Doumbia Amadou
Diallo Seydou
Issa Saidou Mourtala
Tonnelier Jérôme

Analyse du code Acide Formique

Analyse du rapport :
Le rapport explique bien l'architecture utilisée par un diagramme UML ce qui facilite la vision globale du projet.

Les acteurs/agent auraient pu être utilisés pour la prise de décision d'IA au lieu d'une simple délégation. Car c'est par forcement le principe de ce modèle.

Analyse de l'implémentation du projet:
Le choix du langage : scala ;
L'utilisation de scala est à priori un bon choix par rapport à l'utilisation des acteurs. Le projet avait des parties intéressantes utilisant des mécanismes scala, Exemple la composition mix-in pour la définition de l'IA et des stratégies de jeu.

On aurait pu y voir l'utilisation d'un système distribué pour la prise de décision de la commande à envoyer pour une fourmi par exemple. Et par la suite utiliser des algorithmes d'apprentissages pour améliorer les choix et profiter pleinement des acteurs/agents.

La distributivité de la classe (Actor) a permis de produire plusieurs processus en cours et donc, offre plus de possibilités pour la jouabilité.

Le choix du langage permet une évolution facile et l'architecture comporte plusieurs parties améliorables (Ex : AntCommand). Il existe une abstraction intéressante au niveau de la définition du code des fourmis zombies qui pourrait être améliorée si besoin.

Respect des contraintes :
Le langage C a été bien utilisé car ils ont su distinguer les composants indépendants les uns des autres. Ce qui leurs a permis de satisfaire leurs contrainte sans avoir besoin de faire collaborer les deux langages.

L'utilisation des agents/acteurs a été également respectée.

L'utilisation des traits pour définir les types de fourmi permet de modifier facilement les comportements des fourmis.

Lisibilité/Clarté du code :
Le code est clair, espacé et bien indenté. De par les noms de classes et variables, il est assez aisé de comprendre la totalité du code et de son objectif.

Evolution/Réutilisabilité :
Il y a également lieu à poursuivre l'évolution comportementale des fourmis grâce aux traits, ou ajouter de nouveaux acteurs dans le jeu. Il y a donc une bonne possibilité d'évolution à ce niveau-là.

Clarté et exhaustivité du rapport :
Le Rapport est clair. Il est même accompagné d'une représentation de l'architecture, donc il permet de reprendre ce projet sans trop de difficultés.

Commentaires :
Les commentaires sont présents dans les classes quand il est nécessaire, autrement, la clarté du code prend le relais. Donc bonne répartition des commentaires présents dans les classes.

Facilité de la compilation/exécution du code :
(Trop d'outils à utiliser pour faire fonctionner le code)

-Efficacité (Temps et mémoire) :
Comme on n'a pas pu intégrer le module complémentaire AKKA, on n'a pas pu tester à temps l'efficacité des agents/acteurs. Une documentation aurait pu être bénéfique.

Tests de fourmis très peu fournis mais il est possible de paramétrer facilement ce détail.

Comparaison

Structure de données :

On peut noter une différence d'approche, au niveau de la représentation des informations retournées par le serveur.
La totalité des formats de messages retournés par le serveur est codé «en dur » dans une classe dédié dans leur programme, l'utilisation d'une couche d'abstraction aurait permis une extensibilité plus simple si l'API (et donc le format de message) venait à changer.

La manipulation d'objet non typé (défini explicitement) de notre architecture nous permet de contourner ce type d'évolution car il ne nous est pas nécessaire de connaître la structure exacte de l'objet, un changement de l'API n'implique pas la redéfinition d'un objet.

- Modularité :
Leur programme est modulaire car structuré en composant, chaque composant a son rôle et fonctionne indépendamment du reste de l'application.
Notre programme est également séparée en parties ayant un rôle distinct, mais l'architecture est moins flexible et nécessiterait des modifications plus approfondis pour réaliser d'éventuelles modifications sur certains points (par exemple au niveau de l'IA qui n'a pas sa propre représentation hors de la boucle principale du jeu).

Auteurs :
- Ce document est réalisé par le groupe DDT du cours PCOM15 de l'université Paris Diderot.