Devoir de Programmation PC2R 2016 - Des Robots qui Ricochent

Description du devoir

But du devoir

Le but du devoir est de réaliser une application clients-serveur permettant à des utilisateurs de jouer à une adaptation de *Rasende Roboter*[1]: un jeu ou les participants, à chaque tour, doivent trouver la meilleure solution à une énigme de déplacement de pièces au sein d'un plateau de jeu (la description du jeu en question est donnée plus bas).

Exigences techniques

L'application doit être réalisée suivant une architecture client / serveur.
La partie client et la partie serveur doivent être chacune écrite dans un langage différent contenu dans l'ensemble {`C`, `Java`, `OCaml`}.
L'utilisation d'autres langages est possible mais soumise à autorisation de la part de l'équipe pédagogique.
Le client et le serveur doivent pouvoir fonctionner indépendamment l'un de l'autre (et pouvoir fonctionner avec le client et le serveur de l'équipe pédagogique). Ils doivent respecter le protocole ci-dessous.

Règles du jeu

Déroulement d'une session de jeu

Au moins 2 joueurs participent à une session de jeu. Il n'y a, a priori, pas de nombre maximum de joueurs.
Une session de jeu comprends plusieurs tours de jeu. Le score d'un joueur persiste au cours d'une session. La session s'arrête quand le score d'un joueur atteint un score objectif, ou quand le nombre de joueurs descend à 1 ou moins.
Un joueur peut rejoindre ou quitter une session à tout moment. Quand un joueur rejoint le jeu au milieu d'un tour, il commence à participer au tour suivant.
Les informations commmuniquées aux joueurs pour la session sont: le plateau de jeu et la liste des autres joueurs.
Un tour de jeu se déroule en plusieurs phases:

- 1 phase de réflexion pendant laquelle les joueurs découvrent l'énigme et réfléchissent à une solution. Cette phase dure jusqu'à expiration d'un compte-à-rebours long (par exemple, 5 minutes) ou jusqu'à ce qu'un joueur annonce qu'il a trouvé une solution, en précisant le nombre de coups de sa solution.
- 1 phase d'enchères pendant laquelle les joueurs peuvent annoncer qu'ils ont trouvé une solution, à chaque fois en annonçant le nombre de coups de leur solution. Deux enchères du même nombre de coups de deux joueurs différents ne peuvent pas être validées toutes les deux. Un joueur ne peut pas enchérir une valeur plus grande que (ou égale à) une valeur qu'il a deja enchérie. La phase d'enchères s'arrête à l'expiration d'un compte-à-rebours court (par exemple, 30s) et décide du joueur actif (le moins-offrant).
- 1 phase de résolution, pendant laquelle le joueur actif propose sa solution. Il a le temps d'un compte-à-rebours court (par exemple 1 min) pour envoyer sa solution (la suite des coups). Si la solution est bonne (elle résout bien l'énigme) et si elle utilise moins (au sens large) de coups que ce qu'il avait annoncé alors le joueur actif gagne un point et on passe au tour suivant. Si la solution est mauvaise (ou utilise trop de coups), alors le joueur actif est exclu du tour; s'il existe encore d'autres joueurs dans le tour, le joueur suivant dans la phase d'enchères (le moins-offrant suivant) est sélectionné comme joueur actif et une nouvelle phase de résolution a lieu. Sinon on termine le tour, et personne ne gagne de point.

[1] `fr.wikipedia.org/wiki/Ricochet_Robots`

Enigme

Le plateau de jeu est une grille de 16x16 cases. Il y a, pendant le jeu, sur le plateau: des murs (qui bordent des cases), des robots de quatre couleurs (Rouge, Jaune, Vert, Bleu) et une cible, qui occupent des cases. Le plateau de jeu est entièrement entouré de murs.

Au début d'une session de jeu, on définit un plateau de jeu. Une manière (cf. plus bas) est de tirer aléatoirement un certain nombre d'arêtes du plateau de jeu (séparations entre deux cases) et de placer des murs à ces endroits.

Au début d'un tour de jeu, on tire au hasard une case du plateau qui est adjacente à exactement deux murs contigus et on place la cible dessus. On tire dans l'ordre, quatre cases du plateau et on place les quatre robots dessus. Puis on tire une couleur au hasard. Cela constitue l'énigme de ce tour (amener le robot de la couleur choisie sur la cible choisie).

Un coup est un déplacement d'un des quatre robots et est défini par une couleur et une direction (Haut, Bas, Gauche, Droite). Le robot de la couleur choisie se déplace depuis sa position actuelle dans la direction choisie et ne s'arrête que lorsqu'il rencontre un mur ou un autre robot (principe de "Sokoban"[2]).

Une solution de l'énigme est une série de coups. Une solution est bonne quand, à partir de la position initiale de l'énigme, la série de déplacement amène le robot dont la couleur a été tirée au sort au début du tour sur la cible.

Pour résoudre l'énigme, il est souvent utile de déplacer un robot dans une position où il pourra "bloquer" un déplacement d'un autre robot.

Protocole

Le serveur peut être initialisé avec un plateau de jeu (ce qui facilite les tests et la correction). Le serveur écoute sur le port 2016 en TCP. Clients et Serveur échangent selon un protocole texte. Une commande est composée de chaînes ASCII terminées par des `/`. La commande elle-même est terminée par un `\n`. Le protocole est non-ambigu. En cas de doute sur la complètude du protocole, écrire à l'équipe pédagogique.

Connexion

```
CONNEXION/user/
(C -> S) Nouvelle connexion d'un client nomme 'user'
BIENVENUE/user/
(S -> C) Validation de la connexion de 'user'.
CONNECTE/user/
(S -> C) Signalement de la connexion de 'user' aux autres clients.
```

Déconnexion

```
SORT/user/
(C -> S) Déconnexion de 'user'.
DECONNEXION/user/
(S -> C) Signalement de la deconnexion de 'user' aux autres clients.
```

Début d'une session

```
SESSION/plateau/
(S -> C) Plateau de la session courante et initialisation de la session pour le client.
VAINQUEUR/bilan/
(S -> C) Fin de la session courante, scores finaux de la session.
```

[2] https://fr.wikipedia.org/wiki/Sokoban

Phase de réflexion

`TOUR/enigme/bilan`
`(S -> C) Bilan de la session, description de l'enigme courante et initialisation de la phase de réflexion.`
`TROUVE/user/coups/`
`(C -> S) Annonce d'une solution trouvee par 'user' en 'coups' deplacements.`
`TUASTROUVE/`
`(S -> C) Validation de l'annonce par le serveur, fin de la phase de reflexion.`
`ILATROUVE/user/coups/`
`(S -> C) Signalement a un client que 'user' a annonce une solution, fin de la phase de reflexion.`
`FINREFLEXION/`
`(S -> C) Expiration du delai imparti a la reflexion, fin de la phase de reflexion.`

Phase d'enchere

`ENCHERE/user/coups/`
`(C -> S) Enchere d'une solution trouvee par 'user' en 'coups' deplacements.`
`VALIDATION/`
`(S -> C) Validation de l'enchere.`
`ECHEC/user/`
`(S -> C) Annulation de l'enchere car incoherente avec celle de 'user'.`
`NOUVELLEENCHERE/user/coups/`
`(S -> C) Signalement a un client d'une enchere.`
`FINENCHERE/user/coups/`
`(S -> C) Fin des encheres, le joueur actif est user.`

Phase de résolution

`SOLUTION/user/deplacements/`
`(C -> S) Envoi de la solution proposee par le joueur actif.`
`SASOLUTION/user/deplacements/`
`(S -> C) Signalement aux clients de la solution proposee.`
`BONNE/`
`(S -> C) Solution acceptee (a tous les clients), fin du tour.`
`MAUVAISE/user/`
`(S -> C) Solution refusee (a tous les clients), nouvelle phase de resolution, 'user' joueur actif.`
`FINRESO/`
`(S -> C) Plus de joueurs restants, fin du tour.`
`TROPLONG/user/`
`(S -> C) Temps depasse, nouvelle phase de resolution, 'user' joueur actif.`

Chaînes

- `user` est une chaîne de lettres identifiant un joueur.
- `plateau` décrit l'état du plateau pour la session composé d'une suite de "murs":
  - `(x,y,z)`, avec `x` entier, `y` entier et `z` appartenant à $\{\texttt{H}, \texttt{B}, \texttt{G}, \texttt{D}\}$ désigne le mur attenant à la case (x, y) du côté z.

  Par exemple `(3,4,H)(3,4,G)(12,6,H)` est une chaine décrivant trois murs, en haut et à gauche de la case $(3, 4)$ et en haut de la case $(12, 6)$.
- `enigme` décrit l'état du plateau pour le tour courant composé:

  - `(xr,yr,xb,yb,xj,yj,xv,yv,xc,yc,c)` 10 entiers décrivant les positions initiales des robots rouge, bleu, jaune, vert et de la cible (dans cet ordre) et une couleur `c` parmi {`R, B, J, V`}.

- `deplacements` décrit une solution d'un joueur composée d'une série de "déplacements":
  - `cz`, avec `c` appartenant à $\{\texttt{R}, \texttt{B}, \texttt{J}, \texttt{V}\}$, et `z` appartenant à $\{\texttt{H}, \texttt{B}, \texttt{G}, \texttt{D}\}$ désignant le déplacement du robot de couleur c dans la direction z.

  Par exemple, `RDRHVDVHVDRB` représente les déplacements suivants: le robot rouge bouge vers la droite, puis vers le haut, puis le robot vert se déplace vers la droite, vers le haut et vers la droite; enfin, le robot rouge se déplace vers le bas.

- `coups` est un entier décrivant le nombre de coups d'une solution.

- `bilan` décrit l'état courant de la session donné par `t` indiquant le tour courant et une série de `(i,s)` indiquant que le score du joueur `i` est `s`. Par exemple `6(saucisse,3)(brouette,0)` indique que la session en est au 6eme tour de jeu, que le joueur `saucisse` a un score de 3 et le joueur `brouette` a un score de 0.

Points d'intérêts

- **Faisabilité**: Il est probable, en tirant au hasard les murs, d'obtenir un plateau de jeu permettant le tirage d'énigmes impossibles (ou trop difficiles) à résoudre: absence de cases correctes pour la cible, non-connexité du plateau, ... Il peut être judicieux de faire en sorte que le serveur ne propose que des plateaux "intéressants", par exemple en remplaçant le tirage aléatoire par un tirage parmi des "bons" sous-plateaux mis en mémoire dans le serveur (on peut en trouver sur le web, ou les recopier depuis une boîte de jeu[3]), ou plus intéressant, en écrivant un générateur aléatoire suffisamment complexe pour donner régulièrement des bons plateaux.

- **Interface**: Sans être le sujet principal de ce devoir, l'interface du client doit être soignée, si possible: en affichant graphiquement le plateau de jeu avec l'énigme en cours, en montrant à l'aide d'une animation les solutions proposées, et en permettant à l'utilisateur d'entrer sa solution par une série de clic souris sur des boutons idoines, ou par des commandes clavier adéquates.

- **Concurrence**: Le serveur doit, evidemment, gérer les connexions simultanées. Les conflits (par exemple, des enchères de même valeur envoyées simultanément) doivent être gérées au mieux. L'architecture standard comporte l'utilisation d'un thread par client et d'une protection contre les courses pour les ressources partagées.

Extensions

Il est probable que la réalisation d'extension(s) implique un enrichissement du protocole. Il est important que, au maximum, clients et serveurs bénéficiant d'extensions soient compatibles avec ceux n'en bénéficiant pas. En dernier recours, il est possible de rendre deux versions des clients et serveurs ("compatible" et "enrichie").

- **Chat**: intégration d'un module de chat au jeu, permettant aux joueurs de communiquer entre eux.

- **Persistance**: système d'authentification pour les clients, mise en mémoire des statistiques des parties précédentes, possibilité de rejoindre une partie quittée précédemment.

- **Journal**: publication régulière des résultats des dernières parties sur une page web.

- **Extension du plateau**: rajouter des nouveaux composants au plateau: téléporteur, mur "diode", etc (s'inspirer de la version originale du jeu).

- **Client autonome**: programmation d'un client tricheur qui calcule, de manière combinatoire, une solution. C'est encore mieux si le tricheur peut se faire passer pour un humain (comportement d'enchère crédible, messages de chat).

[3] Demander au chargé de TD, le cas échéant.