Projet d'application Client-Serveur

SCS

4 avril 2008

1 Description du projet

Le but de ce projet est l'implantation de la communication permettant de jouer en réseau au jeu SIAM. Les spécifications du travail à réaliser sont définies dans la suite.

Chaque joueur est représenté par un processus, écrit en langage C. Le code du jeu de SIAM, appelé moteur, est en prolog et accède à Java pour communiquer, à travers des sockets, avec le processus joueur. Les parties sont administrées par un serveur arbitre. Le joueur prend en paramètre le nom de la machine et le numéro de port du processus arbitre pour s'y connecter. Des numéros de ports fixes seront attribués aux joueurs pour la préparation du tournoi final. Toutes les communications se feront en mode connecté.

Voici le comportement d'un processus joueur :

1. Il envoie à l'arbitre une requête **IDENTIFICATION** pour s'inscrire.
2. Lorsqu'il est prêt, il envoie une requête **PARTIE** à l'arbitre pour lui demander de jouer. En réponse l'arbitre lui donne les coordonnées du joueur (nom de la machine et numéro de port) avec lequel il doit jouer et lui indique la couleur du joueur, le joueur blanc commence. Si le tournoi est fini, l'arbitre en informe le joueur qui doit terminer son application.
3. Lorsque c'est son tour de jouer, le joueur consulte le moteur pour connaître le coup à jouer. Il envoie une requête **COUP** à l'arbitre puis il en attend la validation. Ensuite, il envoie la même requête à l'adversaire et en attend l'acquittement.
4. Il attend le coup de l'adversaire, le valide auprès de l'arbitre pour éviter toute erreur de transmission.
5. Il informe son moteur du coup joué par l'adversaire.
6. Si la partie est finie, le joueur retourne en 2 sinon il retourne en 3.

2 Protocole de l'arbitre

Voici le contenu des différentes requêtes/réponses du protocole de communication avec l'arbitre et l'adversaire :

```
/* Taille des chaines */
#define TAIL_CHAIN 30

/* Identificateurs des requetes */
typedef enum { IDENTIFICATION, PARTIE, COUP, COUP_ADVERS } TypIdRequest;
```

```
/* Types d'erreur */
typedef enum { OK,              /* Validation de la requete */
       ERR_NOM,        /* Nom inconnu */
       ERR_JOUEUR,     /* Identificateur de joueur inconnu */
       ERR_COUP,       /* Erreur sur le coup joue */
       ERR_TYP         /* Erreur sur le type de requete */
} TypErreur;

/*
 * Structures d'identification
 */
typedef struct {

  TypIdRequest idRequest;         /* Identificateur de la requete */
  char nom[TAIL_CHAIN];           /* Nom du joueur */
  char nomMachine [TAIL_CHAIN];   /* Nom de la machine du joueur */
  short noPort;                   /* Numero de port du joueur */

} TypIdentificationReq;

typedef struct{

  TypErreur err;                  /* Code d'erreur */
  int joueur;                     /* Identificateur du joueur */

} TypIdentificationRep;

/*
 * Structures demande de partie
 */
typedef enum { FAUX, VRAI } TypBooleen;

typedef struct{

  TypIdRequest idRequest;         /* Identificateur de la requete */
  int joueur;                     /* Identificateur du joueur */

} TypPartieReq;

/* Type des pieces */
typedef enum { RHINO, ELEPH } TypAnimal;

typedef struct {

  TypErreur err;                  /* Code d'erreur */
  int adversaire;                 /* Identificateur de l'adversaire */
  char nomMachineAdv[TAIL_CHAIN]; /* Nom machine de l'adversaire */
  short noPortAdv;                /* Numero de port de l'aversaire */
  TypBooleen finTournoi;          /* Indicateur de fin */
  TypAnimal typeAnimal;           /* Pour savoir qui debute la partie */

} TypPartieRep;

/*
```

```
 * Definition d'un point, d'apres le marquage sur le plateau
 */
typedef enum { A, B, C, D, E, L_NULL } TypLigne;
typedef enum { i, ii, iii, iv, v, C_NULL } TypColumn;
typedef enum { NORD, EST, SUD, OUEST, O_NULL } TypOrientation;

/*
 * Definition de la position d'une pièce
 */
typedef struct {

  TypLigne  l;   /* Identificateur de ligne */
  TypColumn c;   /* Numero sur la colonne */

} TypPosition;

/*
 * Definition du déplacement d'une pièce
 */
typedef struct {

  TypPosition posDepart;    /* Position de depart */
  TypPosition posArrivee;   /* Position d'arrivee */
  TypOrientation o;         /* Orientation */

} TypDeplace;

/*
 * Structures coup du joueur pour l'arbitre et adversaire
 * et de l'adversaire pour l'arbitre
 */

/* Propriete des coups */
typedef enum { ENTREE, ENTREE_POUSSEE, SORTIE, CHANGE, DEPLACE, POUSSEE, GAGNE, PERDU } TypPropCoup;

typedef struct {

  TypIdRequest idRequest;    /* Identificateur de la requete */
  TypPropCoup  propCoup;     /* Propriete du coup */
  TypDeplace   deplace;      /* Deplacement de la piece */

} TypCoupReq;

/* Validite du coup */
typedef enum { VALID, TIMEOUT, TRICHE } TypValCoup;

/* Reponse a un coup */
typedef struct {

  TypErreur err;                  /* Code d'erreur */
  TypValCoup validCoup;           /* Validite du coup */

} TypCoupRep;
```

Voici les règles retenues pour l'utilisation de ces types :

IDENTIFICATION le nom à donner est le login sur wesson du joueur, le nom de la machine et le numéro de port doivent permettre à l'adversaire d'établir une connexion avec le joueur. Cette requête permet à un joueur de se faire enregistrer dans le tournoi. Il obtient en retour son identifiant dans le tournoi. Le numéro de port est donné avec sa représentation sur la machine locale (attention pour établir vos connexions il sera nécessaire d'utiliser ce numéro de port sous sa forme réseau).

PARTIE l'identificateur est celui du joueur dans le tournoi. Cette requête permet au joueur de demander une nouvelle partie. S'il a déjà joué toutes ses parties le booléen `finTournoi` est positionné à vrai. Dans ce cas, il doit se déconnecter et finir son exécution ainsi que celle du moteur. Si la valeur de `typeAnimal` est `ELEPH`, il attend une connexion de la part de l'adversaire, puis il commence à jouer. Si la valeur de `typeAnimal` est `RHINO`, il demande à se connecter à l'adversaire dont les coordonnées sont données dans le message de réponse et attend le premier coup.

COUP Une requête `TypCoupReq` est envoyée à l'arbitre pour jouer un coup. Cette requête donne le type de coup joué et le déplacement de l'animal. Le déplacement est doné à partir de la position de l'animal joué, de sa position et de son orientation après le coup. La position d'un animal est identifiée par la case sur laquelle il se trouve. La valeur associée `propCoup` donne le type de coup qui est joué. Il peut s'agir d'une entrée, d'une sortie, d'un changement (case et/ou orientation), d'une poussée, d'un coup gagnant ou d'un coup perdant. Dans le cas d'une entrée, il n'est pas nécessaire d'initialiser la position de départ. Dans le cas d'une sortie, il n'est pas nécessaire d'initialiser la position de sortie et l'orientation.

En réponse à la requête, l'arbitre retourne une validation du coup `TypCoupRep` avec une valeur d'erreur initialisée à OK. Le joueur envoie alors la structure `TypCoupReq` à son adversaire qui l'acquitte avec un `TypCoupRep` initialisé à OK. L'adversaire attend le coup du joueur. Lorsqu'il le reçoit, il le transmet à l'arbitre avec l'`idRequest` positionné à `COUP_ADVERS` et attend la validation. A partir cet instant, il peut s'adresser à son moteur pour lui demander le prochain coup à jouer.

Si le type de coup est `FIN_PARTIE`, l'adversaire valide le coup au joueur et à l'arbitre, puis les joueurs peuvent demander une nouvelle partie.

Quelques précautions sont prises pour éviter les erreurs répétées, voire moins bien intentionnées :

- si un coup est erroné ou que le joueur annonce qu'il a gagné alors que c'est faux, la partie est perdue. L'arbitre retourne une valeur `TRICHE` dans le champ `validCoup` et en informe simultanément le joueur adverse. Chacun des joueurs doit alors demander une nouvelle partie.
- la recherche dans un moteur est limitée dans le temps. Si la limite est dépassée, le coup est considéré comme nul et l'arbitre retourne une valeur `TIMEOUT` dans le champ `validCoup`. Il envoie également à l'adversaire un message `TCoupRep` avec une valeur `TIMEOUT` dans le champ `validCoup`. Le temps d'attente est calculé à partir de l'instant où l'arbitre envoie la validation du coup adverse. Passé ce temps, la partie est perdue. Attention, un joueur peut donc recevoir un `TIMEOUT` alors qu'il est en train de jouer.
- si une structure coup avec des données différentes est envoyée à l'arbitre et au joueur adverse, l'arbitre refuse la validation demandée par l'adversaire et lui envoie, immédiatement après la structure `TypCoupRep` où le champ `validCoup` est initialisée à `TRICHE`, le message `TypCoupReq` et les structures d'alignement éventuelles qu'il a reçu du joueur. Puis le jeu continue. Ce protocole ne permet pas de détecter le tricheur, ce qui est impossible puisque chacun peut affirmer ce qu'il veut, mais il permet au jeu de se dérouler dans de bonnes conditions.

Vous devez développer le protocole de communication entre le joueur et son moteur. Les mes-

sages seront échangés sur des sockets avec un accès java du côté du moteur et un accès en langage C du côté du joueur.

3 Standard de programmation

Pour que vos programmes soient facilement lisibles, vous vous conformerez au standard de programmation également utilisé en Algorithmique Combinatoire.

Pour expliquer votre réalisation, vous nous remettrez un rapport (5-6 pages) expliquant l'utlisation du protocole de communication avec l'arbitre, la mise en place d'un protocole avec le moteur Java et la structure de votre joueur. La longueur de ce rapport n'est pas un facteur déterminant, portez plus d'attention à sa clarté et à la structuration de vos explications. Ce rapport est à remettre lors de votre dernier TP, sous forme électronique au format pdf. A cette occasion, vous présenterez votre travail.

Pour le jour du tournoi, vous nous fournirez votre joueur en le recopiant dans le répertoire `~lphilippe/pub/scs/projet/joueurs`. Pour cela, dans le répertoire : créer un sous-repertoire portant les deux noms du binôme. Dans ce sous-répertoire, mettre tous les fichiers de votre projet et un exécutable appelé joueur. Pensez à modifier les droits d'accès. Cet exécutable (qui peut être un script de lancement) permet de lancer à la fois votre joueur et le moteur avec les numéros de ports qui sont donnés dans le fichier liste (celui qui est donné et le suivant). ATTENTION, tout manquement à cette règle peut vous exclure du tournoi mais, si vous vous y prenez un peu à l'avance, il sera possible de nous demander une validation.