Comparaison groupes noCaml et DHK

Correction:

Les deux implémentations répondent au sujet initial (faire jouer les fourmis en communicant avec le serveur selon le protocole établit).
Cependant, toutes les contraintes n'ont pas été respectées : notre groupe a respecté sa contrainte d'éviter les effets de bords, et n'a pas respecté celle d'implémenter une partie du programme en C.
Le groupe DHK a proposé une solution à sa contrainte de rejouabilité d'une partie sans l'implémenter, et n'a pas réussi à implémenter celle de jouabilité sur navigateur.

Efficacité :

L'IA implémentée par le groupe DHK est plus efficace que celle que nous avons implémentée (puisque la notre se contente d'avancer).

Cependant, en terme d'évolutivité, notre architecture permet une bien meilleur IA que celle proposée par DHK. En effet, leur architecture impose une gestion individuelle des fourmis et des tours de jeu, alors que la notre permet une gestion globale des fourmis, et construit la carte du jeu qui permet de créer des stratégies globales sur plusieurs tours (recherche de chemin par exemple).
Leur façon de traiter la contrainte de rejouabilité d'une partie est également gênante, puisqu'elle bloque leur IA sur un algorithme déterministe.

De par notre utilisation d'une Map, notre consommation mémoire est légèrement supérieur à celle de DHK.

Généralité:

Dans les deux cas, les modules sont clairement identifiables. Je pense que les 2 codes sont difficilement réutilisables en dehors du projet Antroid, à l'exception de notre worldmap/pathfinding.

Clarté:

Le projet de DHK est très bien commenté dans son ensemble. On pourra cependant regretter l'absence de commentaires pour certains passages compliqués, ce qui transforme certaines fonctions en boites noires.
La présence dans le README des dépendances du projet est un très bon point et facilité grandement l'installation du projet.

L'utilisation de packages nous permet d'être plus clairs au niveau de l'organisation de notre projet.

L'interface graphique du projet DHK est inutilisable, de par l'absence de précision sur les champs à renseigner : pour jouer avec cette interface, il faut connaître le protocole de communication avec le serveur.

L'utilisation d'une couche supplémentaire pour le code des zombies est un véritable atout pour la compréhension de cette partie du code.

Robustesse:

Que se passe-t-il (pour les 2 projets) si le temps d'exécution entre deux tours est trop court ?
pas de contrôle du temps d'exécution (que se passe-t-il si le serveur est trop rapide ?)
De par l'utilisation de langages fortement typés, les 2 projets sont globalement robustes.