Licence 3 de Sciences, mention Informatique

Systèmes Distribués

Projet : Génération de jeu et tests de stratégies de joueurs

Caty Gomes Fernandes
caty.gomes-fernades@etu-unistra.fr

1 Vocabulaire

Dans la suite, il faudra distinguer entre un jeu (*"a game"*), une partie (*"a round"*) et des tours (*"a turn"*). Un jeu peut avoir plusieurs parties, et dans une partie les joueurs jouent à tour de rôle jusqu'à la fin de la partie.

Cette distinction est importante car il y aura deux agents particuliers, à savoir le coordinateur de jeu et le coordinateur de partie.

2 Choix

2.1 Démarrage d'une partie

Chaque agent est en même temps un objet distant et un programme indépendant, donc il sera trouvable par un couple d'adresse machine et de port du serveur de nom RMI.

Lorsqu'on lance une partie, le coordinateur de jeu et les joueurs doivent connaître tous les adversaires et tous les producteurs. Il faut donc fournir les informations nécessaires pour permettre l'accès à ces objets distants. Ceci se fait en deux étapes. Dans un premier temps, tous les producteurs et joueurs transmettent leur présence au coordinateur de jeu, qui gardera une liste de ces agents et comment les trouver.

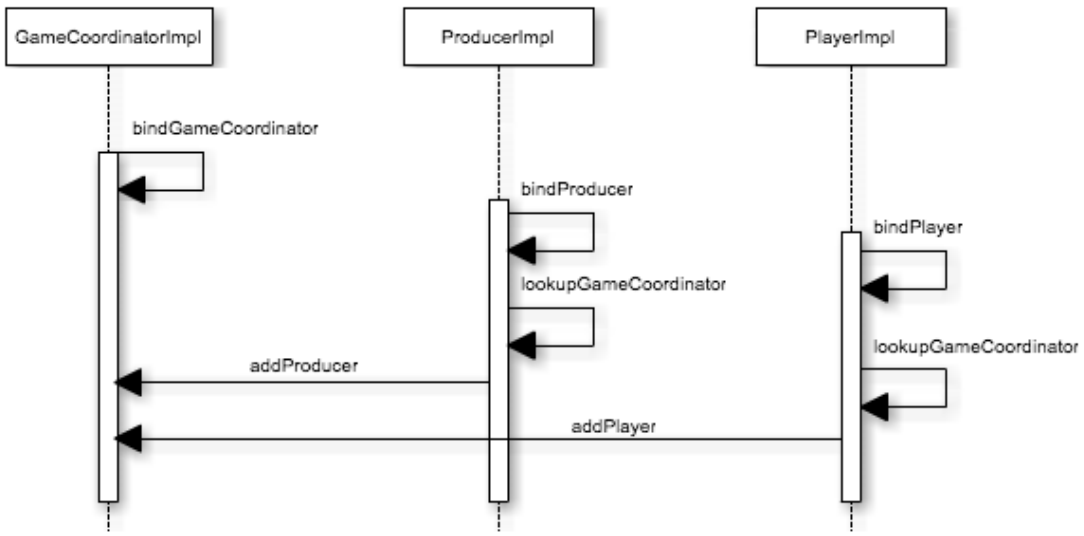

FIGURE 1 – Les producteurs et joueurs signalent leur présence au coordinateur de jeu.

Ensuite, en supposant que tous les agents sont déjà prêts à jouer lorsqu'on lance une partie, le coordinateur de partie ne fait que demander la liste des agents au coordinateur de jeu, et il transmet ces informations aux joueurs.

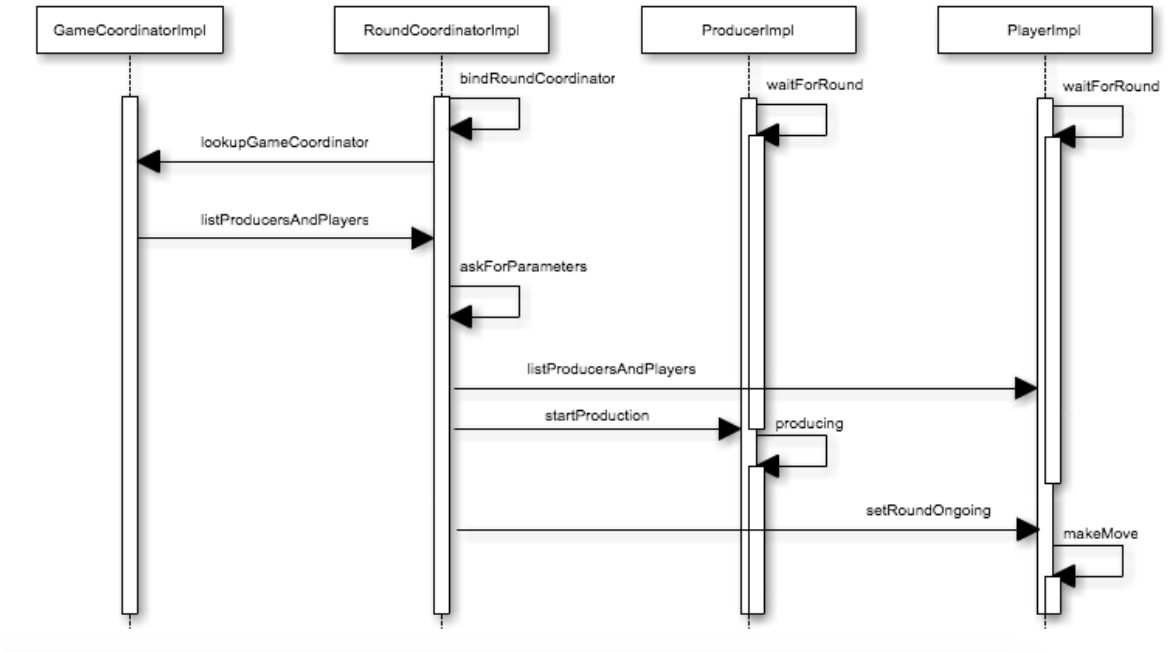

FIGURE 2 – Le coordinateur de partie fait des préparations pour pouvoir lancer la partie.

Remarque par rapport à la figure 2 : Ici il n'y a qu'un joueur pour alléger le diagramme. Il faut au moins deux joueurs pour lancer une partie.

2.2 Producteurs

Un producteur produit un seul type de ressource. Il démarre avec 50 copies de cette ressource, puis attendra qu'un coordinateur de partie lui donne la permission de commencer sa production. Ceci empêche le producteur d'avoir trop de ressources à disposition lors du démarrage de la partie, et en même temps on a une limite inférieure pour le choix d'objectif de la première partie.

Il produira 5 ressources supplémentaires toutes les secondes. Le nombre maximal de copies qu'un joueur peut prendre en une fois est demandé à l'utilisateur à chaque partie. Le producteur ne rend jamais plus que ce nombre de copies, même si un joueur en demande plus.

A la fin d'une partie, la production est stoppée et le nombre de copies est de nouveau fixé à 50. Ceci permet de commencer une nouvelle partie dans les mêmes conditions que la première.

2.3 Joueurs

Un joueur non humain a par défaut une personnalité coopérative, mais une option permet de choisir qu'il soit individualiste. Un joueur coopératif va alterner de ressource de manière cyclique, tandis qu'un joueur individualiste tente d'atteindre l'objectif d'une ressource avant de passer à une autre.

Un joueur humain est demandé à chaque tour quelle ressource il souhaite choisir, et depuis quel producteur il souhaite prendre la ressource choisie.

Quand le joueur atteint ses objectifs, il notifie le coordinateur de partie.

2.4 Coordinateur de Partie

Le coordinateur de partie est chargé de demander les paramètres à l'utilisateur et de transmettre toutes les informations nécessaires aux joueurs pour pouvoir commencer la partie. Si aucun producteur n'est présent ou s'il n'y a pas au moins deux joueurs, il se termine sans lancer

de partie. Il doit aussi signaler le début de la partie aux producteurs pour qu'ils puissent démarrer leur production.

Si les joueurs jouent à tour de rôle, le coordinateur choisit au hasard le premier joueur à commencer, puis notifie chaque joueur un par un que c'est leur tour. Il se met en attente pendant qu'un joueur fait son choix.

Sans tours, il signale simplement à chaque joueur que la partie a commencée.

3 Exécution

Il faut d'abord exécuter le coordinateur de jeu, en donnant comme paramètre le port du serveur de nom RMI précédemment lancé.

```
java GameCoordinatorImpl <port>
```

Par la suite on peut lancer les producteurs et les joueurs dans un ordre quelconque. C'est à ce moment qu'on choisit le type de ressource à produire, et si le joueur est humain ou pas. Par défaut, un joueur a un caractère coopératif, mais l'argument `"i"` permet de le rendre individualiste.

Comme tous les producteurs (resp. tous les joueurs) enregistrent leur objet distant sous le même nom, il faut au moins autant de ports différents qu'il y a de producteurs (resp. joueurs).

```
java ProducerImpl <resource type> <port> <GameCoord Host <GameCoord Port>
java PlayerImpl <port> <GameCoord Host> <GameCoord Port> [h] [i]
```

Le coordinateur de partie est exécuté en dernier.

```
java RoundCoordinatorImpl <port> <GameCoord Host> <GameCoord Port>
```

Avant de lancer la partie, Le coordinateur demande encore les paramètres suivants à l'utilisateur :

— Pour chaque type de ressource, quel est l'objectif à atteindre ?
— Quel est le nombre maximal de ressources qu'on peut prendre à la fois d'un producteur ?
— S'il n'y a pas de joueur humain, faut-il jouer tour par tour ?
— Faut-il permettre l'observation des autres agents ?
— Faut-il attendre que chaque joueur atteigne son objectif ?

4 Tests

Les scripts suivants sont fournis pour lancer des programmes de test :

— `runGame` lance le coordinateur de jeu.
— `runP1`, `runP2` et `runP3` lancent des joueurs, où P2 est un joueur humain et P3 a une personnalité individualiste.
— `runProd1`, `runProd2`, `runProd3` lancent des producteurs, mais Prod1 et Prod3 produisent le même type de ressource.
— `runRound` lance une nouvelle partie.

5 Améliorations

Il y a certains points qui n'ont pas été implémentés, notamment le vol entre joueurs, ou l'évaluation des parties. Les producteurs ne sont pas paramétrables, donc toutes les ressources sont épuisables.

D'autres améliorations seraient de pouvoir enlever un joueur ou producteur du jeu, et de faire une meilleure gestion des erreurs.

A Diagramme de classes