Projet programmation 2 **ENS Cachan – 1AS2 2017**
31 mars 2017 Grosshans Nathan & Schwoon Stefan

Sujet du projet – Troisième partie

1 Travail demandé

Pour cette troisième partie, il vous est demandé de compléter votre simulateur graphique du jeu d'échecs en programmant d'un côté une véritable intelligence artificielle et d'un autre côté un module permettant d'interagir avec d'autres moteurs de jeu d'échecs grâce au protocole CECP.

1.1 Intelligence artificielle

Les intelligences artificielles programmées pour jouer au jeu d'échecs reposent classiquement sur deux procédures fortement liées : la procédure d'exploration de l'espace des mouvements possibles [1] et la procédure d'évaluation des positions atteignables par ces mouvements [2] ; le défi étant bien évidemment d'explorer le plus de mouvements possibles en le moins de temps possible et d'estimer avec le plus de justesse possible la valeur des positions atteintes pour, au final, effectuer un mouvement aussi judicieux que possible.

Il vous faudra au minimum programmer l'algorithme classique de recherche *Alpha-Beta* [3], et ce de manière suffisamment modulaire pour pouvoir facilement changer la fonction d'évaluation d'une partie à l'autre, voire au cours d'une même partie. Du côté de la fonction d'évaluation, il vous faudra au moins prendre en compte le matériel disponible [4] ainsi que la position de chaque pièce présente sur l'échiquier [5], sachant qu'il devra être possible d'activer ou désactiver de manière indépendante chaque critère d'évaluation. Vous pouvez, pour vous simplifier la tâche d'ajustement des valeurs, vous baser sur la fonction d'évaluation « simple » utilisant ces deux critères et décrite sur la page suivante : `https://chessprogramming.wikispaces.com/Simplified+evaluation+function`.

1.2 Communication

Le *Chess-Engine Communication Protocol* (CECP) [6] est un protocole permettant à des interfaces graphiques pour le jeu d'échecs de communiquer, de manière standardisée, avec des moteurs jouant à ce jeu. Un exemple d'un moteur supportant ce protocole est `gnuchess`, installé sur les machines de la salle 411. Il vous est demandé d'étendre votre programme de manière à ce qu'il soit possible de faire piloter un joueur par un moteur de jeu d'échecs utilisant le protocole CECP.

Il ne vous est pas demandé de supporter le protocole avec toutes ses fonctionnalités plus ou moins poussées, mais de telle sorte qu'il soit possible de jouer une partie dont l'un des joueurs sera piloté par `gnuchess`, avec lequel votre programme gérera la communication de façon telle qu'au minimum :

- le moteur puisse jouer soit les noirs, soit les blancs ;
- la communication des mouvements se fasse de manière automatique et transparente dans les deux sens ;
- les informations concernant le temps et toutes les conditions de fin de partie soient transmises au moteur et gérées en cas de retour de celui-ci ;
- l'échiquier puisse être initialisé à une certaine configuration (pour pouvoir potentiellement reprendre une partie à partir d'une sauvegarde), et une partie lancée à partir de cette configuration.

Il est recommandé d'utiliser la commande `gnuchess -x` (et de rajouter `-e` pour de simples tests afin de limiter la consommation utile de temps de calcul) et de communiquer avec `gnuchess` à travers un « pipe » (voir `http://blog.madhukaraphatak.com/pipe-in-spark`, section « Pipe implementation »).

1.3 Améliorations possibles

Une fois toutes les fonctionnalités décrites ci-dessus intégrées à votre programme, il vous est possible de les enrichir de nombreuses améliorations, par exemple en :

1. `https://chessprogramming.wikispaces.com/Search`
2. `https://chessprogramming.wikispaces.com/Evaluation`
3. `https://chessprogramming.wikispaces.com/Alpha-Beta`
4. `https://chessprogramming.wikispaces.com/Material`
5. `https://chessprogramming.wikispaces.com/Piece-Square+Tables`
6. `http://hgm.nubati.net/CECP.html`, `https://www.gnu.org/software/xboard/engine-intf.html`

— mettant en œuvre une ou plusieurs des nombreuses améliorations que l'on peut apporter à l'algorithme Alpha-Beta de base pour augmenter son efficacité (chacune activable et désactivable indépendamment dans l'interface graphique) ;
— mettant en œuvre la prise en compte d'autres critères dans la fonction d'évaluation (chacun activable et désactivable indépendamment dans l'interface grapique) ;
— étendant le support du protocole CECP ;
— ajoutant une fonctionnalité de jeu en réseau en permettant la connexion à un serveur de jeu utilisant le protocole CECP (votre programme agissant alors, au sens du protocole, en tant que moteur de jeu d'échecs).

2 Critères d'évaluation

Globalement, les mêmes critères que pour les deux premières parties s'appliquent pour cette partie.

2.1 Rapport et soutenance

Vous devrez rendre un rapport de 2 à 3 pages (en format PDF, généré par LaTeX) expliquant les choix d'implémentation que vous aurez rencontrés. Dans le cas où votre programme présenterait des défauts, il faudra les mentionner dans le rapport en précisant leurs raisons. Une soutenance d'une dizaine de minutes par groupe sera organisée à la fin de la troisième partie du projet durant laquelle vous nous ferez une démonstration de votre programme.

2.2 Fonctionnalités du code

Bien évidemment, votre projet sera évalué par ses fonctionnalités. S'il remplit tout ce qui est demandé, rajouter d'autres fonctionnalités pourra apporter un bonus. **Pour les groupes de 3 personnes**, le minimum de fonctionnalités attendu comprend aussi au moins une des améliorations possibles suggérées plus haut (sachant que l'amélioration devra bien évidemment être suffisamment étoffée). L'ergonomie de l'interface graphique pour ce qui concerne son intégration des nouvelles fonctionnalités sera également prise en compte dans l'évaluation de votre projet : il n'est pas nécessaire que celle-ci soit très évoluée ou ait une apparence particulièrement travaillée, mais elle devra être suffisament fonctionnelle et intuitive à l'utilisation.

2.3 Organisation du code

Votre projet devra impérativement être organisé hiérarchiquement, en le séparant en fichiers, classes et méthodes. Vous tâcherez de séparer du mieux possible les différentes fonctionnalités ajoutées au cours de cette partie en classes. Gardez sous le coude la règle de ne jamais avoir de fonction trop longue ou de fichier trop grand.

2.4 Qualité du code

L'utilisation adéquate de la programmation orientée objet et de Scala sera un critère important dans l'évaluation, notamment pour compléter et étendre votre programme en factorisant et en réutilisant au mieux votre code. Dupliquer du code dans des classes sous-entendrait une mauvaise compréhension de l'héritage. De même, préférez des directives fonctionnelles concises à des boucles `for` et `if` imbriquées, voir la Section 4.4 de l'introduction à Scala. La mise en forme, la présence de commentaires et la cohérence des noms de classes, méthodes et variables devront être suffisamment décentes pour une lecture agréable du code. Nous vous demandons également de documenter votre code, comme expliqué dans la Section 7 de l'introduction à Scala.

3 Dates importantes

— Le code et le rapport seront à rendre avant le mardi 16 mai à 23h59.
— La soutenance pour la troisième partie aura lieu le vendredi 19 mai.