Maîtrise d'informatique
Documents autorisés

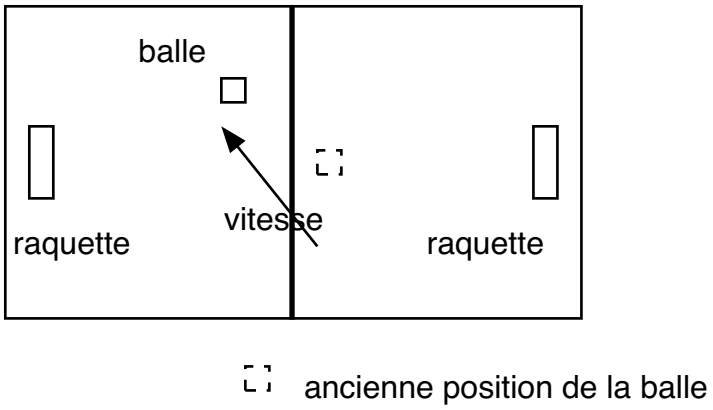
Programmation Objet et Distribution
Durée 3 heures

Université Pierre et Marie Curie
Année 2000-2001

Examen du 20 septembre 2001

Problème : Pong

Le premier jeu vidéo interactif a été la simulation d'une partie de Ping-Pong entre deux joueurs humains. Dans ce jeu chaque joueur est représenté par une raquette rectangulaire, la balle est un carré qui se déplace selon une trajectoire simple et rebondit sur les cotés de la table ou sur la raquette d'un joueur.



Quand la balle dépasse la ligne de la raquette du joueur, celui-ci a perdu le point. Une partie se déroule en 21 points avec 2 points d'écart entre joueurs.
Ce sujet se propose de réaliser une version de ce jeu en réseau sous la forme d'un client/serveur. Le serveur gère la table et la balle, les clients représentent les joueurs et peuvent avoir une action sur les raquettes, qui doit être confirmée par le serveur. Les communications entre client et serveur suivent le protocole texte suivant :

- contrôle de la partie : "DEBUT", "FIN" et "SCORE" suivi des 2 scores;
- position de la balle : "BALLE" suivie de la position en entiers;
- position d'une raquette : "RAQUETTE" suivi du numéro de la raquette suivi de sa position;
- mouvement d'une raquette : "GAUCHE" et "DROITE"

Seules les demandes des mouvements de la raquette locale sont envoyées par le client au serveur, toutes les autres communications se font dans le sens serveur vers client, y compris les positions réelles des deux raquettes.

Etape 1 : un serveur en O'Caml

Pour simplifier les entrées/sorties, on suppose connues les fonctions d'écriture d'une ligne et de lecture d'une ligne sur un descripteur de fichier de types suivants :

```
val write_line : Unix.file_descr -> string -> unit
val read_line : Unix.file_descr -> string
```

Le squelette du serveur est aussi donné sous forme d'une classe abstraite :

```
class  virtual serv_socket p  =
  object (self)
    val port = p
    val mutable sock = ThreadUnix.socket Unix.PF_INET Unix.SOCK_STREAM 0
    initializer
      let mon_adresse =
        (Unix.gethostbyname(Unix.gethostname())).Unix.h_addr_list.(0)
      in Unix.bind sock (Unix.ADDR_INET(mon_adresse,port)) ;
         Unix.listen sock 2
    method private client_addr = function
        Unix.ADDR_INET(host,_) -> Unix.string_of_inet_addr host
      | _ -> ''Unexpected client''
    method virtual run :unit -> unit
  end ;;
```

1. Ecrire une sous-classe concrète `serv1` de `serv_socket` qui définit la méthode `run` en acceptant 2 connexions sur la socket serveur `sock` et conserve dans des variables d'instance les sockets clientes obtenues.
2. On cherche à écrire la classe principale `pong` de déroulement du jeu. Pour l'instant on ne s'occupe que de la communication du serveur vers les clients. Le constructeur de cette classe aura comme paramètres les deux sockets clientes. Indiquez les éléments principaux de cette classe, en particulier comment la balle se déplace ou rebondit. Vous pouvez choisir un algorithme naif : la balle part toujours du même endroit avec la même vitesse, quand la balle rencontre un coté du jeu elle rebondit de manière symétrique, de même quand elle rencontre une raquette.

   Implantez cette classe en suivant vos indications.
3. Ecrivez une classe `serv2` qui hérite de la classe `serv1` et qui lance le jeu sur une instance de la classe `pong` sans tenir compte des réponses des clients.
4. Pour implanter complètement le serveur, on décide de lancer deux threads d'écoute des clients. Proposez une organisation de communication entre les threads du serveur pour tenir compte des mouvements des raquettes des clients.
5. Implantez la classe `serv3` pour un serveur complet.

Etape 2 : un client interactif en Java

1. Indiquez sur un dessin l'interface que vous aimeriez construire pour un client de ce jeu, en indiquant la nature de chaque élément graphique. Il est fortement conseillé pour la zone de la table d'utiliser un Canvas.
2. Ecrivez un client Java qui ne gère pas les interactions de l'utilisateur et n'affiche que les informations du serveur.
3. Indiquez comment modifier ce client pour gérer une interaction souris indiquant les mouvements du joueur. Le déplacement de la souris sur un composant AWT engendre une événement `MouseEvent` qui sera traité par la méthode `mouseMoved` des objets traitant les événements souris et respectant l'interface `mouseMotionListener`.
4. Implantez les modifications proposées.

Etape 3 : un client pour s'entrainer

Devant la difficulté de trouver des partenaires d'entrainement, on veut utiliser un client toujours disponible. Pour cela il est demandé d'écrire un programme joueur. Le langage d'implantation est laissé libre au choix du programmeur (O'Caml ou Java). Pour cela votre client doit avoir sa propre représentation du jeu.

1. Ecrire un premier client qui ne fait que conserver l'état du jeu en fonction des transmissions du serveur.
2. Modifier-le pour implanter une stratégie de jeu qui consiste à minorer la distance entre la raquette et la balle.
3. Proposez une meilleure stratégie pour votre client. Quels sont les changements à apporter à votre programme? Détaillez-les.