



UNIVERSITÉ DE NANTES

Master 1 ALMA

Projet de Réseaux et Protocoles Internet

Bomber Unicorn

Étudiants :

Coraline MARIE et Vincent RAVENEAU

Intervenant :

Pierrick PASSARD

21 mars 2014

Table des matières

Introduction	2
Conception	3
Dépendances externes	3
Fonctionnalités du logiciel	3
Interface graphique	3
Développement	5
Serveur	5
Client	5
Déroulement d'une partie	5
Structure du code	6
Conclusion	7

Introduction

Dans un monde où les connexions sont de plus en plus présentes, il est indispensable, en tant que futurs ingénieurs en informatique, de connaître et de comprendre les différentes structures de réseaux existantes sur lesquelles notre monde évolue.

Dans le cadre du cours *Réseaux et Protocoles Internet*, nous avons pour objectif la création d'une application réseau de type *client/serveur*. Pour ce faire, nous avons recréé le jeu multijoueur de BomberMan avec de nouveaux graphismes : **Bomber Unicorn**.

Ce rapport présente donc la création de **Bomber Unicorn** au travers de nombreuses étapes de travail, ainsi que les différentes fonctionnalités de ce jeu. Pour cela nous verrons d'une part la conception de l'application, et le développement d'autre part.

Conception

Dépendances externes

Avant même de concevoir un logiciel, il faut au préalable le penser, ainsi que définir ses besoins et ses différentes fonctionnalités. Dans cette optique, notre cahier des charges nous imposait de concevoir notre application en **langage C**. Ce langage ne fût pas choisi pour ses performances dans la conception d'applications réseau, mais plutôt dans une optique pédagogique. En effet, il est moins évident de programmer des échanges entre clients et serveur en **langage C** plutôt qu'en **Java** ou en **C++**, où tout est mis à disposition pour aider le programmeur.

En ce qui concerne la partie graphique de l'application, nous avons choisi d'utiliser le **CSFML** qui est le binding officiel de la **SFML** pour le langage C. La **SFML** étant une interface de programmation destinée à construire des jeux vidéo ou des programmes interactifs en **C++**.

Fonctionnalités du logiciel

Pour ce projet, aucune fonctionnalité ne nous a été imposée, excepté qu'il fallait créer une application de type *client/serveur*. Pour ce faire, nous avons décidé d'utiliser le serveur en tant que moteur du jeu et les clients en tant que joueurs. Le serveur est capable de gérer jusqu'à 100 parties en simultané (si la machine qui le l'héberge est assez puissante), et chaque partie comprend de 2 à 4 joueurs. Pour des raisons de simplicité, de performance et de fluidité, une partie se déroule en tour par tour (un client doit attendre son tour pour jouer).

Les règles du jeu sont simple, pour gagner, un joueur doit éliminer tous les autres en réduisant leur point de vie à 0. Pour ce faire, chaque joueur dispose au départ de 3 points de vie, et peut poser jusqu'à 3 bombes en même temps sur la carte. Lorsqu'une bombe est posée, elle change d'état progressivement jusqu'à exploser. Un joueur peut être blessé par ces propres bombes, et lorsque l'une d'elle explose, les joueurs présent autour de la bombe sont également touchés et perdent un point de vie.

Interface graphique

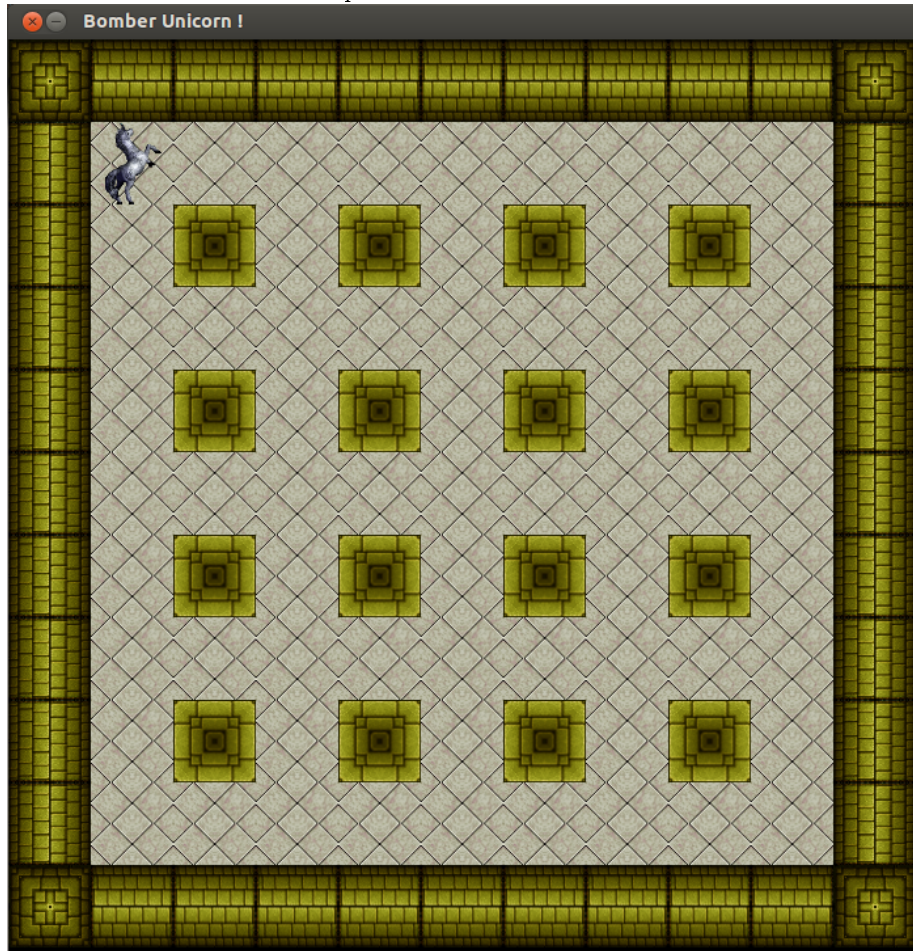
En ce qui concerne l'interface graphique, nous l'avons entièrement construite à l'aide de la **CSFML**. Il s'agit d'un ensemble d'éléments graphiques, appelés *sprite*

que l'on place dans une fenêtre. Nous avons créé l'intégralité de nos sprites nous même à l'aide d'une base de donnée libre d'images et d'un logiciel infographique.

Voici quelques exemples de sprites :



Voici la carte créée avec les sprites :



Développement

Les clients et le serveur communiquent en s'envoyant divers messages. Ces messages concernent l'état du jeu ou l'état des joueurs (position, points de vie, nombre de bombes, tour de jeu...). Ces messages sont sous la forme de chaînes de caractères et agissent sur les différentes variables du jeu.

Serveur

Lorsque le serveur est lancé, il reçoit en paramètre le nombre de parties à gérer, et le nombre de joueurs nécessaire par partie.

```
[[[[[[[Partie du rapport à faire par toi si possible]]]]]]]
```

Client

Avant de lancer les divers clients, il faut s'assurer que le serveur tourne déjà. Lorsque un client est exécuté, il doit avoir en paramètres : une chaîne de caractères (le nom du joueur) et une adresse (l'adresse ip du serveur). Grâce à ces paramètres, le client demande une connexion au serveur qui, s'il n'est pas surchargé, répond en lui renvoyant le numéro de la partie qui lui a été attribué et son numéro de joueur. Si le serveur est surchargé, il répond qu'aucune partie n'est disponible.

Déroulement d'une partie

Lorsque le client est connecté au serveur et qu'il possède un numéro de partie et un numéro de joueur, alors la partie peut commencer. Le joueur dispose de deux états, mis à jour par le serveur : l'un lorsque c'est à son tour de jouer, et l'autre quand ça ne l'est pas.

Si c'est à son tour de jouer, alors le client récupère une action de l'utilisateur et l'envoie au serveur. Le serveur peut alors lui renvoyer deux messages :

- Un message disant que l'action de l'utilisateur est refusée (déplacement incorrect...), dans ce cas le client récupère une nouvelle action de l'utilisateur, la renvoie au serveur et re-attend un nouveau message.
- Un message avec les nouvelles coordonnées des joueurs et des bombes, qui signifie que l'action a été acceptée, que le client doit mettre à jour l'interface graphique, et que ce n'est plus à son tour de jouer.

Si ce n'est pas à son tour de jouer, alors le client peut recevoir deux messages différents du serveur :

- Un message lui annonçant que tous les joueurs ont fini de jouer et que c'est à son tour.
- Un message avec les nouvelles coordonnées des joueurs et des bombes pour que le client mette à jour l'interface graphique.

Structure du code

Pour simplifier la création et la lecture du code du client, nous l'avons divisé en plusieurs parties :

- `client.c` : fichier principal appelant les différentes fonctions du client.
- `clientfunctions.c` : fichier secondaire contenant les différentes fonctions indispensables au bon fonctionnement du client.
- `graphic.c` : fichier contenant les différentes fonctions liées à l'affichage graphique gérée par le client.

Conclusion