RAPPORT DE PROJET – BTS SNIR 2

PROJET PROGRAMMATION INFORMATIQUE « PONG »

Projet réalisé par

Julien HUYGHE
Nicolas JOUSSEAUME
Lucas LEBOURHIS

Projet encadré par

Sébastien ANGIBAUD

SOMMAIRE

I. INTRODUCTION

Dans le cadre de notre seconde année du Brevet Technicien Supérieur Systèmes numériques, il nous est proposé un projet de 12h nous permettant de découvrir les différents principes de fonctionnement d'applications communicante via les couches de transport UDP et TCP.

Ayant une passion commune pour les jeux-vidéos ayant marqués l'histoire par leurs avancées, notre groupe composé de Julien HUYGHE, Nicolas JOUSSEAUME et Lucas LEBOURHIS, a saisi l'opportunité d'exploiter cet intérêt commun pour proposer un projet au responsable M. Sébastien ANGIBAUD.

II. OBJECTIFS TECHNIQUES

Etant, pour le moment, dans un projet scolaire limité en temps, nous avons décidé de restreindre notre projet informatique en sélectionnant les éléments essentielles constituant le PONG.
Nous avons donc, dans un premier temps, choisi de mettre en place un serveur et un client TCP afin de connecter à distance plusieurs périphériques ensemble et dans un second temps l'interface de jeu du PONG.
Le but du projet est donc de permettre à deux joueurs de jouer dans une partie commune et d'afficher en temps réel, les différentes actions de chacun des joueurs sur l'écran de l'autre.

III. GESTION DU PROJET

Nous avons choisi de séparer ce projet en trois taches distinctes :

- Julien HUYGHE : gestion de la programmation coté client
- Lucas LEBOURHIS : gestion de la programmation coté serveur
- Nicolas JOUSSEAUM : gestion de l'IHM et du fonctionnement du jeu

Deux personnes se sont occupées de la communication entre le serveur et le client puisque, étant une notion nouvelle, cela semblait préférable de s'organiser de cette manière pour découvrir cette notion. De plus, ayant acquis de l'expérience en programmation l'année dernière, une simple personne sur la gestion de l'IHM était plus pertinent.

Pendant que Nicolas JOUSSEAUME commençait à réaliser son application fenêtrée, Lucas LEBOURHIS et Julien HUYGHE faisions la connexion entre le client et le serveur. Notre but était de réaliser en premier lieu la connexion entre les deux périphériques et d'ajouter petit à petit les éléments du programme de Nicolas pour pouvoir régler les problèmes que nous pourrions rencontrer pendant que Nicolas avance de nouveau sur sa partie.

IV. REALISATION DU PROJET

1. COMMUNICATION SERVEUR-CLIENT

Grâce aux annexes mise à notre disposition au début du projet, il a été très facile de faire se connecter le client au serveur.

A. Création des classes « Serveur » et « ClientTCP » et utilisation

```h
#ifndef SERVEUR_H
#define SERVEUR_H
#include <QtNetwork>
#include <QObject>



class MainWindow;

class Serveur : public QObject
{

Q_OBJECT

private slots:
    void sessionOuverte();
    void connexionClient();
    void RecevoirInfoBarreJ2();

private:

    QTcpServer *m_tcp_server;
    QTcpSocket *m_socket_client;
    QNetworkSession *m_network_session;
    quint16 m_blockSize;
    MainWindow *m_w;

public:
    int m_infoRecu;
    Serveur(MainWindow * w);
    void EnvoyerInfoBarre(int pos_Y);

};

#endif
```

```h
#ifndef CLIENTTCP_H
#define CLIENTTCP_H
#include <QtNetwork>
#include <QObject>

class MainWindow;

class ClientTCP : public QObject
{

Q_OBJECT

private slots:

    void RecevoirInfo();
    void afficherErreur( QAbstractSocket::SocketError socketError);

private :
    QTcpSocket *m_tcpSocket;
    quint16 m_blockSize;
    QNetworkSession *m_networkSession;
    MainWindow *m_w;

public:
    int m_info;
    ClientTCP(MainWindow * w);
    void ChangerInfo(int changement_pos_Y);
    void EnvoyerInfoBarre(int pos_Y);


};

#endif
```

Comment vous pouvez l'apercevoir, la création des classes sont approximativement les mêmes à l'exception de certaines procédures ayant un rôle spécifique en fonction de leur place (coté client ou cote serveur).
Vous pouvez notez la ressemblance existante entre les deux rectangles. Ce sont ces quelques lignes qui permettent la communication entre le serveur et le client :

- QTcpServer : rend possible les connexions entrantes TCP. Soit, nous saisons le port par lequel la communication sera effectuée, soit le serveur en prendra un automatiquement.
- QTcpSocket : protocole de transport fiable adapté pour la transmission continue de données.
- QNetworkSession : permet de contrôler les interfaces réseaux du système
- Serveur(MainWindow * w) : on a une agrégation de serveur à MainWindow. C'est-à-dire que le serveur (et le client car il a aussi cette ligne) dépend de la fenêtre crée.

2. INTERFACE HOMME-MACHINNE

3. FONCTIONNEMENT

A. REFLEXION

On s'est demandé, comment faire pour pouvoir faire transiter les informations de la barre du joueur 1 sur l'écran du joueur 2 ? avec le joueur 1 jouant sur le serveur et le joueur 2 sur le client et venant se connecter sur le serveur.
Après de nombreuses réflexions, nous sommes arrivés à la conclusion qu'il fallait faire en sorte que chaque joueur puisse bouger leur propre barre et envoyer en temps réel les coordonnées seulement lors de son déplacement.
L'idée était bonne, cependant seul le joueur serveur pouvait bouger sa barre. Nous avons donc décidé que seul le serveur exécutera les calculs que ce soit pour le joueur client que pour le joueur serveur.

B. CODE PAS A PAS

Tout d'abord, on se place dans les fonctions permettant de faire bouger les barres des joueurs car s'est à ce moment-là que nous devons envoyer l'information au joueur adverse que le joueur a bien bougé sa barre.

```
serv.EnvoyerInfoBarre(getCoordoneesBarreJ1()*(-1));
Client.EnvoyerInfoBarre(getCoordoneesBarreJ2());
```

Puisque la barre se déplacement seulement suivant un axe, on à juste à envoyer l'évolution de sa coordonnée à son adversaire. L'apparition d'une multiplication pas -1 à son importance. Cela va permettre au client de savoir si la position qui lui est envoyé est la sien ou celle du serveur (avec la multiplication).

Lorsque le client va recevoir une information, il va regarder si la valeur reçue est positive pour ainsi appeler la méthode (1) ou positive pour appeler la méthode (2).

(1) `m_w->positionnerMaBarre(InfoRecu);`

(2) `m_w->positionnerBarreJoueurAdv(InfoRecu);`

Pourquoi avons-nous fait comme ça ? parce, dans notre programme, le serveur fait le calcul de la position de sa barre mais également de celle du client lorsque ce dernier lui dit s'il bouge sa barre par le biais de ce code :

```
if(e->key() == Qt::Key_S)
{
    if(coordoneesBarreJ2 < 525){
        DemandeDescendreBarre = true;
        if(DemandeDescendreBarre){
            x = 1;
            Client.EnvoyerInfoBarre(x); //1 = descendre barre
            x = 0;
        }

          descendreBarreJ2();
    }
}
```

Explication :

Si on appuie sur la touche assignée à l'action de faire descendre la barre alors on vérifie si sa position et si elle est correctement on passe la variable « DemanderDescendreBarre » à « true » ce qui permet de rentrer dans la conditionnelle suivante. Si « DemanderDescendreBarre » est « true » alors on envoie un « 1 » correspondant a l'action de descendre la barre.

La variable x est initialisée à 0 plus haut dans le programme.