Rapport de projet PC2R

Jeu de lettre

Kévin GESNOUIN -- Alexis MALAMAS

06/04/2017

SOMMAIRE

1. Description Générale

L'objectif du projet est de réaliser une application de type client-serveur permettant de jouer au jeu du Scrabble entre plusieurs utilisateurs. Chaque client possède sa propre interface sur laquelle se trouvent un plateau et un tirage courant de 7 lettres. Les utilisateurs proposent des placements sur le plateau afin de former des mots. À la fin d'un tour chaque utilisateur augmente son score selon sa proposition et le plateau est mis à jour suivant le meilleur placement. Une session de jeu se termine lorsqu'il n'y a plus aucune lettre de disponibles ou lorsqu'il n'y a plus d'utilisateurs connectés au serveur.

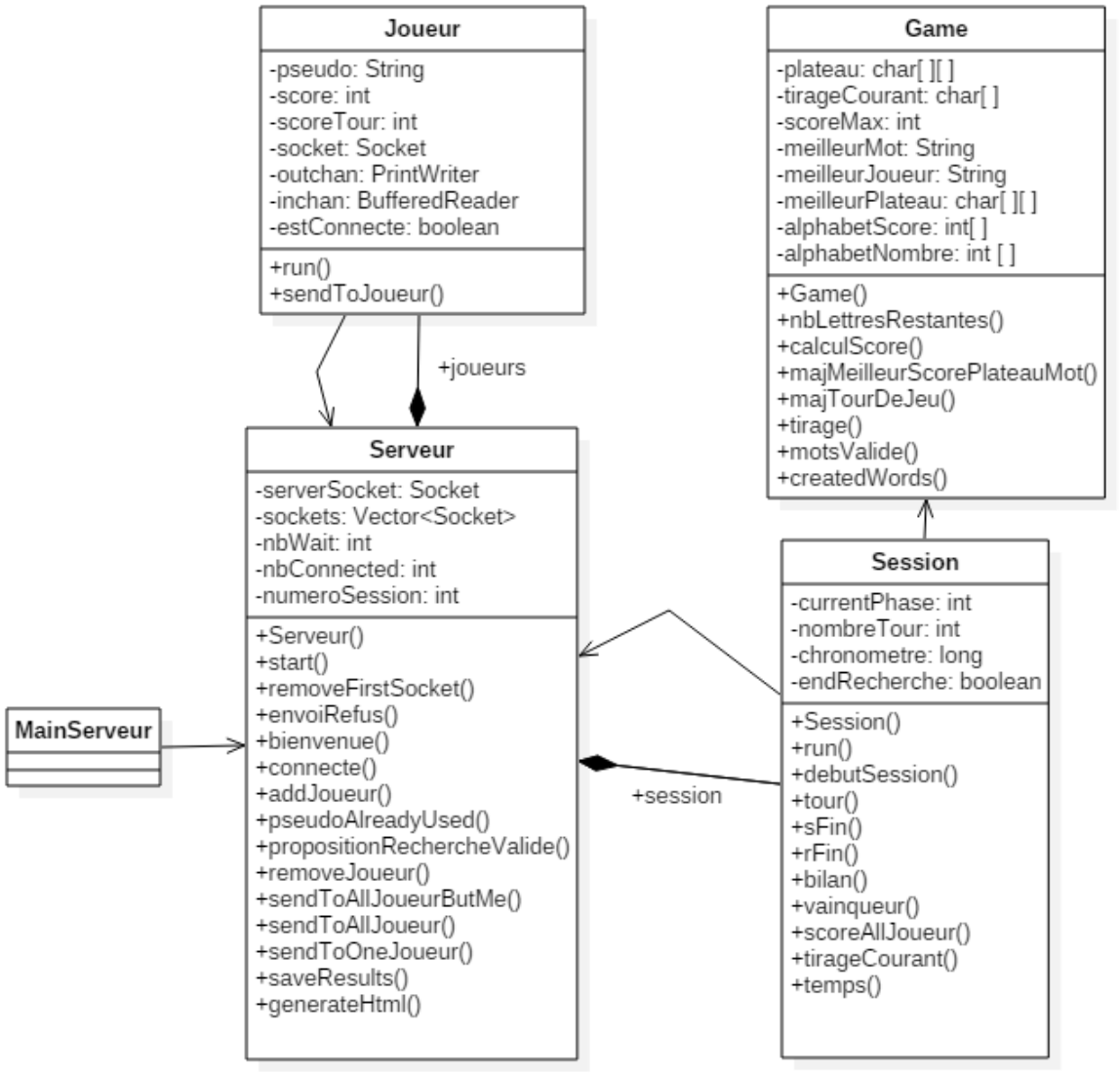
En plus du jeu de base des fonctionnalités ont été ajouté. En effet côté client, l'utilisateur peut communiquer avec les autres utilisateurs de façon publique ou privée via la console. Côté serveur, nous avons opté pour un dictionnaire en ligne lorsque le serveur peut se connecter à Internet sinon il utilise un dictionnaire en locale. Les résultats des sessions sont consultables sur une page web. On peut y retrouver les scores de chaque participant des sessions précédentes.

Pour réaliser ce projet, nous avons choisi d'utiliser le langage JAVA côté serveur et le langage C côté client ainsi que la librairie SDL 2.0 pour réaliser l'interface graphique de notre application.

Le code source du projet est également disponible sur GitHub à partir du lien suivant : https://github.com/AlexisMalamas/ClientServeur.

2. Hiérarchie des classes

Notre serveur possède un pool de thread de joueurs et une session. Lors du lancement du serveur, le pool de thread est créé et la session sera créée lorsque le premier joueur se connectera. De plus la session possède un attribut de la classe Game permettant d'appeler les fonctions propres au jeu du scrabble.



Lorsqu'un joueur est le premier à se connecté, le serveur crée un objet de la classe Session qui est un Thread. Les Threads de la classe Joueur sont responsables de l'écoute des clients selon le protocole de l'énoncé.

3. Problèmes rencontrés

Présentation des différents problèmes rencontrés lors de la réalisation du projet.

Côté serveur

La mise en place des pools de threads nous a posé quelques problèmes. Au début de la réalisation, nous avons opté pour lorsqu'un client se connecte, un thread est créé pour ce client. En effet dans le sujet, il n'y a pas de limite de joueurs mais la machine est limitée par sa capacité. C'est pourquoi nous avons choisi de mettre en place un pool de threads. Nous avons donc dû revoir toute l'organisation du serveur et interactions entre la classe Serveur et la classe Joueur.

Nous avons également rencontré des problèmes lorsqu'un joueur se déconnecte d'une session et qu'un autre joueur souhaite se connecter. En effet, lorsque le joueur se déconnecte, le thread qui était associé à ce joueur doit pouvoir s'occuper d'un autre joueur qui arriverait. Notre problème était que le thread restait connecté au joueur même s'il s'était déconnecté. Pour régler cela nous avons mis en place des variables pour voir si notre joueur est déconnecté afin que le thread puisse savoir quand un joueur est déconnecté et peut traiter un autre client.

Un autre problème rencontré était lorsqu'un utilisateur se déconnecte de façon anormale, c'est-à-dire sans envoyer un message de protocole SORT au serveur avant de se déconnecter du serveur. En effet, on obtenait une boucle infinie de message d'erreur dans la console. Pour régler ce problème nous avons mis en place une variable "normalDeco" qui permet de déterminer si l'utilisateur s'est déconnecté de façon anormale. Grâce à cette variable le serveur permet au thread de ne plus écouter l'utilisateur qui s'est déconnecté anormalement.

Côté client

Les problèmes les plus importants rencontrés côté client étaient les problèmes liés à l'utilisation de la SDL. En effet, au début du projet, nous n'avions pas remarqué que le programme consommer de plus en plus de RAM au cours de son exécution. Ce qui fait lors des tests de débogage, au bout d'un certain temps les textes sur la fenêtre disparaissaient d'un seul coup. On avait donc des problèmes de fuites mémoire puisque dans le gestionnaire des tâches on pouvait constater que le programme demander de plus en plus de ressources jusqu'à utiliser 2GB de RAM. Pour résoudre le problème nous avons parcouru plus en détail la documentation de la SDL. Nous avons donc constaté après cela, que dans le code nous avions oublié de détruire la texture liée à l'affichage d'un texte après son rendu sur l'écran. Les autres problèmes rencontrés côtés client étaient des bugs mineurs liés à l'utilisation des chaines de caractères en C.

4. Manuel d'utilisateur

Installation

Vous pouvez trouver les fichiers C et Java du projet dans la session 3603412 dans un dossier de nom ProjetPC2R.

Lancement sur Linux :

Pour le client il est nécessaire de faire les commandes suivantes :

sudo apt-get install libSDL2-dev
sudo apt-get install libSDL2-image-dev
sudo apt-get install libSDL2-ttf-dev
sudo apt-get install libSDL2-net-dev

Ensuite pour compiler il faut faire les commandes suivantes :
gcc -c connexion.c
gcc -c window.c
gcc -c main.c
gcc -o connexion. window.o main.o `sdl2-config –cflags –libs` -lSDL2_image -lSDL2_net -lSDL_ttf -pthread
./main

Pour le serveur un fichier jar "serveur.jar " est mise à disposition. Pour le lancer, il suffit de faire la commande java -jar serveur.jar

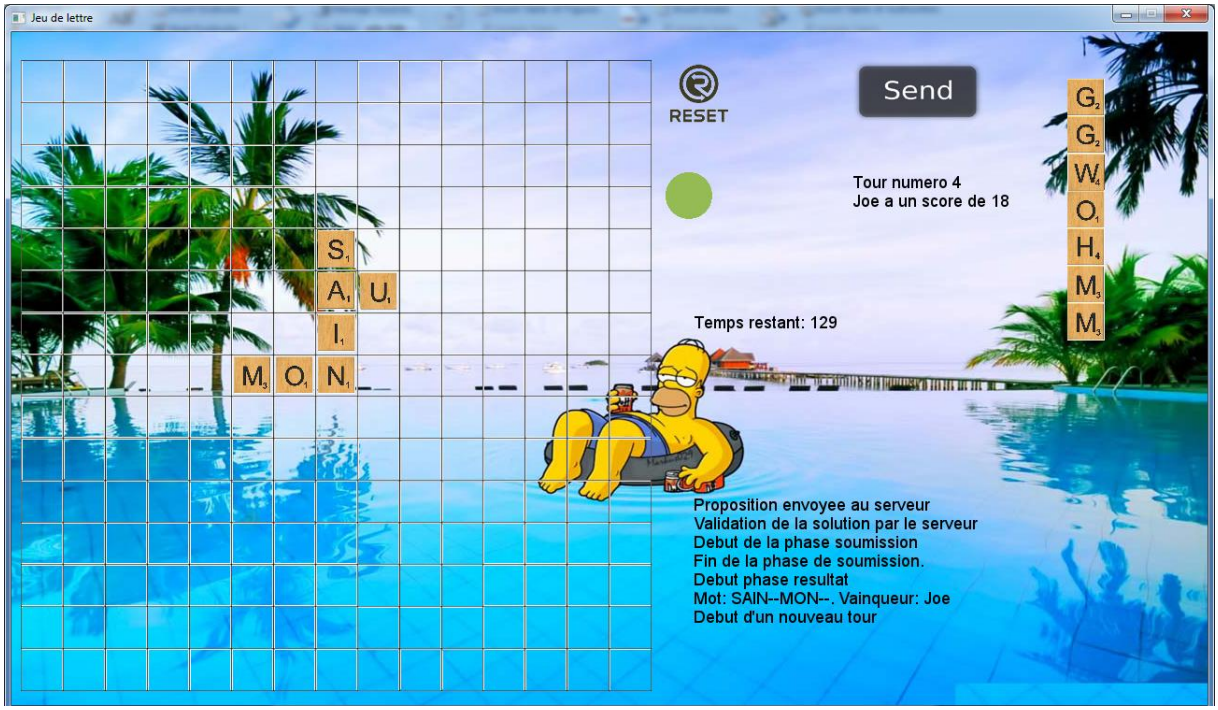
Lancement sur Windows :

Pour le client, il suffit d'utiliser l'IDE codeblocks et de lancer le projet avec le fichier Projet_SDL2.cbp. Toutes les librairies sont déjà fournies et les dépendances sont déjà effectuées.

Pour le serveur, il suffit d'ajouter le projet dans l'IDE Eclipse et de lancer le MainServeur.java

Interface client

Une fois l'exécutable lancé, il vous suffit d'entrer votre pseudo de joueur dans la console. Si le serveur accepte votre pseudo, une fenêtre de jeu apparait.



Capture d'écran de la fenêtre de jeu

La grille de jeu

Sur la partie gauche de la fenêtre vous avez la grille de jeu qui sera vide si vous êtes le premier à rejoindre une session ou si la partie n'en est encore qu'au premier tour de jeu.

La boite de dialogue

En bas à droite de la fenêtre se trouve la boîte de dialogue. C'est ici que seront affichés les messages du serveur mais également des autres utilisateurs. La boîte de message affiche un nombre limité de messages les uns en dessous des autres dans leur sens d'arrivée. Une fois la limite atteinte, la boîte de dialogue est nettoyée et de nouveaux messages peuvent alors s'afficher.

Temps et meilleur mot

Au-dessus de la boîte de dialogue est affiché le temps restant de la phase courante et au-dessus se trouve un voyant (rond) qui est de couleur verte si vous êtes la personne qui a

proposé le meilleur mot. Ce voyant est rouge si vous n'avez pas encore proposé de mot ou si votre mot proposé n'est pas le meilleur.

Scores

À droite du voyant d'indication de meilleur mot est affiché le numéro du tour courant et en dessous votre score mais aussi celui des autres joueurs. Les scores sont mis à jour au fur et à mesure de la partie. Remarque: vous gagnez des points même si vous n'êtes pas le meilleur mot de la partie.

Lettres disponibles

Entièrement à droite de la fenêtre, se trouvent des lettres affichées en colonnes. Ces lettres correspondent aux lettres disponibles pour le tour courant. Pour placer une lettre sur le plateau de jeu, il vous suffit de cliquer dessus. La lettre disparaît alors des lettres disponibles et ensuite il ne vous reste plus qu'à cliquer sur une case du plateau de jeu pour placer la lettre que vous avez choisie.

Envoi d'une proposition

En haut à gauche des lettres disponibles se trouve le bouton "send" qui permet d'envoyer votre proposition au serveur. Le serveur indiquera s'il valide ou non votre proposition dans la boîte de dialogue. Lorsque le serveur accepte votre proposition, vous pouvez toujours essayer de trouver une solution de meilleur score que celle que vous avez proposée. Le serveur gardera en mémoire la solution de meilleur score.

À gauche du bouton "send" se trouve le bouton "reset". Ce bouton est utile lorsque l'on place une lettre au mauvais endroit sur le plateau. Il permet de remettre le plateau de jeu comme il l'était au début du tour.

Envoi d'un message

Pour envoyer un message public ou privé il faut utiliser la console. Un message peut être envoyé à tout moment du jeu aux autres utilisateurs.

Message public : écrire le message dans la console puis appuyer sur entrée.

Message privé : écrire dans la console selon le modèle suivant: nom_destinataire/message

Exemple : dans la console ===> bob/bonjour

Envoi d'un message privé qui dit "bonjour" à bob

Les messages privées et publics des utilisateurs sont affichés dans la boîte de dialogue de la fenêtre du client.

5. Extensions

Pour ce projet nous avons réalisé les deux extensions nécessaires qui sont le système d'échange de message entre utilisateurs et la fonctionnalité de meilleur mot qui permet à un utilisateur de savoir en temps réel si son mot proposé est le meilleur.

Nous avons également réalisé deux extensions facultatives qui sont le dictionnaire distant et le journal.

Dictionnaire distant

Pour vérifier les mots proposés et formés par l'utilisateur le serveur se connecte à un dictionnaire à distance en utilisant l'adresse internet d'un fichier texte contenant les mots du dictionnaire. Si la connexion à internet échoue, le serveur utilise un dictionnaire local.

Journal

Lors de la fin d'une session, les résultats des joueurs sont publiés sur une page web. En effet lorsqu'une session se termine, une fonction permet d'ajouter dans un fichier HTML les résultats de la session. On pourra ainsi sur cette page consulter le nombre de tours, les pseudos et les scores des joueurs ayant participé à la session. Les joueurs ayant quitté en cours de session auront également leurs scores affichés dans le tableau de la session.

La fonction ne fait qu'ajouter dans le fichier HTML, elle n'écrase donc pas les résultats des sessions précédentes et ceux même si on relance le serveur plusieurs fois. Ce journal participe donc aussi à la persistance de l'application car il met en mémoire les statistiques des parties précédentes.

Page web affichant les résultats des sessions