Analyse de projet : TinySnake

Introduction

Ce projet est le troisièmes projet du cours de programmation d'applications distribuées en réseaux. Lors du dernier projet, il a fallut concevoir un jeu Snake en réseau pour quatre joueurs. Le système de communication était le RMI. Le RMI, bien que simple à mettre en place coûte chère en performance et fait une surcharge du réseau du à l'utilisation du registre RMI. C'est pour alléger ce réseau que le projet est modifié afin d'utiliser la connexion par socket.

Outils utilisés

- IDE : Visual Studio Code ;
- Langage : Java 15 ;
- Environnement de développement : Windows 10.

Compilation

Le projet s'exécute via un makefile. Il contient les règles suivante :

- *cc-all* : Compilation du client et du serveur.
- *cc-client* et *cc-server* : Compilation du client ou du serveur au choix.
- *run-client* et *run-server* : démarrage du client ou du serveur.
- *javacoc-client* et *javadoc-server* : Génération de la java doc pour le clients ou le serveur.

L'ip du serveur peut être spécifié à la suite de la commande tun-client à l'aide de `IP=xxx`.

Méthodologie utilisée

Protocole TCP / IP

L'utilisation des socket dans la programmation réseau, permet le choix entre le protocole UDP ou TCP. Pour ce projet c'est le second qui a été retenu. Bien qu'il soit un peu plus lourd, il est vital que les paquets ne ce perdent pas en route dans un jeu vidéo, ou cela pourrait amener à des bugs.

Transferts des données

Les données envoyées sont des liste de bytes, mais afin de savoir quel type de données sont échangées, chaque paquet est divisé en deux tableau. Le premier est le header, il contient un nombre entier qui indique la nature du paquet. Le second est le body, il contient la données en lui même.

Tableau des headers du client vers le serveur :

Valeur	Signification
1	Le client s'enregistre avec un Pseudo (body : `String`)
2	Le client est prêt à jouer (body : `LinkedList<Object>`)
3	Le client appuie sur une touche (body : `LinkedList<Object>`)
4	Le clients demande le plateau de jeu (body : `UUID`)
5	Le client demande la liste des scores actuelle (body : `UUID`)
6	Le client demande si il y a eu un game-over (body : `UUID`)
7	Le client quitte la partie (body : `LinkedList<Object>`)
8	Le client demande une nouvelle partie (body : `UUID`)
9	Le client demande le podium (body : `void`)

Tableau des headers du serveur vers le client :

Valeur	Signification
1	Envoie des identifiants au nouveau client (body : `Map<UUID, UUID>`)
2	Un nouveau client est près a jouer (body : `LinkedList<Object>`)
3	Le jeu commence (body : `Boolean`)
4	Il y a une update dans l'interface graphique (body : `int`)
5	Envoie le plateau de jeu (body : `PlayArea`)
6	Envoie la liste des scores (body : `Map<String, int>`)
7	Envoie si il y a eu un game-over ou non (`Boolean`)
8	Envoie un nouveau jeu (`UUID`)
9	Envoie des meilleurs scores (body : `Map<String, int>`)

UUID

Afin de pouvoir identifier chaque joueur connecté et chaque jeu en cours d'exécution, la librairie `java.utils.UUID` est utilisé. Lorsqu'un joueur ce connecte et rejoint une partie, le serveur lui envoie deux identifiants unique. Un pour le client, et un pour la partie. Lorsque le joueur a fini une partie et redémarre une nouvelle, il garde son identifiant et reçois un nouvel identifiant de jeu.

Architecture du projet

Le projet est divisé en deux parties. Le client et le serveur. La partie client ne fait que afficher et envoyer des données au serveur. En aucun cas un calcul est effectué de ce coté. Une fois que le serveur reçois la donnée, celui-ci la traite et renvoie le résultat au(x) client(s). L'entièreté du projet peut être vu comme un modèle multi-tiers. La partie client est la couche de présentation, et le serveur contient la couche métier et persistance. Certains modèles ne sont présent que dans la partie serveur, d'autre uniquement dans la partie client, alors que d'autre sont utilisé des deux coté. Une classe statique `Tools` est aussi présentes des deux coté, elle contient des outils qui peuvent être utilisés dans n'importe quel projet et qui ne sont pas disponible en java.