Projet Réseau

Bomberman

Guillaume Magniadas

Contents

1 Introduction

1.1 Projet

Dans le cadre de notre cours de Réseau, nous devions former un groupe et choisir un projet qui serai principalement en réseau. Le groupe s'est formé assez naturellement en fonction de nos affinités, je me suis donc retrouvé avec Mehdi, Julien, Lièes et Etienne. Le choix du projet était totalement libre mais nous n'avons pas beaucoup hésité sur celui-ci car nous savions déjà à l'avance le projet que nous voulions faire : un Bomberman.

1.2 Bomberman

Bomberman est un jeu où l'on incarne un bonhomme qui pose des bombes, et notre but est de casser des blocs qui constituent la map, de survivre et de tuer ses adversaires pour être le dernier survivant.

1.3 Protocole

La première étape pour ce projet fut de rédiger un protocole, nous étions d'abord partis sur un protocole où tout était calculé côté client et où le serveur ne servait que de passerelle, puis il fut abandonné lors de l'arrivée de Lucas au sein du groupe, qui nous fit changer le protocole pour plus le penser côté serveur, donc dans notre protocole final, tout calcul est effectué côté serveur, et les clients ne servent qu'à l'affichage.

Voici sa version final :

```
########################## Log in and out ################################

C -> S
LOG USERNAME

Note : L'odre n'est pas important, le serveur peut envoyer CND avant
meme que le client ne se log, donc toujours traiter la commande CND
peut importe si vous vous ete log ou pas

S -> C
CND NUM
ERR LOG
ERR FUL
```

Envoie une liste a un client qui se connecte contenant le nombre de membres deja connecte ainsi que leur numero et leur pseudo correspondant.
Note : La liste ne contient pas le client qui vient de se connecter (Cf note ARV)
S –> C
LST NUMBER NUM USERNAME NUM USERNAME...

Note : Un client qui vient de se connecter recoit LST et ARV (La liste contenant les clients deja connecte et ARV contenant lui meme)
S –> A
ARV NUM USERNAME

S –> A
LFT NUM

Pas obligatoire
C –> S
QUT

//Debut Pas Obligatoire !!
C –> S
MSG (un message textuelle)

S –> A
MSG NUM (Un message textuelle)

Note : l'hote est la personne qui gere le serveur, normalement elle est defini par le fait qu'il sagit de la premiere personne connecte, donc le numero 0, a voir apres si on rajoute des commandes pour prevenir le changement d'hote.

########## MAP SELECTION ##########

Ces commandes permettent de changer la map du serveur sans avoir a connaitre les map a l'avance
Commande pour changer la map active du serveur, par defaut a 0.

Hote -> S
NXT
PRV

Le serveur renvoie la map active en cas de changement de map
(a tout le monde, pour qu'ils puissent suivre ce qu'il se passe)
S -> A
ACT Num "Nom de la map"

##################### CONFIG ############################

S -> C (Une fois log)
S -> A (Lors d'un changement)
CFG BOMB POWER SPEED BMB-DUR VIE %ALL BMB POW SPD
Les options doivent pouvoir fonctionner meme si cette commande
n'est pas recu.

Commande pour choisir les option de partie :
    SET BOMB POWER SPEED DureeBombe VIE

BOMB = Numbre de bombes en debut de partie. Minimum 1.
POWER = Puissance en debut de partie. Minimum 1.
SPEED = vitesse en debut de partie (un chiffre entre 0 et 9,
0 etant le plus lent, 9 le plus rapide)
DureeBombe = La duree des bombes avant d'exploser en millisecondes
VIE = Nombre de coup subis permis avant de mourir. Minimum 1

Commande concernant les chances
CHA %All Bmb Pow Spd

%All = %de chance qu'un bonus aparraisse, valeur entre 0 et 100

La somme des 3 dernier paramettre vaut le total

Bmb = chance qu'un bonus de bombe apparaisse sur total
Pow = chance qu'un bonus d'explosion apparaisse sur total
Spd = chance qu'un bonus de speed apparaisse sur total

exemple : CHA 60 5 5 10
il y a 6 chance sur 10 qu'un bonus spawn

5 + 5 + 10 = 20
donc si un bonus apparait il a 5 chance sur 20 d'etre un bonus de bombe,
5 sur 20 un bonus d'explosion et 10 sur 20 d'etre un bonus de speed.

//Fin Pas Obligatoire !!

Hote -> S
RDY

########## Debut de partie ##########

Quand tout le monde est connecte le serveur envoie aux clients la map et
la position de chaque joueur. Chaque client previent quand il est pret
puis le server envoie un signal de depart.

S -> A
MAP HEIGHT WIDTH DATA
0 VIDE
1 DESTRUCTIBLE
2 INDEST

Envoie a tout le monde le nombre de joueur ainsi que leur numero, leur
pos sur la map et leur pseudo

S -> A
ALL NUMBER NUM1 X Y NUM2 X Y NUM3 X Y...

S -> A
STR SLEEP

########## Pendant la partie ##########

Le client envoie les inputs clavier, deplacement et bombes. Le serveur
verifie que les actions sont possible et renvoie les positions des
joueurs et des bombe dans ce cas.
C'est le serveur qui annonce l'explosion des bombes.

C -> S
MOV L //Gauche
MOV R //Droite

MOV B //Bas
MOV T //Haut

S -> A
POS NUM X Y

C -> S
BMB

S -> A
BMB X Y POW

S -> A
EXP X Y POW

Informe les clients des bloques casse par les bombes qui ont explose
S -> A
BRK X1 Y1 X2 Y2....

Le serveur decide des bonus qui apparaissent sur la map suite a
l'explosion de certains bloc, il envoie leur position et type,
si un joueur passe sur un bonus, le serveur l'annonce aux autres.

S -> A
BNS X Y TYPE
0 NB BMB
1 POW
2 SPEED

S -> A
GOT X Y

Le serveur annonce les morts, voire la fin de la partie.
S -> A
DTH NUM NUM
    Numero du joueur tue ainsi que du tueur

S -> A
END NUM

########################## Ended ############################

```
L'hote envoie au serveur qu'il veut recommencer
H -> S
RST

Puis le serveur l'envoie a tout le monde
S -> A
RST
```

Ce protocole final n'a pas été obtenu d'un coup, il a subit énormément de retouches avant d'atteindre sa forme final ci-dessus.

2 Développement

2.1 Déppart

Une fois le protocole établi, je me suis mis directement à développer le projet, mon langage choisi était le C++ avec la librairie SFML pour ce qui est de l'interface Graphique.

Je me suis directement lancé dans la rédaction du code, sans vraiment réfléchir à la structure finale du projet. J'ai commencé par le chargement d'une carte sous forme de data, comme entendu dans le protocole, j'ai fait en sorte de tout charger de manière optimisée en utilisant des vertexArray (outils fournis par SFML), j'ai ensuite fait un menu à choix multiple pour choisir le mode (client ou serveur) puis j'ai commencé à écrire le serveur.

2.2 Début de serveur

Le serveur est la première vraie partie du jeu que j'ai implémenté, car sans lui le client n'aurait pas vraiment eu de sens, je me suis donc inspiré des serveurs créés en cours pour le mettre en place, et j'ai fait une classe protocoleServeur qui est la classe qui s'occupe de traiter chaque commande reçue (Note : Le serveur ignore les commandes envoyées avec un intervalle beaucoup trop faible, pour éviter d'être submergé). Le serveur est découpé en 3 phases, la phase de lobby, la phase de jeu, et la phase de fin de partie.

2.3 Début de client

J'ai, en parallèle du serveur, commencé à implémenter le client, lui aussi dispose d'une classe propre, protocole, qui s'occupe de gérer les commandes. L'avantage de développer les deux de façon parallèle est de pouvoir tester chaque ajout des deux cotés, et donc de ne pas avoir à revoir du code trop ancien, quand on se rend compte qu'il ne fonctionne pas forcément comme souhaité.

2.4 Avancement

J'ai donc travaillé à un rythme assez régulié, en rajoutant des fonctionnalités petit à petit, tel que les déplacements, les collisions, la pose de bombe (sans explosion encore). Puis vient l'ajout de l'explosion des bombes. Les calculs étant gérés côté serveur, c'est à celui-ci que revient la tâche de s'occuper de l'explosion des bombes. Chaque bombe posée doit posséder un timer interne la faisant ensuite exploser, pour cela j'ai opté pour des threads avec des sleep des header chrono et thread. Donc, quand un joueur pose une bombe, le serveur crée un thread qui dort 2 secondes puis qui s'occupe de la tâche d'explosion de la bombe. Ensuite, un petit algo assez simple s'occupe de la détection de mur cassable/autre bombe/joueur sur son passage (Note : Un thread doit quand même vérifier s'il a toujours une bombe posée sur sa case et s'il s'agit bien de la même qu'avant son repos, au cas où une autre bombe ne l'aurait pas explosé). Côté client, je ne fais que faire disparaître les bombes et les blocs signalés par le serveur, et je rajoute un effet de flammes sur la portée de la bombe, elle aussi gérée par un thread qui dort une seconde puis qui s'occupe de supprimer le vertexArray des flammes. (Note : ce thread m'avais posé des soucis, c'était à cause de lui que j'obtenais un rare segfault, en effet, il arrivait qu'il y ait un conlit entre ce thread et le thread principal, car il pouvait des fois supprimer l'arrayBuffer en même temps que le thread principal le dessinait. C'est après plusieurs réflexions que j'ai découvert qu'il était la source de ce segfault, et j'ai donc rajouté un mutex pour l'empêcher de supprimer pendant un dessin des flammes.). Après tout ces ajouts, le jeu commencait à être bien jouable, il ne manquait que quelques fonctionnalités.

2.5 Finalisation

Pour rajouter du fun au jeu, nous avons rajouté un chat au Lobby, pour pouvoir discuter de nos victoires, passé ou futur entre plusieurs parties.

Cet ajout fut très simple, vu qu'il ne s'agit que d'envoi de messages. Ensuite, un ajout simple mais primordial, les Bonus ! J'ai rajouté les bonus d'augmentation de nombre de bombes, de puissance et de vitesse, apparaissant de manière aléatoire après l'explosion de murs. Après ça, j'ai surtout travaillé la présentation des menus, pour avoir le projet le plus soigné possible, une bonne correspondance entre les menus, pouvoir revenir au début facilement sans avoir à tout relancer pour changer de serveur, l'ajout d'une sélection de map simple ainsi que d'un lecteur de musique aléatoire simple également, l'intérêt ici est aussi de rendre le jeu modulable pour l'utilisateur, il a juste besoin de faire glisser une nouvelle musique dans le dossier music/game pour la rajouter au jeu, et pareil pour la map : les map étant simple à comprendre, un utilisateur pourrait en rajouter facilement, et jouer avec n'importe quel autre client compatible sans avoir besoin de lui donner le fichier ! L'ajout de musiques sur les menus et lobby, de bruitages au jeu fut aussi une des dernières choses rajoutées pour rendre le jeu le plus fun et propre possible.

2.6 Ajout de dernière minute

Nous avons rajouter quelques commandes vers la fin pour rendre le jeu plus agréable pour l'utilisateur comme NXT, PRV et ACT qui permettent à l'hôte de choisir la map de facon simple, un RST qui permet de revenir au lobby en fin de partie pour pouvoir recommencer sans avoir à relancer le serveur, ainsi que des informations sur le tueur, pour savoir qui a tué qui, ainsi qu'une information sur le gagnant, pour savoir qui a gagné la partie. La dernière fonctionnalité que nous avons rajouté est le fait de pouvoir choisir ses options de jeu. On peut modifier le nombre de bombes, la taille d'explosion, la vitesse au départ d'une partie, le temps d'explosion d'une bombe, le nombre de vie). Et du coup j'ai mis un petit temps d'invincibilité si on est touché par une bombe et que ce n'est pas notre dernière vie), le pourcentage de change d'apparition d'un bonus, et le ratio de chacun des 3 bonus. Ces options sont découpées en 2 commandes, SET et CHA.

3 Explication Fonction

Dans cette section, je vais expliquer de manière brève le fonctionnement des fonctions principales.

3.1 Main

Le main est très classique, c'est lui le vrai Menu : il s'occupe de lancer le menu graphique, et en fonction de ses retours, s'occupe de lancer un serveur ou un client.

3.2 Menu

Fonction Graphique (toute les fonctions graphiques utilisent les méthodes SFML) pour afficher les modes (serveur ou client) ainsi que quelques informations telles que le Titre du jeu, l'ip Locale et mon nom.

3.3 Server

Fonction qui lance le serveur. (Après avoir ouvert une socket pour celui-ci au préalable.)

3.4 loby

Fonction Graphique qui lance un lobby. Dans celui-ci on peut communiquer avec les autres clients connectés, on peut décider de la map et du démarage de la partie si on est l'hôte.

3.5 Partie

Fonction Graphique qui lance la partie. Cette fonction est la fonction principale d'affichage du jeu, permettant de jouer.

4 Explication Classes

Dans cette section, je vais expliquer de manière brève le fonctionnement des Classes principales.

4.1 Protocole et ProtocoleServer

Ces deux classes servent à gérer les commandes reçues côté client et côté serveur. La classe ProtocoleServer est plus complète car elle gère aussi le déroulement de la partie.

4.2 Carte

Il s'agit de la classe principale d'une partie côté client. Elle gère l'affichage de la carte, des blocs, de la suppression de ces derniers, des bombes, de l'affichage des explosions et des bonus.

4.3 Log

Classe qui gère entièrement l'affichage du chat dans le lobby.

4.4 Player

Classe qui gère les joueurs, leur position et leur affichage s'ils sont en vie.

4.5 TempoMessage

Classe qui gère l'affiche de messages brefs au cours de la partie comme la mort d'un joueur ou le joueur victorieux en fin de partie.

4.6 SoundHandler

Classe statique qui permet de jouer n'importe quel son à n'importe quel moment (SFML gère la lecture audio) et gère aussi les lecteurs audio pour pouvoir lancer autant de sons que l'on veut avec de simples méthodes sans écraser d'autres sons.

4.7 OptionChoice

Classe qui sert à la gestion des options d'une partie pendant le lobby.

5 Manuel

Pour compiler le jeu :
-: Make
Pour lancer le jeu :
-: ./bombermanGm
Il est aussi possible de lancer le serveur seul :
-: ./bombermanGm "Numero de port"

Tout ce qu'il faut savoir pour jouer au jeu se trouve dans le fichier Manuel.txt, Le voici ;

Menu

Sur le menu, fleche du haut, fleche du bas pour naviguer dans le menu, entrer pour choisir (Z et S fonctionnent aussi)

Quand vous devez rentrer l'adresse ou le port, tapez le simplement puis faites entree.

Lobby

Sur le lobby, les lettres permettent d'ecrire des messages, et entrer permet de les envoyer.
Vous pouvez effacer le contenu actuel du chat
en cliquant sur Clear (Le bouton Bleu).
Si vous etes l'hote,
vous pouvez choisir la map avec les fleches gauche et droite.
vous pouvez changer les options de partie en cliquant sur les petits + et −.
vous devez lancer la partie quand vous le souhaitez en appuyant sur le bouton Start vert en haut a droite, ou en faisant la fleche vers le haut si vous ne voulez pas utiliser votre souris ;).

En partie

Vous etes toujours le personnage Bleu.

En partie, il faut utiliser ZQSD pour se deplacer et espace pour poser une bombe, il y a 3 bonus : vitesse, plus de puissance et plus de bombes. Ils sont assez explicites :)

Une partie se termine quand il ne reste plus qu'un joueur, si vous etes l'hote, appuyer sur entrer pour envoyer l'ordre de retour vers le lobby.

Note

Apuyer sur la touche echap vous fera toujours revenir sur le menu, et si vous etes deja sur celui-ci, cela fermera le jeu.

Map

Il est possible de rajouter des map au dossier map et de jouer dessus en partie.
Pour ce faire il suffit d'ecrire une map en s'inspirant des map deja faites. C'est-a-dire, indiquer la longueur et la hauteur en entete du fichier, puis ecrire des chiffres entre 0 et 2 inclus (0 = case Vide, 1 = Bloque cassable, 2 = bloque incassable) et ce (longueur * hauteur) fois. Le fichier doit avoir comme extension .txt et une fois ceci fait, rajouter un fichier avec .txp comme extention possedant les positions possibles de spawn pour les joueurs, c'est-a-dire des couples X et Y autant de fois qu'il y a de spawn. (Je conseille d'en mettre au moins 4) (Note : Le resultat est indetermine si vous selectionnez une map avec moins de joueurs que presents en partie..)

Musique

Il est possible de rajouter des musiques dans le dossier music/game, il suffit de glisser des fichiers audios dedans.
A noter que certains formats comme le mp3 ne sont pas supportes. (Je conseille le format .ogg)

Note : Certain des membres de mon groupes ont rendu les flammes mortelle. Personnellement je n'ai pas fait cet ajout car je trouvais que cela rendait le jeu trop mou.

Je rajoute qu'il est aussi possible de changer les options de partie avec des commandes dans le chat :

- /set [NbBombes] [TailleExplo] [Vitesse] [tempsExploBombe] [NbVie]
- /chance [ChanceBonus] [RatioBombes] [RatioExplo] [RatioSpeed]

Je les avait implémentées avant les boutons plus et moins, je les ai laissé dans le code car elles ne gènent en rien le déroulement du programme mais sont beaucoup moins pratiques que les boutons, donc ne sont pas très utiles.

6 Crédits

Comme pour le Manuel, les crédits se retrouvent dans le fichier Crédits.txt que voici :

—-

Bomberman de Guillaume Magniadas, dans le cadre d'un projet Réseau.

Développé par Guillaume Magniadas.

Tout le contenue utilisé est libre de droit (excepté lorsque précisé)

Font:

Avara : https://fontlibrary.org/fr/font/avara

Bruitages:
Son Loby : https://opengameart.org/content/ui-sound-effects-pack

Bombes explosion : domaine publique, pas de liens

Musiques:

Menu = https://opengameart.org/content/grassy-world-overture-8bitorchestral

Loby = https://opengameart.org/content/waiting-iii

Musique en jeu : Circlerun : https://opengameart.org/content/adventure-platform-menu-bgm-pack-i

asgore - /! non libre de droit /! : Undertale - Asgore Theme

Sprites:
Personnage /! trouvé sur google, aucune certitude sur les droits /!

Fond menu : https://imgur.com/gallery/VZ9H2

Bombe /! trouvé sur google /! (le fond du loby en découle)

—-

7 Mot de la fin

Ce projet fut très intéressant, déjà pour la liberté totale à son sujet, et car le jeu choisi (Bomberman) est un jeu que j'apprécie beaucoup, ce qui m'a donné envie de réaliser le projet le plus propre possible, un jeu auquel j'aurai moi-même envie de jouer. Mes objectifs principaux dans ce projet fut, au-delà de le terminer à temps, d'avoir un programme soigné et optimisé dans le but d'enrichir mes connaissances et ma maîtrise du C++. En effet, j'ai fait de mon mieux afin que le jeu prenne le moins de ressources possibles. A l'heure où j'écris ces lignes, le programme prend environ 16% d'un des coeurs de mon cpu (un vieil AMD 4 coeur cadancé à 1.9GHz) et environ 1% de ma mémoire vive (je possède 6 Go de ram). Bien que je ne possède pas beaucoup d'élément de comparaison, je trouve cette consommation raisonable.