Thomas Bouix EISE-4
Aziz Idomar

Time Bomb

Réalisation d'un jeu en réseau avec affichage graphique

Mme. Braunstein 2019-2020

Sommaire

Contexte

Le programme que nous développons ici est un mini-jeu en réseau avec interface graphique, reprenant le principe du *Time Bomb*. Cet exercice se déroule dans le cadre du cours de C++ de Mme Braunstein, pour les spécialités *MAIN* et *EISE* à l'école Polytech Sorbonne. Nous décriverons donc ici les principales fonctionnalités de notre application, en accord avec les contraintes indiquées dans le sujet.

Notice d'utilisation

Le jeu fonctionne sur le principe client-serveur. Il faut donc lancer un serveur, et que 4 clients s'y connecte (le jeu en graphique ne supporte que 4 joueurs).

Pour compiler tous les exécutables, aller dans le répertoire du jeu puis lancer la commande :
$*make*

Lancer un serveur en spécifiant son port :
$./timebomb_serv <port_number>

Lancer les différents clients, depuis votre machine ou une autre ayant un accès Internet :
$./timebomb_cli <server_ip> <server_port> <client_ip> <client_port> <player_name>

Règles du jeu

Un rôle caché est remis à chaque joueur. Chacun incarne soit un membre de l'équipe rouge, et doit faire exploser la bombe la trouvant parmis toutes les cartes, soit un membre de l'équipe bleue, qui doit désamorcer la bombe en trouvant tous les désamorceurs. Des cartes neutres (dites de sécurité) sont ajoutées afin de rendre plus difficile la recherche des autres.

Un joueur est désigné arbitrairement pour commencer. Il devra alors tirer une carte chez un des autres joueurs. Ensuite, ce sera au joueur chez qui on a tiré qui devra jouer à son tour, avec comme seule contrainte qu'il ne peut pas tirer chez le joueur précédent (ni lui même bien entendu).

Au bout d'un certain nombre de cartes tirées (fonction du nombre de joueur), on redistribue toutes les cartes encore en jeu, et le joueur chez qui on vient de tirer tire à son tour. La partie se termine quand la bombe est tirée par l'un des joueurs, ou quand tous les désamorceurs en jeu sont tirés.

Pour plus d'informations, consulter ce lien : https://www.regledujeu.fr/time-bomb/

Structure du programme

Notre jeu est composé de deux programmes, un **serveur** qui gère le moteur du jeu, et un **client** avec interface graphique qui s'y connecte.

Serveur

C'est dans le programme **timebomb_serv** que toute la structure du jeu est gérée. On y trouve le diagramme de classe suivant :

Une classe ***Player*** permet de représenter les différents joueurs dans une partie. Cette classe est mère de deux classes filles : ***Bot*** et ***Real***. La première est-elle même la classe mère de ***Strong*** et ***Weak***, qui permettent l'implémentation de joueurs gérés par la machine. Cette fonctionnalité n'a pas été exploitée ici mais on pourrait penser à implémenter une IA qui puisse faire jouer ces deux types de joueurs, et qui en plus permettrait des interactions dans le Chat de la partie. La classe *Real* est la classe utilisée lors de la connexion de vrais joueurs.

Les différentes cartes à joueur héritent toutes de la classe abstraite ***Card.*** Celle-ci est donne naissances aux classes ***Bomb***, ***Defuser*** et ***Safety***, les trois types de cartes du jeu.

Une super-classe ***Game*** permet de gérer la partie dans son ensemble. Son constructeur prend en argument un nombre de joueur souhaité pour une partie et commence ensuite l'initialisation de la partie, avec la distribution des cartes, des rôles, du ChatBox etc…

La classe ***ChatBox*** permet de gérer la connectivité des clients au serveur. C'est en fait l'interface entre le moteur du jeu et les clients qui s'y connectent. Elle traite dans un thread les messages arrivant par réseau, et permet également d'y répondre. Toutes les informations sur les clients sont stockés dans un vecteur d'objets ***Client***. Une fois tous les clients réunis, c'est cette classe qui communiquera à ***Game*** leurs noms, ip, etc…

Client

Le rôle du programme **timebomb_client** est de permettre la connexion d'un joueur avec le serveur principal, de pouvoir communiquer avec, et surtout d'afficher graphiquement l'état de la partie avec la bibliothèque SDL.

Protocole de communication

Nos deux types de programmes, *timebomb_serv* et *timebomb_client* communiquent à l'aide d'un protocole défini dans le fichier *communication_protocol.txt*. Ce fichier décrit les différentes sortes de messages que peuvent s'envoyer un client et le serveur. En effet, les actions graphiques exécutées sont communiquées au serveur via des envois de chaînes de caractères, et le serveur envoie des informations sur la partie pour que le client adapte son affichage.

Par exemple, un clic sur la deuxième carte du troisième joueur (indiquant un tirage) enverra le message suivant :
draw thomas 1
Ce à quoi le serveur pourra répondre :

```
R 2        // On joue le deuxième round
C 3        // 3 cartes ont été tirées dans ce round
D 4        // 4 defusers ont été tirés dans la partie
```

Respect des contraintes

Le sujet imposait l'utilisation d'au moins **8 classes** et **3 niveaux de hiérarchie**. Ces contraintes sont respectées : nous utilisons en effet au moins 12 classes, dont l'une d'entres-elles, *Player*, se décline deux fois, une classe fille *Bot*, elle-même mère des classes *Weak* et *Strong*.

Il fallait aussi avoir au moins **deux fonctions virtuelles**. Les nôtres sont les destructeurs ~Player() et ~Card() ainsi que la fonction Card::to_string(). Cette dernière permet de pouvoir utiliser la résolution dynamique de type, en appelant les fonctions to_string() sur des pointeurs de type *Card **. Quand aux destructeurs, il est important qu'ils soient virtuels car il est possible que l'on ai à *delete* un objet héritant de *Card* à partir d'un pointeur sur *Card*. Dans ce cas, si le destructeur n'est pas *virtual*, le comportement sera indéfini et l'on risque d'appeler le destructeur de *Card*.

Pour ce qui est des **conteneurs de la STL**, nous utilisons essentiellement les **vector** qui nous permettent de stocker des données, comme par exemple l'ensemble des cartes de la partie ou bien l'ensemble des joueurs connectés.

Fonctions remarquables

Une première fonction remarquable est la fonction **Game::play()**. Premièrement, le nom de l'ensemble {classe, fonction} rend admirablement bien. Ensuite, cette fonction utilise l'ensemble du code implémenté pour le moteur du jeu, et orchestre le déroulement de la partie en général. En effet, elle désigne le prochain joueur, distribue les cartes, autorise ou non les actions demandée par les joueurs, et communique des informations sur la partie en prenant en compte le destinataire. Elle est constituée d'une boucle while() qui ne se termine qu'à la fin de la partie et utilise un pointeur sur *ChatBox*, qui lui confère le pouvoir de communiquer en réseau.

Améliorations possibles

De nombreuses améliorations sont possibles pour notre jeu. Premièrement, l'affichage graphique n'a été implémenté que pour 4 joueurs pour l'instant, alors que le moteur du jeu autorise une partie allant jusqu'à 8 joueurs.
De plus, nous avons préparé l'ajout de joueurs gérés par le serveur, symbolisés par la classe *Bot*. Ces joueurs robots pourraient permettre de compléter une partie où il manque des joueurs. Il serait possible d'entraîner ces robots à parler dans le chat avec l'intelligence artificielle.
Enfin, il pourrait être intéressant de pouvoir rejouer plusieurs parties avec les mêmes joueur et de sauvegarder leurs scores dans un fichier extérieur au programme.