

C++

POLYTECH SORBONNE

RAPPORT

Le monde d'après

SENNOUN Naël

RAHBI Aissam

CovidMen

Table des matières

1	Introduction	3
2	Description de l'appli	3
2.1	Installation	3
2.2	Lancer l'application	3
2.3	Règles du jeu	5
2.4	Particularité des types de Covidmons	7
3	Fiertés au niveau du code	7
4	Diagramme UML	8
5	Qu'a-t-on appris ?	9
6	Que peut-on améliorer ?	9

Table des figures

1	Méthode par défaut	4
2	Méthode par port précisé	4
3	Méthode en ligne	4
4	Choix dresseur	5
5	Choix covidmon	5
6	Combat d'arène	6
7	Deviendrez vous un winner?	6
8	communication dresseur	7
9	UML	8

1 Introduction

Dans le cadre du cours de Programmation orienté objet - C++, nous allons créer un programme afin d'appliquer les notions étudiées en cours et d'y ajouter une touche personnelle. Ce projet va nous permettre de découvrir les interfaces graphiques en C++, nous avons décidé de rajouter une partie en réseau pour pimenter le projet et nous donner quelque chose de nouveau à apprendre en C++, nous en sommes plutôt fiers. On utilisera la bibliothèque SFML, qui est une bibliothèque multimédia, elle nous permettra de gérer le son, l'interface graphique, et la partie réseau. À travers ce jeu à 2 joueurs, nous allons établir une communication entre eux grâce à un serveur. Le code est découpé en 2 parties. Les fichiers `server.cc`, `serveur.hpp` et `main_serveur.cpp` gèrent les aspects serveur, récupération et retransmissions des données aux clients, le tout grâce à des paquets. La seconde partie regroupe les aspects clients, cette dernière s'occupe de la partie qui gère l'interface graphique, le son et de l'envoi des paquets des joueurs au serveur. Nous allons commencer par une brève introduction aux règles du jeu et à la programmation graphique avant d'expliquer le code.

2 Description de l'appli

2.1 Installation

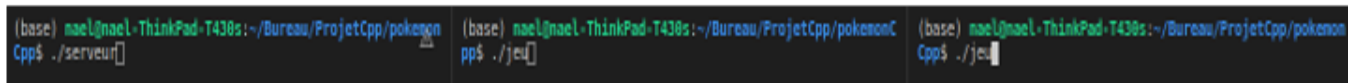
Pour jouer au jeu il vous faudra télécharger SFML, on peut le faire très facilement sur Linux :

```
sudo apt-get install libsFML-dev
```

2.2 Lancer l'application

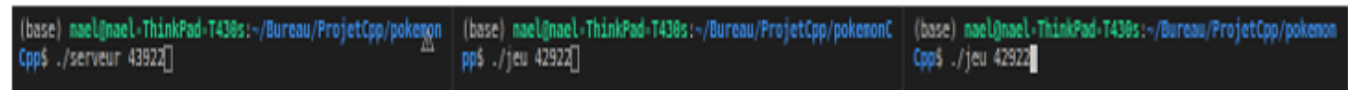
Il y a 3 manière de lancer le jeu :

- La méthode par défaut, vous serez connecté sur le port 30001 par défaut et en local.
- La méthode ou le port de connexion du serveur et du client est précisé en argument dans les fichiers main.
- La méthode en ligne, on précise le port de connexion pour le serveur, mais aussi l'adresse IP de celui qui lance le serveur, (Pour cette méthode il faut au préalable activer le port souhaité (Network Address Translation) avec votre box internet et préciser le protocole utilisé, on a choisi d'utiliser le protocole TCP, donc on précise TCP.



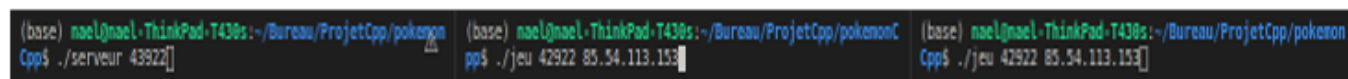
```
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
Cpp$ ./serveur  
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
pp$ ./jeu  
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
Cpp$ ./jeu
```

FIGURE 1 – Méthode par défaut



```
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
Cpp$ ./serveur 43922  
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
pp$ ./jeu 42922  
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
Cpp$ ./jeu 42922
```

FIGURE 2 – Méthode par port précisé



```
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
Cpp$ ./serveur 43922  
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
pp$ ./jeu 42922 85.34.113.153  
(base) nael@nael-ThinkPad-T430s:~/Bureau/ProjetCpp/pokemonC  
Cpp$ ./jeu 42922 85.34.113.153
```

FIGURE 3 – Méthode en ligne

Pour connaître son IP public il existe des sites, en voici un :
<https://www.whatismyip.com/what-is-my-public-ip-address>

2.3 Règles du jeu

Lancez le jeu et choisissez un personnage avec votre souris

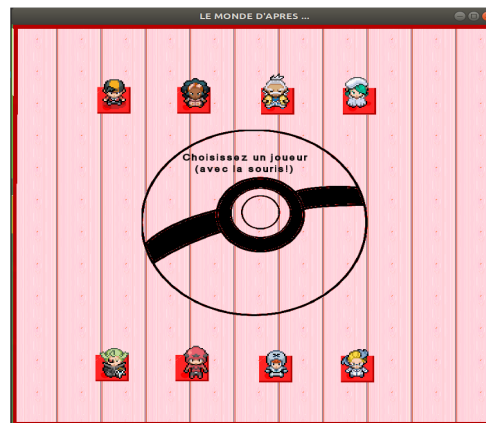


FIGURE 4 – Choix dresseur

Maintenant que le dresseur est choisi vous êtes connecté au serveur et vous pouvez vous déplacer à l'aide des flèches du clavier dans la seconde fenêtre. Vous choisissez ensuite l'un des 16 Covidmons présents. Vous pouvez voir votre adversaire se déplacer s'il est connecté.



FIGURE 5 – Choix covidmon

Une fois le Covidmon choisi, vous pouvez entrer dans l'arène.
Combattez !

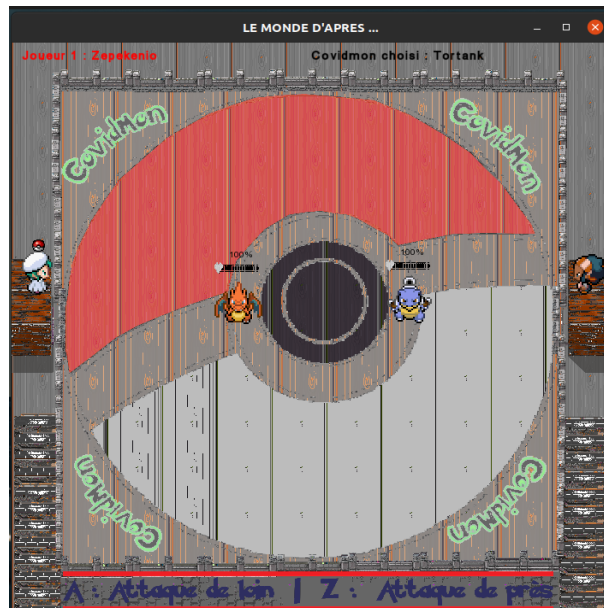


FIGURE 6 – Combat d'arène

Les personnages disposent de deux manières d'attaquer : une attaque de loin (touche A) et une attaque de près (touche Z). La règle est simple, le premier qui meurt à perdu.



FIGURE 7 – Deviendrez vous un winner ?

2.4 Particularité des types de Covidmons

Les covidmons peuvent attaquer, de loin ou de près, les attaques de loin font moins de dégâts que celles de près.

Il y a 4 types de Covidmons Feu/Eau/Plante/Vol, un classique pour l'instant, on n'innove pas. Chaque type de Covidmon possède au moins un point faible et/ou point fort contre un autre type de Covidmon.

Si le type d'un Covidmon est fort contre son adversaire, on appliquera un coefficient multiplicateur de 1.25 sur les dégâts de l'attaque lancée, en revanche si le Covidmon est faible contre son adversaire, le coefficient sera de 0.75.

- Type vent : 500 Pv, efficace contre Eau et Plante (Spécificité : plus rapide que les autres). La couleur des attaques des covidmons de type Vole est blanche.
- Type Feu : 650 Pv, efficace contre Plante. La couleur des attaques des covidmons de type Feu est rouge orangée.
- Type Eau : 800 Pv, efficace contre Feu. La couleur des attaques des covidmons de type Eau est bleu.
- Type Plante : 800 Pv, efficace contre Eau. La couleur des attaques des covidmons de type Plante est verte.

3 Fiertés au niveau du code

Tout est expliqué dans les commentaires, on en est fier car c'est la première fonction qu'on a utilisée pour observer le déplacement des dresseurs en réseau.

```
void Serveur::communication_dresseur(std::size_t i)
{
    // Paquet qui va servir a savoir si le données concerne le dresseur ou le covidmon
    sf::Packet receivePacket_type;
    // Les infos recues
    sf::Packet receivePacket_data;
    // Si une paquet est reçu on entre dans le if
    if (this->Clients[i]->receive(receivePacket_type) == sf::Socket::Done)
    {
        this->Clients[i]->receive(receivePacket_data);
        receivePacket_type >> this->_type;
        if (this->_type == "dresseur")
        {
            // On a chargé les informations dans cette ordre avec le client donc on le recupere dans cet ordre
            receivePacket_data >> this->_nom >> this->_dir >> this->_animation >> this->_x >> this->_y >> this->_bg;
            sf::Packet sendPacket;
            sendPacket << this->_nom << this->_dir << this->_animation << this->_x << this->_y << this->_bg;
            // On envoie le paquet au client qui n'a pas envoyer le paquet
            for (std::size_t j = 0; j < this->Clients.size(); j++)
            {
                if (i != j)
                {
                    this->Clients[j]->send(sendPacket);
                }
            }
        }
    }
}
```

FIGURE 8 – communication dresseur

5 Qu'a-t-on appris ?

Même si on ne l'a pas mentionné précédemment, on a appris à utiliser le logiciel Gimp, car bien entendu, on a designé les décors et les attaques des covidmons (Par contre on a trouvé les sprites de nos covidmons et personnages sur Internet). On a appris à utiliser la partie audio, graphique et network de SFML. De plus, on appris beaucoup de choses théoriques sur la programmation en réseau.

6 Que peut-on améliorer ?

Il est important de connaître ses faiblesses pour mieux progresser, c'est pourquoi il est important se questionner sur ce qu'on aurait pu faire de mieux.

Il est flagrant que dans notre jeu si quelqu'un a un trop petit pc et que la fenêtre 700x700 pixels est trop grande pour lui alors il ne pourra pas bien voir ce qu'il fait dans le jeu car l'écran sera coupé, c'est ce qu'on a remarqué à la fin de notre projet quand on a voulu tester notre jeu avec des amis. Pour régler ce problème on aurait pu faire en sorte que nos fonctions de placement des dresseurs/covidmons et la scale du sprite de nos images soit faite en fonction de la taille de l'écran.