Jeux de plateau en réseau

Projet Applications réseau M1 Miage Apprentissage Paris Dauphine 2013/2014

F. Sikora (D'après un sujet de M. Chilowicz)

1 Dates importantes

— Envoi des binômes par email : 2 mai 2014.
— Rendu de la RFC : 11 mai 2014.
— Rendu du projet : 29 juin 2014, 23h.
— Soutenances : 30 juin 2014, à partir de 14h.

2 Versions

Ce sujet est susceptible d'être modifié.
— 19 mai : dates de rendu et petites corrections.
— 29 avril 2014 : Mise en ligne du sujet.

3 En quelques mots

On vous demande de réfléchir à la création d'un protocole permettant de jouer à des jeux de plateau en réseau entre différents joueurs (ou IA), avec l'aide d'un arbitre. Les joueurs pourront se connecter à un arbitre proposant une partie d'un certain jeu puis jouer en tour par tour. Ils pourront également discuter entre eux.

On vous demande dans un premier temps d'écrire une RFC (Request For Comments) décrivant le protocole, puis une implémentation en Java du protocole.

4 Description des protocoles

Les jeux de plateaux peuvent être définis selon certains points : nombre de joueurs, taille du plateau, ordre des coups, mise à jour du plateau...

On demande l'écriture d'un protocole réseau gérant les communications entre clients (joueurs) et serveur (arbitre). Il doit définir comment les joueurs se connectent à un arbitre/jeu, comment ils échangent des coups (légaux ou illégaux) jusqu'à la déclaration d'un vainqueur, comment les erreurs sont gérées. Ce protocole doit être indépendant d'un jeu en particulier et être générique à tout jeu de plateau respectant certaines conditions (les messages ne spécifient rien concernant le jeu). Un arbitre peut bien sûr gérer plusieurs joueurs par partie, mais également plusieurs parties simultanées. Le protocole est libre, mais doit utiliser TCP.

Une fois connecté à un jeu, les joueurs peuvent vouloir discuter au cours de la partie. On demande également de décrire la manière dont ces communications sont effectuées. Celles-ci doivent se faire en UDP directement de joueur à joueur (le serveur ne relaie pas les messages). Ils doivent cependant passer par le serveur pour connaitre les coordonnées des autres joueurs.

5 Implémentation

On demande l'implémentation d'arbitres respectant la RFC proposée.

Dans un premier temps, implémentez en Java un arbitre pour le jeu Pierre-Feuille-Ciseaux-Lézard-Spock (https://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock). Le nombre de joueurs est fixé à 2, le plateau consiste uniquement en 2 entiers comptant le nombre de manches gagnantes pour chaque protagoniste (on joue en X manches gagnantes, défini par l'arbitre). L'arbitre doit demander les coups d'une manche à chaque joueur en même temps, mais signaler le coup de l'autre une fois tous les coups de la manche joués.

On demande également l'implémentation en Java d'un client texte pour ce même jeu (optionnellement avec une interface graphique).

On demande également l'implémentation d'un autre arbitre pour un autre jeu de plateau, par exemple : puissance 4, dames, échecs, 2048 (Fibonnacci) multijoueur (1 plateau par joueur et un temps limité, par ex : http://emils.github.io/2048-multiplayer/), EinStein würfelt nicht (https://en.wikipedia.org/wiki/EinStein_w%C3%BCrfelt_nicht!), Perudo (https://fr.wikipedia.org/wiki/Perudo), Ruzzle. (Me contacter pour d'autres jeux).

Ce second arbitre peut-être ajouté au premier (par exemple, le serveur propose 2 jeux au joueur) (plutôt que de dupliquer tout le code).

Optionnellement, on demande l'implémentation d'un arbitre générique (sorte d'API) pour un arbitre de jeu de plateau, indépendamment du jeu, pour faciliter l'ajout d'arbitres pour de nouveaux jeux. Ce serveur générique pourra être lancé avec en argument le nom d'une classe implémentant un arbitre pour un jeu en particulier. Il s'agira d'une sorte de bibliothèque, avec des classes génériques gérant un plateau, des coups, des joueurs et les communications réseaux, et proposant une série d'interfaces à implémenter pour l'ajout d'un jeu.

6 Conditions de rendu

Le projet est à effectuer en binôme, *i.e.* par 2 personnes. En cas de nombre impair d'élèves, un seul groupe sera autorisé à effectuer le projet en trinôme (noté plus sévèrement). Pour éviter les mauvaises surprises, envoyez votre binôme par email (florian.sikora@dauphine.fr) avant le 2 mai 2014.

Première étape : description du protocole. Une première RFC de votre protocole est à rendre avant le 11 mai 2014, 23h00, sur l'espace MyCourse dédié. Les noms des deux personnes du binôme devront apparaître dans la RFC, dans le nom du fichier ainsi que dans les commentaires de l'espace MyCourse. Attention, l'espace est fermé automatiquement. La RFC doit être dans le même format que les autres RFC, c ?est-à-dire un document texte ASCII rédigé en anglais (voir la RFC 2223 pour le format d'une RFC..., voir aussi ce lien http://www.rfc-editor.org/formatting.html). La RFC doit seulement décrire les méthodes de communications (format des messages, ordres des messages, protocoles utilisés...) entre les différentes entités intervenant dans votre protocole. Aucun détail d'implémentation ne

doit être donné. La RFC 1350 (TFTP) peut-être prise en exemple comme une RFC simple décrivant ceci.

Seconde étape : implémentation de joueurs et arbitre(s) pour le protocole en langage Java. Théoriquement, deux implémentations de différents binômes pourraient communiquer entre elles si elles suivent la même RFC (théoriquement toujours, une implémentation en langage C++ lancée sur une machine personnelle pourrait communiquer avec une implémentation Java sur un téléphone androïd). Votre projet est à rendre avant le 29 juin 2014, 23h00 sur l'espace MyCourse dédié. Une fois votre fichier envoyé, vous devez avoir un écran de confirmation avec votre fichier et la date de l'envoi. **Il y aura un point en moins par heure de retard, une heure entamée est due.** Le format de rendu est une archive au format **ZIP** contenant :

- Un répertoire *src* avec les sources de votre implémentation java.
- Un répertoire *classes* vide dans l'archive, devant contenir les classes du projet une fois compilé.
- Un répertoire *docs* contenant :
  - La première version de la RFC, au format précisé plus haut.
  - Une version éventuellement mise à jour de la RFC, au même format que la première.
  - Une documentation pour l'utilisateur *user.pdf* décrivant à un utilisateur comment se servir de votre projet.
  - Une documentation pour le développeur *dev.pdf*, devant justifier les choix effectués, indiquer les différences entre les 2 RFC, les avantages et inconvénients du protocole, présenter l'architecture choisie, indiquer quelles ont été les difficultés rencontrées au cours du projet ainsi que la répartition du travail entre les membres du binôme.
  - Un répertoire vide *doc* pour la javadoc.
- Des jars exécutables de joueurs et d'arbitre(s) pouvant être lancés au moyen de la commande `java -jar`.
- Optionnellement un fichier ant *build.xml* permettant de compiler, créer les jar, générer la javadoc, etc.

L'archive aura pour nom Nom1Nom2.zip, où Nom1 et Nom2 sont les noms des membres du binôme par ordre alphabétique. L'extraction de l'archive devra créer un dossier Nom1Nom2 contenant les élements précisés ci-dessus.

Il va sans dire que les différents points suivants doivent être pris en compte :

- Uniformité de la langue utilisée dans le code (anglais conseillé).
- Uniformité des conventions de nommage et de code (convention Java obligatoire).
- Projet compilant sans erreur et fonctionnant sur les machines de l'université.
- Gestion propre des différentes exceptions.
- La documentation ne doit pas être un copié-collé du code source du projet.
- Les sources doivent être commentées, dans une unique langue, de manière pertinente (pas de commentaire "fait un test" avant un if.).
- Le nom des variables, classes et méthodes doivent être choisis judicieusement.
- Le code devra être propre, les exceptions correctement gérées, les classes correctement organisées en packages. La visibilité des méthodes et champs doit être pertinente (privée ou non...).
- La RFC et le projet doivent évidement être personnels à chaque binôme. Un détecteur automatique de plagiat sera utilisé.
- Un projet qui fonctionne en réseau vu l'intitulé de la matière...

La documentation (rapports, commentaires...) compte pour un quart de la note finale. On

préférera un projet qui fonctionne bien avec peu de fonctionnalités qu'un projet bancal avec plus de fonctionnalités (login/highscore, GUI...). L'utilisation d'un gestionnaire de version type CVS ou SVN est fortement conseillée.

6.1 Soutenance

Une soutenance d'un quart d'heure aura lieu binôme par binôme, à partir de 14h00 le 30 juin selon le planning donné ci-après (avec le binôme associé évidement). Elle doit être préparée et menée par le binôme.