Université Pierre et Marie Curie

Projet PC2R

Client/Serveur iSketch

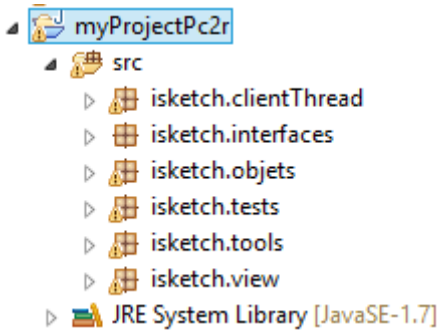
Jérôme Rahault
15/12/2013

Choix réalisation et Présentation de l'application :

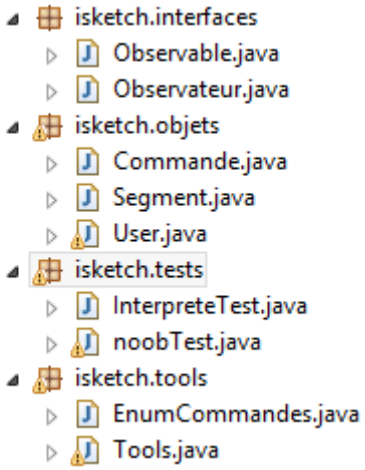
J'ai choisi de coder le client en Java car c'est le langage que je maitrise le mieux pour réaliser une interface graphique. J'ai codé le serveur en C car c'est celui que je maitrise le mieux après Java.

L'interface graphique utilise Swing pour plus de simplicité et un résultat correct.

J'ai opté pour une réalisation plutôt minimaliste, au détriment de la généricité, pour que le code soit plus lisible. Je n'utilise presque pas d'interface ni de classe abstraite et pas non plus de factory ou de MVC. Le projet contient donc très peu de classe regroupée en trois principaux pôles. Un sur le client à proprement parlé avec ses buffers et sa socket (package clientThread). Un sur l'interface graphique contenant la fenêtre et les panneaux qu'elle utilise (package view). Et le dernier sur les outils utilisés par les deux premiers (packages interfaces, objets et tools). Je commencerai par présenter ce dernier, car plus abstrait.



Outils :



Le package « objets » contient uniquement les objets qui m'ont permis à stocker des donner.
Le Segment à dessiner dans le DrawPanel avec ses attributs (Color couleur, int epaisseur, Point p1, Point p2).
Le User (String nom, int nbPoints) implements Comparable pour trier facilement (avec Collection.sort) la liste des joueurs en fonction de leurs nombre de points.

La Commande (String nom, ArrayList<String> attributs) me permet de stocker une commande plus lisible qu'une ligne de String dans le code et plus facile à utiliser.

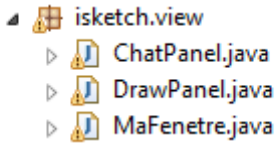
Les interfaces utilisées concernent uniquement le pattern Observer qui permet au client d'être informé à chaque fois que le client :

- rentre du texte dans le chat ;
- dessine un trait ;
- se connecte/déconnecte.

Le package « tools » contient une classe Tools regroupant uniquement des méthodes statiques, essentiellement pour la transformation d'une commande en format String vers le format Commande ou l'étape inverse.
EnumCommande sert simplement à utiliser un switch avec les noms des commandes (appuyé par une méthode dans la classe Tools)

Le package « test » a servi uniquement pour quelques premiers tests basiques principalement sur les méthodes de Tools. Les autres tests sont dans les classes testées elles même (exemple : les mains de MaFenetre, DrawPanel, ChatPanel et ClientThread qui ne consiste presque uniquement à créer l'objet représentant déjà un test assez complet)

Interface Graphique :

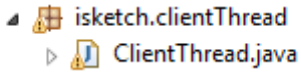


MaFenetre est la fenêtre qui s'ouvre lorsqu'on lance un client. En plus des deux grandes parties « ChatPanel » à gauche et « DrawPanel » à droite, elle a également une barre de menu en haut qui permet de se connecter/déconnecter du serveur et d'effacer son dessin (pour le moment fonctionnel qu'en local).

Le ChatPanel est formé de 3 parties, à gauche l'ensemble des messages envoyés par les joueurs, à droite la liste des joueurs avec leur nom et leur nombre de point (classé par ordre décroissant de points grâce à 'CompareTo'). Et enfin en bas, le champ de texte qui permet d'envoyer un message. L'envoie est gérée par le bouton 'envoyer ' ou la touche 'entrée' du clavier. La saisie est bloquée à 150 caractères.

Le DrawPanel est constitué de la zone de dessin en blanc et de sa barre d'outils en bas permettant de changer la couleur ou l'épaisseur du trait (initialisés à noir et à 10 pixels par défaut respectivement).

Client et Socket :

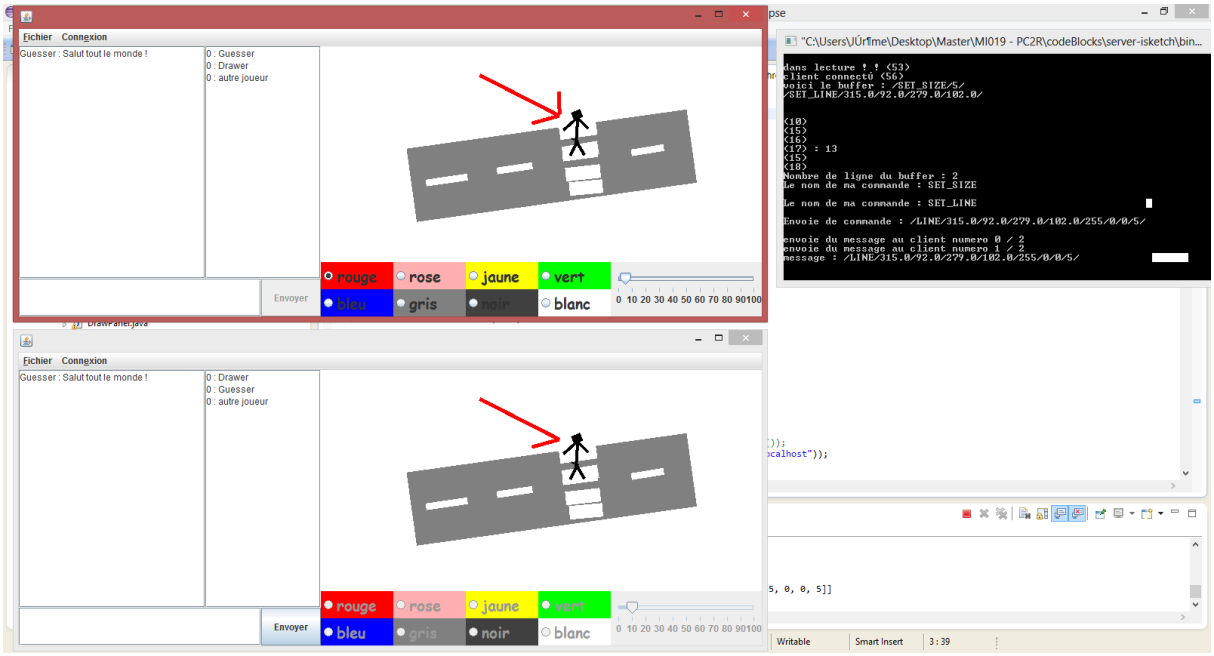


Gére la réception et l'envoie de commande.

L'envoie de commande se fait à chaque fois qu'un observateur effectue un 'update(commandes)', la commande est construite dans chacun des panneaux de l'interface graphique.

La réception se fait dans le thread du client (la lecture étant bloquante), elle appelle une méthode qui gère toutes les commandes reçues par un switch, qui appelle les composants graphique pour les mettre à jour (le client ayant la MaFenetre en attribut).

Exemple d'une partie :



(La capture d'écran est disponible en plus grand format dans le répertoire du projet)

Mode Emplois :

Il est possible de lancer les clients et le serveur dans l'ordre que vous voulez car les clients ne se connectent pas dès leur lancement. Il est quand même plus logique de lancer d'abord le serveur.

Lancement du serveur :
./bin/Release/server-isketch

Lancement des clients :
L'exécution de la fonction main contenu dans ClientThread ouvre automatiquement deux fenêtres de client.

2 fenêtres s'ouvrent. Les instructions suivantes sont à réaliser pour chaque client.

- Cliquez sur connexion/se connecter dans la barre de menu en haut à gauche.
- Choisissez un pseudo. (Évitez pseudo vide, pseudo avec « / », « \n » ou tout autres caractères spéciaux, préférez les lettres de l'alphabet, les chiffres et/ou espaces)
- Attendez que chaque client se connecte.
- Vous êtes devineur : seul le chat est accessible, le champ de texte en bas à gauche vous permet de discuter avec les autres joueurs.
- Vous être dessinateur : seul le panneau de dessin est accessible, choisissez la couleur et l'épaisseur de votre trait (avec les combo-box et le slider) et cliquez sur le panneau de dessin tout en restant appuyé jusqu'à une seconde destination pour dessiner un trait.
- Maintenant à vous de jouer !

Limite du projet

Etant seul, j'ai vraiment manqué de temps pour les finissions. Finissions qui se sont retrouvées cruciales au final… En effet, j'ai développé mon application sur windows 8 (l'installation de Linux en dual boot étant pratiquement impossible) et au moment de corriger les soucis pour rendre le serveur (en C) compilable sur Linux, j'ai eu des erreurs de compilations incompréhensibles que je n'ai pu résoudre.

L'échange de message par le chat et de dessin par le panel de dessin fonctionne correctement mais la simulation d'une partie complète n'est pas terminée (les méthodes sont codées dans « game.c » mais elles ne sont pas terminées ni testé)