Rapport du *Timebomb* d'**IHM**, partie 2: **Interface.s & communications**

[par Kalumvuati Duramana, Duran Mike, Ledda Damien & Senol Mustafa]

Sommaire

Interface.s

L'interface utilisée sera graphique (plus précisément en Qt -- qui est, en soi, du C++ version UI), et le noyau logiciel (en C++). Les règles de base seront bien sûr implémentés (pour l'instant, il nous reste juste à savoir comment afficher les cartes des joueurs sans qu'il y ait de triche possible), mais si jamais on réussit à avoir une base complètement fonctionnelle (on y est presque à l'heure où ces lignes sont écrites), d'autres modes peuvent s'ajouter à celles-ci (dont le mode *More bombs!* cité dans la première partie du rapport).

Pour le moment, le noyau logiciel ne comporte que ce qui est nécessaire pour le fonctionnement du jeu (sous mode console, évidemment). C'est-à-dire:

- une classe **Player**, qui identifiera le.s joueur.s avec leur équipe associée (`getTeam`) et son nombre de cartes (`getNbCard`), s'il détient les ciseaux (`haveScissors`) ou les lui donne (`takeScissors`)/les prête au joueur coupé (`giveScissors`);
- une classe **Plateau** qui servira de table de jeu principale avec le nombre de joueurs (`getNbPlayer`), *shuffle shuffle* distribuant les cartes (`distribuer`) et gérant la jouabilité;
- une classe **Card** qui identifiera les cartes Rôle et Câble (`getType`), et vérifiant si elle a été retournée (`setStatus`);
- et une base donnant les valeurs globales utilisées dans le noyau logiciel.

Note: la liste ci-dessus ne contient qu'une liste non exhaustive des fonctions incluses dans les headers

Communication entre les modules

Pour faire fonctionner le projet, les interfaces sont notamment liées entre elles via divers boutons, à qui sont associées des fonctions spécifiques. En voici une liste non exhaustive à l'heure où ce rapport est écrit:

- `on_Exit_clicked`, disponible au menu, qui quitte le jeu avec `close()`;
- `on_Return_to_menu_clicked`, qui lie la plupart des interfaces dont les paramètres et la sélection du mode de jeu au menu, permettant de cacher l'interface actuelle pour montrer celle du menu via un pointeur qui lui est associé (de classe **MainWindow**);
- `on_Rules_b_clicked`, liant le menu aux règles, permettant d'afficher les règles par le biais d'un pointeur de classe **rules** après avoir caché le menu;
- `on_Settings_b_clicked`, qui en fait de même pour les paramètres (pointeur de classe **settings**)
- `on_Start_b_clicked`, qui commencera à initialiser le jeu;
- `on_previous_clicked`, ramenant de l'écran de l'insertion des pseudos vers le choix du nombre de joueurs par un pointeur de classe **gamemode**;

- `on_cut_wires_clicked`, qui est une des fonctions particulières en soi: non seulement elle fera les autres lors de son implémentation graphique (à l'heure où on écrit ce rapport, on l'a connecté au mode console via la fonction `startGame`, qui initialise le jeu en attribuant un rôle et 5 cartes à chaque joueur), mais avant ça, elle compte le nombre de joueurs dont le pseudo a été inséré (OK, d'une façon un peu piquante, on doit l'admettre);
- `on_radio_x[_y]_clicked`, disponible uniquement dans la sélection du nombre *maximal* de joueurs, mènera à l'écran des pseudos, mais avec un twist: le nombre maximal de joueurs sera transmis dans la classe **player_names** et dont la liste changera en fonction du nombre transmis via un `switch`. Les joueurs dont le nombre est supérieur au maximum sont donc cachés par `setVisibility(0)`.