Sujet de projet informatique : Chasse au trésor en réseau

Sujet proposé par Vincent Cheval

Objectif : L'objectif de ce projet est de créer un logiciel permettant de jouer à une chasse au trésor en réseau. Le logiciel inclura :

- une partie serveur qui génère les parties, accepte les connexions de client "humain" et peut aussi générer des joueurs "Ordinateurs".
- une partie client permettant de se connecter à un serveur et de participer à la chasse au trésor.

Réalisation technique : Le projet doit normalement être réalisé en Java. Cela peut être discuté.

Principe : La chasse au trésor est modélisé par un plateau de jeu rectangulaire avec un nombre prédéfini de cases. Une case peut soit être vide, soit contenir un trésor, soit correspondre à un mur ou enfin correspondre à un trou. Les trésors ont plusieurs valeurs possibles (5, 10, 15 et 20).

Au lancement de la partie, le serveur place aléatoirement sur des cases vides les joueurs. Les joueurs doivent alors se déplacer sur le plateau de jeu pour aller récupérer le plus de trésors avec les conditions suivantes :

- Les déplacements ne peuvent être que verticals ou horizontals
- Les joueurs ne peuvent traverser un mur
- Deux joueurs ne peuvent se trouver sur la même case
- Si un joueur tombe dans un trou, il est éliminé
- Lorsque tous les trésors ont été récupérés, la partie est terminée ; le joueur ayant le plus de point gagne.

On considéra plusieurs modes de jeu.

1. *Le speeding contest* : Dans ce mode de jeu, tout le monde a la vision complète du plateau avec les trous, murs et trésors. Il faut juste être le plus rapide pour gagner. On interdira le mix entre joueurs "Humain" et "Ordinateur" pour ce mode.
2. *Le tour par tour* : A nouveau, tout le monde a la vision complète du plateau mais on doit attendre son tour pour jouer.
3. *Le brouillard de guerre* : Dans ce mode de jeu, les joueurs connaissent uniquement la position et la valeur des trésors. Chaque joueur peut voir les murs et joueurs dans un rayon de deux cases autour de lui. En

revanche, il ne connait que le nombre de trous autour de lui (dans un rayon d'une case). Pour aider, les joueurs peuvent dépensés des points de trésors pour révéler :
- les pièges autour de lui pendant 5 tours
- une partie de la carte pendant 3 tours. Pendant ces trois 3 tours, le joueur peut voir la position des joueurs, les tours et les trésors encore présents.

Le prix des deux commandes pourra être modifié à la génération de la carte.

Méthodologie : On commencera par l'élaboration des modes de jeu *speeding contest* et *tour par tour*.

On donnera une spécification pour les communications entre clients et serveurs qu'il faudra suivre scrupuleusement. Cela permettra en l'occurrence de pouvoir utiliser des clients et serveurs implémentés par différentes personnes de participer au même jeu. Par example, la spécification inclue les points suivants :
- la communication entre clients et serveurs se fera via socket ;
- il n'y aura pas de communication directement entre clients, uniquement client-serveur ;
- un serveur devra être capable d'accepter un nombre arbitraire de client avant de démarrer la partie ;
- les clients et serveurs ne communiqueront que via une liste prédéfinie de commandes qui sera donnée au début du projet.
- ...

Vous serez libre de programmer comme vous le souhaitez la structure interne du client et du serveur, DU MOMENT QUE le client et le serveur satisfont la spécification. Néanmoins, certaines librairies, structures de données et algorithmes seront proposés/conseillés pour vous faciliter la tâche. Une représentation graphique du jeu ne sera pas obligatoire (visualisation en texte conviendra) mais elle sera la bienvenue.

Objectifs minimaux : Il sera donc demandé au minium d'implémenter un serveur et un client (pour humain) qui suivront la spécification qui auquel on pourra jouer. Il faudra implémenter le speeding contest et le tour par tour.

Objectifs supplémentaires : Vous pourrez ensuite implémenter le mode de jeu *brouillard de guerre*, une représentation graphique et les joueurs "Ordinateur". Pour ces derniers, vous êtes libre de choisir comment implémenter l'intelligence artificielle des joueurs "Ordinateur". Si vous avez plusieurs idées, vous pouvez alors faire des compétitions entre joueurs "Ordinateur" et évaluer vos performances. Dans ce cas, il ne sera pas nécessaire d'implémenter

tous les modes de jeu.

1 Liste des commandes

La spécification de la communication entre le serveur et les clients se fait via une liste de commandes. Pour faciliter la lecture, on notera S et C le serveur et client respectivement.

Phase d'identification : Lorsque le client se connecte au serveur, il doit tout d'abord s'identifier en donnant son nom de joueur.

$$C \xrightarrow{\textit{100 HELLO PLAYER Bob}} S$$
$$S \xrightarrow{\textit{101 WELCOME Bob}} C$$

Le serveur doit retourner la commande *990 Too large words in command* si la taille du nom dépasse 30 caractères ; ou la commande *901 This name is already used* si le nom déclaré par le joueur est déjà utilisé par un autre joueur.

Création ou joindre une nouvelle partie : Un joueur peut créer une nouvelle carte via la commande suivante

$$C \xrightarrow{\textit{110 CREATE m SIZE x y HOLE h TRES n}} S$$
$$S \xrightarrow{\textit{111 MAP CREATED id}} C$$

où x et y sont la largeur et hauteur de la carte, h est le nombre de trous, n est le nombre de trésors et m le mode de jeu (1 pour speeding contest, 2 pour tour par tour, 3 pour brouillard de guerre). Le serveur retourne un nombre représentant la nouvelle carte.

Un joueur peut demander la liste les parties joignables :

$$C \xrightarrow{\textit{120 GETLIST}} S$$
$$S \xrightarrow{\textit{121 NUMBER k}} C$$
$$S \xrightarrow{\textit{121 MESS 1 ID id m x y h n}} C$$
$$\dots$$
$$S \xrightarrow{\textit{121 MESS k ID id m x y h n}} C$$

Une carte par message.

Ensuite il peut rejoindre la partie de son choix :

$$C \xrightarrow{\textit{130 JOIN id}} S$$
$$S \xrightarrow{\textit{131 MAP id JOINED}} C$$

Lorsqu'un joueur rejoint une carte, le nom du nouveau joueur doit être annoncé (broadcasté) aux autres joueurs connectés à cette partie.

$$S \xrightarrow{\textit{140 Bob JOINED}} C$$
$$C \xrightarrow{\textit{141 Bob ACK}} S$$

Le créateur de la carte peut alors demander le lancement de la partie :

$$C \xrightarrow{\textit{150 REQUEST START}} S$$

Le serveur broadcast à tous les joueurs la requête :

$$S \xrightarrow{\textit{152 START REQUESTED}} C$$
$$C \xrightarrow{\textit{152 START r}} S$$

où r est YES ou NO. Si tous les joueurs répondent YES alors le serveur broadcast la commande annonçant que je jeu démarre.

$$S \xrightarrow{\textit{153 GAME STARTED}} C$$

Sinon le démarrage est annulé et le nom des joueurs ayant refusé est donné à tout le monde.

$$S \xrightarrow{\textit{154 START ABORDED k}} C$$
$$S \xrightarrow{\textit{154 MESS 1 PLAYER p1 p2 p3 p4 p5}} C$$
$$\dots$$
$$S \xrightarrow{\textit{154 MESS k' PLAYER p1 p2 p3 p4 p5}} C$$

Où $k' = \lceil k/5 \rceil$.

Informations initiales Un joueur peu demander la liste des trous, des trésors ou des murs sur la carte.

$$C \xrightarrow{\textit{400 GETHOLES}} S$$
$$S \xrightarrow{\textit{401 NUMBER k}} C$$
$$S \xrightarrow{\textit{401 MESS 1 POS x1 y1 ... x5 y5}} C$$
$$\dots$$
$$S \xrightarrow{\textit{401 MESS k' POS x1 y1 ... x5 y5}} C$$

Où $k' = \lceil k/5 \rceil$.

$$
\begin{aligned}
&C \xrightarrow{\textit{420 GETWALLS}} S \\
&S \xrightarrow{\textit{421 NUMBER k}} C \\
&S \xrightarrow{\textit{421 MESS 1 POS x1 y1 ... x5 y5}} C \\
&\dots \\
&S \xrightarrow{\textit{421 MESS k' POS x1 y1 ... x5 y5}} C
\end{aligned}
$$

Où $k' = \lceil k/5 \rceil$.

$$
\begin{aligned}
&C \xrightarrow{\textit{410 GETTREASURES}} S \\
&S \xrightarrow{\textit{411 NUMBER k}} C \\
&S \xrightarrow{\textit{411 MESS 1 POS x1 y1 v1 ... x5 y5 v5}} C \\
&\dots \\
&S \xrightarrow{\textit{411 MESS k' POS x1 y1 v1 ... x5 y5 v5}} C
\end{aligned}
$$

Où $k' = \lceil k/5 \rceil$.

Les x et y représentent la position dans la carte. Pour les trésors, les variables v représentent le type de trésors (0 pour valeur inconnue ou bien sinon le chiffre indique la valeur du trésor).

Se déplacer sur la carte Un joueur peut se déplacer sur la carte en utilisant l'une de ces commandes :

$$
\begin{aligned}
&C \xrightarrow{\textit{200 GORIGHT}} S \\
&C \xrightarrow{\textit{200 GOLEFT}} S \\
&C \xrightarrow{\textit{200 GOUP}} S \\
&C \xrightarrow{\textit{200 GODOWN}} S
\end{aligned}
$$

Le serveur répond par l'une de ces commandes :

$$
\begin{aligned}
&S \xrightarrow{\textit{201 MOVE OK}} C \\
&S \xrightarrow{\textit{202 MOVE BLOCKED}} C \\
&S \xrightarrow{\textit{203 MOVE OK TRES v}} C \\
&S \xrightarrow{\textit{666 MOVE HOLE DEAD}} C
\end{aligned}
$$

Lorsque le joueur est sur une case trésor, la valeur du trésor est donnée (même si elle était déjà connue). De plus, lorsque le mouvement est validé,

la nouvelle position du joueur est broadcasté aux autres joueurs.

$$S \xrightarrow{\textit{510 Bob POS x y}} C$$
$$S \xrightarrow{\textit{511 Bob POS x y TRES v}} C$$
$$C \xrightarrow{\textit{512 Bob UPDATED}} S$$

Notons que le joueur a découvert la valeur d'un trésor, cette valeur est également broadcasté avec la position du joueur. A noté qu'au début de la partie (sauf pour le mode brouillard de guerre), la position de tous les joueurs est également broadcasté via cette commande.

Lorsqu'un joueur tombe sur un trou, il faut broadcaster la mort du joueur

$$S \xrightarrow{\textit{520 Bob DIED}} C$$
$$C \xrightarrow{\textit{521 Bob UPDATED}} S$$

Fin de partie Lorsque la partie se termine, il faut broadcasté le vainqueur :

$$S \xrightarrow{\textit{530 Bob WINS}} C$$
$$C \xrightarrow{\textit{600 GAME OVER}} S$$

Erreur Si le message reçu ne correspond à aucune commande, le serveur / client retourne la commande *999 Command Error*.

Commandes spécifiques au mode tour par tour Le serveur broadcaste qui a le droit de jouer.

$$S \xrightarrow{\textit{500 Bob TURN}} C$$
$$C \xrightarrow{\textit{501 TURN UPDATED}} S$$

De plus, si le serveur reçoit une commande de mouvement d'un joueur alors que ce n'est pas son tour, il retourne la commande d'erreur *902 Not your turn.*

Commandes spécifiques au mode brouillard de guerre En plus des commandes au tour par tour, il faut ajouter une commande pour dépenser des points pour révéler les pièges autour du joueur

$$C \xrightarrow{\textit{300 REVEAL HOLE}} S$$
$$S \xrightarrow{\textit{301 PAYMENT VALIDATED}} C$$

$$C \xrightarrow{\textit{310 REVEAL MAP x y}} S$$
$$S \xrightarrow{\textit{311 PAYMENT VALIDATED}} C$$

La commande *905 Not enough point* peut être retourné s'il n'y a pas assez de point pour acheter la commande.

Au début de chaque tour correspondant à la commande, le serveur transmettra les informations demandés en ré-utilisant les commandes de broadcast pour la position des joueurs et la commande :

$$S \xrightarrow{\textit{320 SENDING HOLES}} C$$
$$S \xrightarrow{\textit{320 NUMBER k}} C$$
$$S \xrightarrow{\textit{320 MESS 1 POS x1 y1 ... x5 y5}} C$$
$$\dots$$
$$S \xrightarrow{\textit{320 MESS k POS x1 y1 ... x5 y5}} C$$
$$C \xrightarrow{\textit{321 HOLE RECEIVED}} S$$

Pour les murs :

$$S \xrightarrow{\textit{330 SENDING WALL}} C$$
$$S \xrightarrow{\textit{330 NUMBER k}} C$$
$$S \xrightarrow{\textit{330 MESS 1 POS x1 y1 ... x5 y5}} C$$
$$\dots$$
$$S \xrightarrow{\textit{330 MESS k POS x1 y1 ... x5 y5}} C$$
$$C \xrightarrow{\textit{331 WALL RECEIVED}} S$$

Pour les trésors :

$$S \xrightarrow{\textit{340 SENDING TRES}} C$$
$$S \xrightarrow{\textit{340 NUMBER k}} C$$
$$S \xrightarrow{\textit{340 MESS 1 POS x1 y1 v1 ... x5 y5 v5}} C$$
$$\dots$$
$$S \xrightarrow{\textit{340 MESS k POS x1 y1 v1 ... x5 y5 v5}} C$$
$$C \xrightarrow{\textit{341 TRES RECEIVED}} S$$

2 Structure du réseau

Comme mentionné plus haut, la communication entre le serveur et les clients. Pour cela on pourra utiliser les deux packages :

```
import java.net.*;
import java.io.*;
```

Avec ces packages, on utilise deux classes pour la gestion des sockets. La classe *serverSocket* du coté serveur qui va attendre les connections des clients ; et la classe *clientSocket* du coté du client.

```
private ServerSocket serverSocket;
serverSocket = new ServerSocket(port);
serverSocket.accept();
        // Bloque jusqu'à accepter la connexion d'un client

private Socket clientSocket;
clientSocket = new Socket(ip, port);
```

On utilisera également des buffers pour lire et écrire à travers ces sockets.

```
private PrintWriter chOut;
private InputStreamReader streamIn;
private BufferedReader chIn;

chOut = new PrintWriter(clientSocket.getOutputStream(), true);
streamIn = new InputStreamReader(clientSocket.getInputStream());
chIn = new BufferedReader(streamIn);
```

On peut alors écrire et lire sur chOut et chIn respectively en utilisant les commandes suivantes

```
chIn.readLine();
chOut.println("hello client");
```