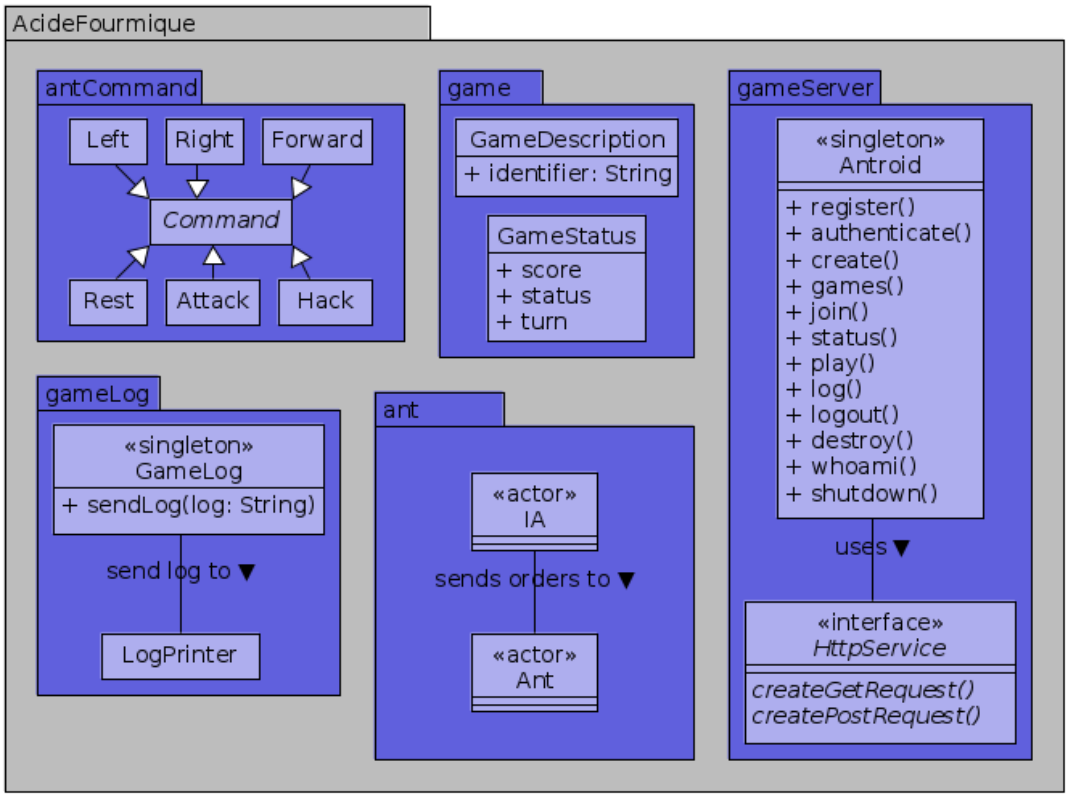
Progammation Comparée – Antroid
Acide Fourmique
Rapport

Equipe : DO Syou-On, ILLOUS Hugo, LAM Victor

Contraintes :
- Ecrire une partie du code en C
- Utiliser les acteurs/agents

Technologies utilisées :
- Scala/C
- Akka (programmation par acteurs)
- Lift-json (parsing json)

Architecture :



gameServer : communique avec le serveur de jeu Antroid par envoi de commande (register, auth, create, play…). Cette communication avec le serveur est effectuée avec l'outil « Curl ». Cependant, pour réduire le couplage entre cet outil et notre code, nous avons ajouté une façade HttpService, que

l'objet CurlService implémente.

antCommand : représente l'ensemble des commandes possibles des fourmis, ainsi que le langage utilisé par les fourmis zombies. Grâce aux possibilités qu'offre le langage Scala, nous avons pu créer nos propres opérateurs, facilitant ainsi la création (et la lisibilité) du code d'un zombie.

game : représente l'état d'une partie et sa description.

gameLog : récupère le log d'une partie et l'affiche sur la sortie standard. Le LogPrinter est un serveur écrit en C qui est lancé dès l'exécution du programme principal, en tâche de fond. Il se met en attente de la réception d'un log d'une partie et, une fois reçu, l'affiche puis s'arrête.

ant : à l'heure actuelle, ne contient que les classes IA, Ant et Main, qui forment le squelette de la gestion des fourmis. La classe IA est un acteur qui va créer 2 fourmis et va demander à la première d'attaquer et à la deuxième d'avancer, et envoie un message indiquant que l'action a été effectué. La classe Ant en fonction du message envoyé affichera l'action demandé par l'IA, avant d'envoyer un message lui indiquant que l'action a été faite.

Le contenu du package ant avait pour but de gérer le comportement des fourmis à l'aide d'un acteur IA qui va créer le nombre de fourmi nécessaire au bon fonctionnement de la partie et qui aura pour but de créer chaque fourmi de la partie avec un rôle déterminé représenté par des traits. Chacun de ses rôles se verra associer un ou plusieurs comportement qui sont eux même des traits. En fonction de ces comportements la fourmi effectuera des actions propre à ce comportement. Les comportements définie seront Coward, Aggressive, Helper, Camper, Follower. Les rôles et leur comportement associé seront Soldier (Agressive), Explorator (Coward), Sentinelle (Camper + Explorator), Supporter (Follower + Helper). L'acteur IA contrôle le statut des fourmis, à savoir récupérer les informations concernant le niveau de la santé, de l'acide, ainsi que son entourage, et enverra un message à la fourmi qui effectuera l'action recommandé par son rôle. L'IA aura également pour rôle de stocker dans une matrice ou autre structure le contenu de la carte. En effet, à chaque tour, l'IA devra mettre à jour la carte, qui sera mis à jour à chaque tour dû au fait que les explorateur vont explorer la carte.

Nous avons envisagé que les soldats pouvaient changer la quantité d'acide utilisé en fonction de sa vie, et que si jamais il tombait à court d'acide, il fuirait le combat vers la source de nourriture la plus proche découverte pas les explorateurs.

Problèmes rencontrés :

- Le parsing des réponses envoyées lors d'une requête « play » et « log » n'a pas été effectué. Le serveur de log affichera donc le log au format Json.
- Nous pensions intégrer un parseur Json en C pour parser le log reçu par le serveur de log mais, par manque de temps, ceci n'a pas pu être fait.
- La récupération du status d'une partie ne fonctionne que pour les parties publiques, ceci étant dû au format du champs « visibility » renvoyé par la requête status (le champs est soit une string, soit une liste de string).
- Le serveur de jeu Antroid ne reconnaît pas la primitive « see_ant », ou les instructions « Attack » et « Store » (à cause des entiers pour les instructions) d'un programme zombie. La commande Attack fonctionne pourtant bien en tant que commande de fourmi.
- La gestion des fourmis par les acteurs n'a pas été intégrée au projet et seul un squelette a été implémenté avec la gestion des messages sans les actions avec le serveur.