Programmation Comparée

Josian Chevalier, Vladislav Fitc, Thi Ngoc Tam Nguyen

26 mars 2015

1 Contraintes

Utiliser un minimum de bibliothèques
On doit pouvoir rejouer l'exécution de programme

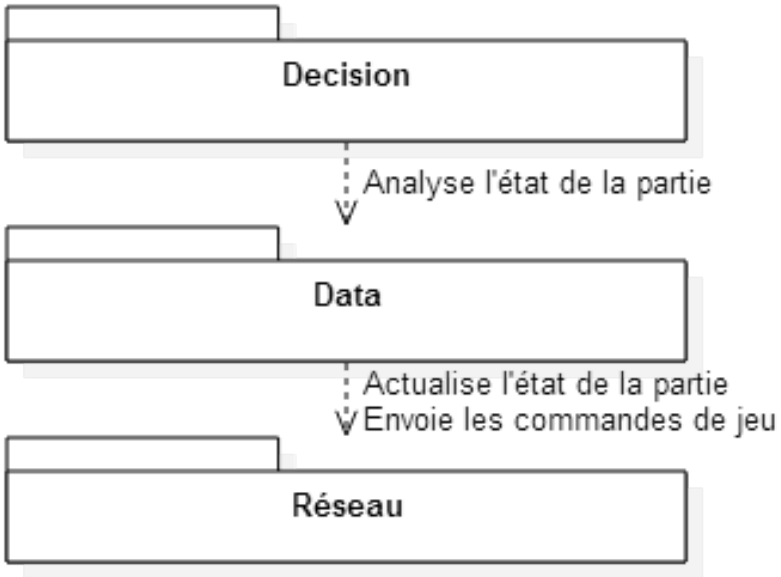
2 Architecture

Notre système se découpe en trois couches principales :

a. **La couche réseau :** Elle s'occupe des communications avec le réseau.

b. **La couche data :** Contient les informations sur l'état du jeu, est mise à jour par la couche réseau.

c. **La couche décision : (ou IA)** Contient les fonctions nécessaires à l'analyse du jeu et la prise de décision.



La couche réseau envoie les nouvelles observations à la couche de données. Après avoir actualisé les données, la couche Data appelle la couche de décision afin de jouer un tour.
La couche de décision choisis les déplacements à effectuer et demande à la couche de données de les effectuer. La couche de données met en cache les prochains déplacements à effectuer, et la couche réseau les appliquera pour le changement de tour
On a également un module joueur, qui représente le joueur qui recherche de nouvelles parties, ou en créé une.

3 Réseau

La couche réseau effectue toutes les communications avec le serveur. Elle contient simplement la bibliothèque des fonctions permettant d'envoyer des commandes au serveur, ou de récupérer les observations des fourmis.

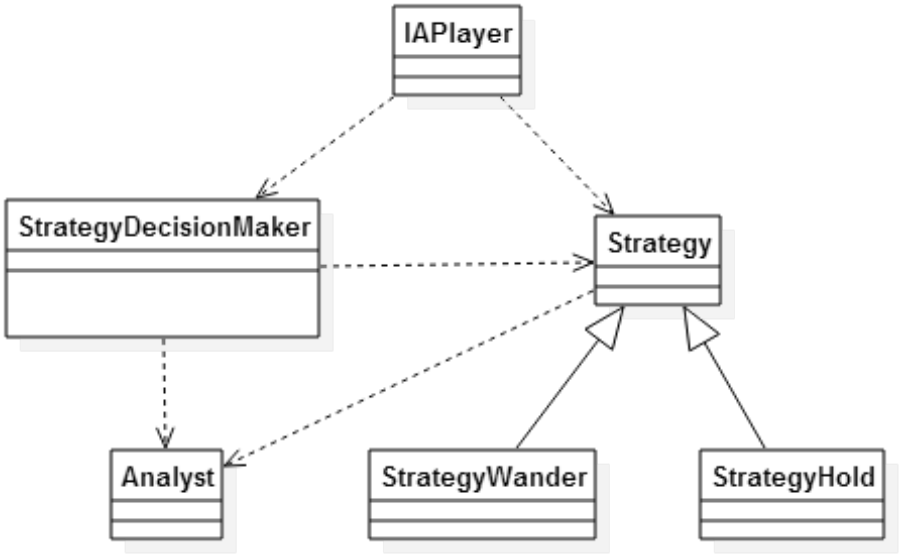
4 Data

La couche Data contient une représentation du monde qu'elle actualise en interrogeant le serveur. Elle contient également les fonctions pour faire évoluer les fourmis dans ce monde, fonctions qui appellent celles de la couche réseau. Elle permet d'isoler totalement la couche d'analyse de la couche réseau, permettant une interaction plus simple entre les deux.

La carte est composée de cases qui peuvent contenir des fourmis.
Les cases contiennent toutes les informations concernant les obstacles ou ressources éventuelles, et contiennent également des fourmis.
Les fourmis contiennent toutes les informations liés à leur état et les commandes nécessaires à les manipuler.

5 Décision

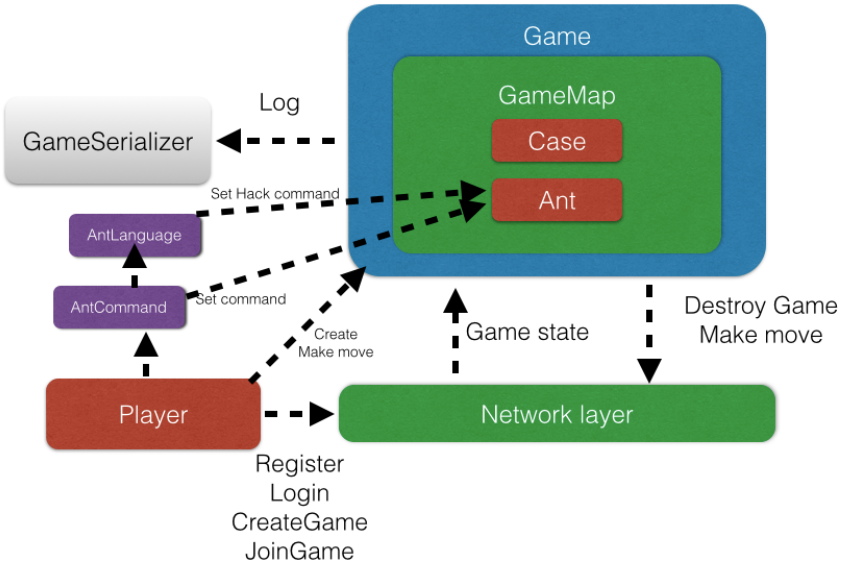
La couche de Décision interroge Data sur l'état du jeu, et analyse les données qu'elle en tire afin de définir la meilleure stratégie à adopter.



La classe IAPlayer va jouer un tour. Pour cela, elle va faire appel à la fonction pour jouer un tour de son objet Strategy.
La classe Strategy contient les fonctions nécessaires à déplacer les fourmis en accord avec une stratégie précise. La classe Strategy est une classe abstraite, et ses classes filles implémentent des stratégies différentes. Par exemple, la classe StrategyWander déplace les fourmis afin de découvrir la carte, et la classe StrategyHold va les commander de manière à défendre des ressources.
Le stratégie Wander n'a hélas pas été implémenté totalement.
La classe StrategyDecisionMaker choisit la stratégie la plus adaptée à l'état de la partie, et va passer à IAPlayer un objet de la sous-classe de Strategy adapté.
La classe Analyst est un helper qui contient toutes les fonctions utiles à l'analyse d'un monde.
Les commandes de déplacement de fourmis sont envoyer à la couche Data

6 Interactions entre les couches

Les interactions entre les modules de données et de réseau sont détaillées ci-dessous :



On peut constater que le module Player interagit directement avec la couche réseau pour créer ou rejoindre des parties, ainsi que pour jouer 'à la main'. C'est dans le module GameSerialiser que l'on enregistre des logs qui permettront de rejouer l'exécution, en stockant toutes les commandes qui ont été jouées. Enfin, on manipule des objets AntCommand afin de communiquer avec les fourmis.