5 Algorithme d'approximation

Dans les deux fichiers de ce module, nous importons java.util.ArrayList et plateauDeJeu.plateau.
Dans Solution.java, le constructeur manipule des listes, fait des appels au constructeur du plateau et à ses méthodes.
Toutes les méthodes (ColonneAJouer, TriListe, CoupDuDebut, GaseBloquanteDirect et CaseGagnanteDirect) de Solution manipulent également des listes et font des appels au constructeur de Plateau et ses méthodes.

Dans Noeud.java, le constructeur utilise également le constructeur et les méthodes de Plateau, il

a également un attribut ArrayList.
Méthodes de Noeud utilisant seulement les listes : SupprimeElementListe, DetermineMeilleurCoup et AlphaBeta.
Méthodes manipulant seulement le plateau : BaseInverse, LowInverse, Vertical et MenacePairImpair.
Méthodes manipulant à la fois des listes et des plateaux : Aftereven, Claimeven, AlgoRegleDeBase, AjoutElementListe et AlgoMinimax.

# 4 Description du fonctionnement

## 4.1 Éxécution du programme

Le projet est à lancer en double cliquant sur p4.jar dans le dossier sous windows. Dans le terminal windows il suffit de se placer dans le dossier contenant le jar éxécutable p4 est de lancer en tapant p4.jar. Sous Linux il suffit de se placer également dans le dossier et de lancer avec la commande java -jar p4.jar.

## 4.2 Utilisation de l'interface du jeu

Au lancement de l'application, une fenêtre affichant le menu s'ouvre. Dans cette interface, on dispose de trois boutons : Lancer la partie, Charger la partie, Statistiques.
Pour pouvoir lancer une nouvelle partie, deux conditions sont à remplir :

Premièrement, l'utilisateur doit cocher une case parmi les trois proposés, représentant le type de partie désiré :
- Homme VS Machine : Si l'utilisateur veut jouer contre la machine, avec l'utilisateur jouant en premier.
- Machine VS Homme : Si l'utilisateur veut jouer contre la machine, avec la machine jouant en premier.
- Homme VS Homme : Si l'utilisateur veut jouer contre un autre utilisateur.

Deuxièmement, l'utilisateur doit compléter les cases représentant la taille du plateau sur lequel il veut jouer, compris entre un tableau minimun de 4 lignes sur 4 colonnes et un tableau maximum de 8 lignes sur 8 colonnes. (Un tableau de Puissance 4 standard étant de 6 lignes sur 7 colonnes).
Après avoir rempli ces deux paramêtres, l'utilisateur peut lancer la partie.

En cliquant sur le bouton "Lancer la partie", la fenêtre du menu se ferme et la fenêtre du jeu apparaît. Sur cette fenêtre s'affiche le plateau avec les dimensions et le type de partie désiré par l'utilisateur, ainsi que deux boutons :
- Menu : éffface la fenêtre de jeu et affiche la fenêtre du menu.
- Sauvegarder : enregistre le plateau du jeu en cours avec le type de partie, les dimensions du plateau et les pions précédement placés durant la partie. Pour pouvoir mettre un pion dans le plateau, l'utilisateur doit cliquer sur la colonne où il souhaite le placer. Si l'utilisateur joue contre la machine, ce dernier jouera immédiatement après, et donc redonnera le tour à l'utilisateur. Lorsque qu'une condition d'arrêt est detectée (si 4 pions d'une même couleur sont alignés ou si le plateau est rempli), la partie s'arrête et aucun pion ne peut être posé.

En cliquant sur le bouton "Charger la partie", la fenêtre du menu se ferme et la fenêtre du jeu apparaît avec le plateau sauvegardé par l'utilisateur (type de partie, taille du plateau, pion déjà placé).

En cliquant sur le bouton "Statistiques", une nouvelle fenêtre apparaît dans laquelle s'affiche trois pourcentages :
- Le pourcentage de victoire de l'utilisateur contre la machine lorque l'utilisateur commence.
- Le pourcentage de victoire de l'utilisateur contre la machine lorsque la machine commence.
- Le pourcentage de victoire de l'utilisateur jouant en premier face à un autre utilisateur.

# 5 Points délicats de la programmation

Durant la phase de programmation, nous avons relevé quelques points à discuter.

## 5.1 Modifications

Dans notre programme, nous avons effectué certaines modifications :

Dans la classe Plateau, nous avons ajouté deux attributs : int nbTour (prenant en compte le numéro du tour actuel pour savoir qui est entrain de jouer) et int typePartie (permettant de savoir si le joueur commence ou la machine, ou si deux joueurs jouent).
Nous avons aussi importé GslStatistiques pour modifier le fichier de statistiques lorsqu'une partie est terminée.
Dans la classe Statistiques, nous n'utilisons pas la fonction incrémentation, car ce rôle est effectué par la fonction poserPion dans la classe Plateau. La fonction ratio a besoin d'un attribut appelé choix permettant de savoir le type de partie joué et donc connaître quelles statistiques modifier.

Dans la classe GslStatistiques et GslPlateau, nous importons java.io.IOException pour gérer les problèmes et lever les exceptions lors de sauvegarde et de chargement. Nous avons retiré les fonctions sauvegarde et charge de ces deux classes parce qu'elles sont utilisées par la classe Gsl.

Dans la classe dessinPlateau, nous avons rajouté l'attribut Plateau monPlateau qui est obligatoire pour pouvoir dessiner le plateau. Nous n'avons pas besoin de int joueur, car il n'est pas utile dans cette classe.

Dans la classe FenetreMenu, nous n'utilisons pas les implémentations ItemListener et TextListener car elles demandent de créer deux nouvelles fonctions qui ne sont pas nécessaire. Nous n'utilisons pas l'attribut FenetreJeu Fenetre1 (et l'attribut FenetreMenu Fenetre2 dans la classe FenetreJeu) car ils ne sont pas nécessaire pour afficher les fenêtres du programme.

Dans la classe Noeud, nous avons ajouté un attribut pour deux fonctions : l'attribut int i dans la fonction AjoutElementListe, représentant la colonne actuellement analysée, et l'attribut int testDeCalcul dans la fonction AlgoMinimax permettant de savoir si cette fonction doit appliquer les règles de la classe. Les fonctions RechercheNoeudPlusFort et VerifieNoeudPresent ne sont pas necécessaire.
L'importation de GestionnaireStatistiques n'est utilisé ni dans la classe Noeud, ni dans la classe Solution.
Dans la classe Main, Nous importons seulement InterfaceGraphique.FenetreMenu, étant la seule importation necéssaire.

## 5.2 Soucis d'implémentation

Nous avons eu des soucis dans l'implémentation de l'IA pour qu'il puisse jouer contre l'homme (si il doit commencer ou laisser la main en fonction du type de partie que l'on a sélectionné).
L'implémentation et l'association des règles de base de l'algorithme utilisés par l'I.A : La fonction Baseinverse fût un ennui majeur car nous nous sommes d'abord concentrés sur la côté défensif sans inclure la prise en compte des pions alliés déjà posés. Nous avons donc dû améliorer cette fonction pour que l'I.A ne fasse pas des erreurs abusives.
L'implémentation de la fonction partieTerminee, qui ne fonctionnait pas dans certains cas : elle regardait seulement si l'alignement des pions était égal à 4 et ne prenait pas le cas où l'on alignait plus de 4 pions.
De plus, il se déplaçait trop loin dans le tableau (ArrayIndexOutOfBoundsException).
L'implémentation de la fonction MouseClicked a été faite à l'intérieur du addMouseListener afin

d'éviter la génération automatique d'autres fonctions dont on ne se sert pas (MousePressed, MouseReleased, etc...).

### 5.3 Améliorations possibles

Pour améliorer le programme, nous aurions tout d'abord dû utiliser plus de règles de base. En effet dans la thèse de Victor Allis, celui-ci évoque neuf règles dont cinq indispensables à implémenter.
Les implémentations de toutes ces règles auraient pu permettre la victoire assurée, dès lors qu'elle commence la partie, et augmente la pertinence du choix de la machine.
Quand on crée un arbre dans la méthode AlgoMinimax de la classe Noeud représentant les suites possibles de Plateau, nous avons, par défaut, trouver un système pour ne pas appliquer les règles ni descendre plus profondémment dans les plateaux lorsque la partie est finie (c'est à dire une partie gagnée ou nul).
Par conséquent nous avons appliqué l'un des principes d'alphaBeta avant même d'avoir implémenter cette fonction. Mais cette fonction aurait pu être utilisée sur un plateau de manière à ne pas étudier deux fois les plateaux symétriques car les règles vont réagir de la même manière dans un sens comme dans l'autre.
De plus pour nous aider dans cette idée de symétrie, nous aurions pu implémenter RechercheNoeudPresent afin qu'il recherche les noeuds similaires dans un arbre de Noeud contenant les plateaux.

Pour ce qui est du jeu, nous aurions pu créer un bouton permettant de rejouer une partie afin de pouvoir simplifier nos tests mais cela aurait modifié nos spécifications.

## 6 Conclusion

### 6.1 Organisation interne au sein du groupe

La séparation du travail a d'abord été réalisée sur un module pour chaque membre du groupe.
Ces modules sont le Gestionnaire de sauvegarde et chargement, l'interface graphique, le gestionnaire de statistiques et le plateau.
Par la suite nous nous sommes répartis le travail à faire pour le module d'algorithme d'approximation représentant l'intelligence artificielle de notre projet.
La répartition des tâches s'est donc réalisée comme ceci :

**Module Gestionnaire de chargement :**
Anthony s'est occupé de programmer ce module.

**Module Statistiques :**
Yann s'est occupé de programmer ce module.

**Module Interface graphique :**
Kahnvud s'est occupé de programmer les classes FenetreJeu, FenetreMenu, dessinPlateau et FenetreStatistique de ce module. Kahnvud, Quentin et Yann se sont occupés ensemble de l'implémentation de l'intelligence artificielle dans la classe dessinPlateau.
Anthony s'est occupé de tout ce qui concerne la sauvegarde et le chargement de cette partie.

**Module Plateau :**
Dans la classe Plateau, Quentin s'est occupé des constructeurs et des accesseurs de la classe Plateau. Ainsi que de la méthode premierecasevide et de partieterminee.
Quentin s'est occupé de faire la partie concernant la classe plateau pour la méthode poserpion et Anthony s'est occupé de faire la partie concernant la sauvegarde pour la fonction poserpion.

**Module Alogrithme d'approximation :**
Anthony s'est occupé de la classe Solution pour ce module. Puis tout le monde à participé à l'implémentation de la classe noeud.
Kahnvud s'est occupé des méthodes Claimeven et Aftereven.
Yann s'est occupé des méthodes Lowinverse et Menacepairimpair.
Anthony s'est occupé des constructeurs et des accesseurs de la classe Noeud ainsi que des méthodes Ajoutelementliste, Supprimeelementliste, Determinemeilleurcoup, Algominimax, Alphabeta, Algoregledebase, Vertical et RechercheNoeudPlusFort.

## 6.2 Conclusion technique

Ce projet s'est révélé très enrichissant dans le sens où il a consisté en une approche concrète d'une intelligence artificielle. La prise d'initiative, le respect des délais et le travail en équipe ont été des aspects essentiels pour la réalisation de ce projet. Nous avons réussi à bien nous répartir les tâches afin de réaliser nos objectifs dans le temps et l'ambiance générale du groupe était très bonne.
Globalement, nous avons consolidé nos connaissances générales et approfondi notre efficacité pour le travail de groupe. Les principaux problèmes que nous avons rencontrés restent l'implémentation des règles de bases du puissance 4 dans le module de l'algorithme d'approximation.

Pour le nombre d'heures et lignes de code passées par parties :
Module Plateau : 4 heures et 362 lignes de code.
Module Statistiques : 1 heure et 77 lignes de code.
Module Gestionnaire de Sauvegarde et Chargement : 12 heures et 301 lignes de code.
Module Interface graphique : 16 heures et 508 lignes de code.
Module Algorithme d'approximation : 60 heures et 1071 lignes de code.

Dans le cahier des charges, nous avions prévu 898 lignes de code au total et nous avons 2325 lignes au final.
Pour le nombre d'heures nous avions estimé au total 76 heures et nous avons réalisé le programme en 93 heures.