IUT de Paris - Rives de Seine
Université Paris Cité

GP 101 & 102 : Félix Liburski – Hugo Pereira

Table des matières

Partie I : Présentation de l'application

I.1. Introduction

Le projet consiste à reproduire le jeu du *démineur* tout en l'adaptant aux attentes du sujet. Notre programme est développé en langage VB .NET. Nous avons utilisé l'IDE de *Visual Studio* pour développer ce programme. Ce dernier est encodé en UTF-8.

Notre programme est capable de traiter une partie complète d'une partie du *démineur* et gère la sauvegarde des parties. Pour exécuter notre programme, il suffit de lancer le fichier exécutable en plaçant le fichier texte de configuration (*config.txt)* dans le même répertoire. On note qu'on pourra choisir dans les paramètres du jeu de changer ou non le chemin d'accès à ce fichier.

Pour veiller au bon déroulé de ce projet, nous avons utilisé le logiciel GIT qui est un logiciel de gestion de versions. En effet, ce logiciel s'est avéré très utile lors de ce projet collaboratif plutôt long. Cela nous a permis, entre autres, de pouvoir faire des modifications simultanées, c'est-à-dire une possibilité de fusionner intelligemment notre travail. De plus, l'avantage de cette méthode est d'avoir le pouvoir de garder en mémoire toutes les anciennes versions (historique) mais aussi de savoir QUI a modifié QUOI et QUAND.

De plus, notre organisation a été complétée par l'utilisation de Discord pour échanger des messages, des fichiers ou même programmer des réunions hebdomadaires. Enfin nous avons utilisé Trello, qui sert à planifier des tâches dans le temps. Le respect de ces deadlines a été primordial pour une avancée satisfaisante au cours du temps.

Partie II : Structure de l'application

La structure a été déterminée en grande partie par le cahier des charges du sujet. Nous avons juste rajouté un formulaire pour le choix des thèmes car cela permet de pouvoir illustrer les thèmes avec des photos (captures d'écran de notre jeu).

II.1. Les formulaires

a. Accueil

Figure 1. *Formulaire d'accueil*

Ce formulaire est le premier aperçu que le joueur se fera du jeu. C'est pourquoi nous avons particulièrement travaillé le *design* de ce dernier. L'image de fond représente une bombe et toutes les fenêtres sont paramétrés avec un logo qui rappelle une explosion. Le formulaire répond bien aux attentes du cahier des charges. Nous avons juste rajouté une image pour le bouton qui permet d'accéder aux réglages. Enfin, nous avons mis un petit « i » (comme *information*) en haut à droite qui nous conduit vers une courte présentation de nous ainsi que de notre travail (figure 1).

b. Scores

Figure 2. *Joueurs classés par ordre croissant*

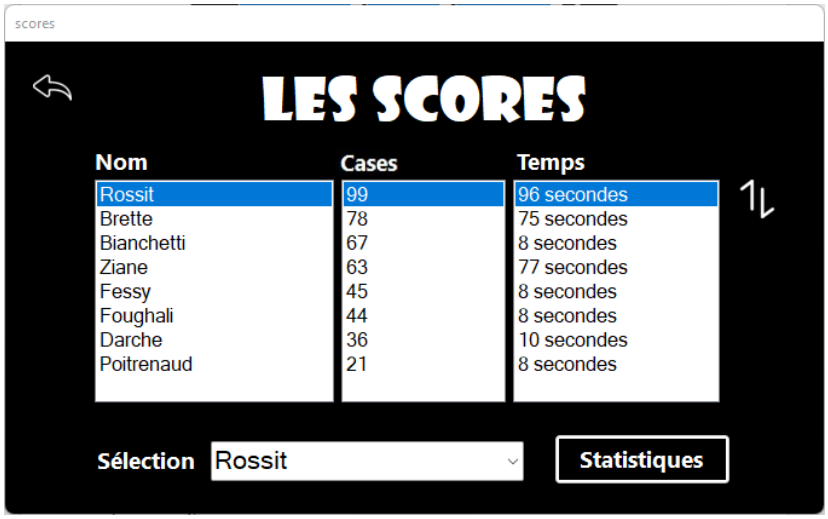
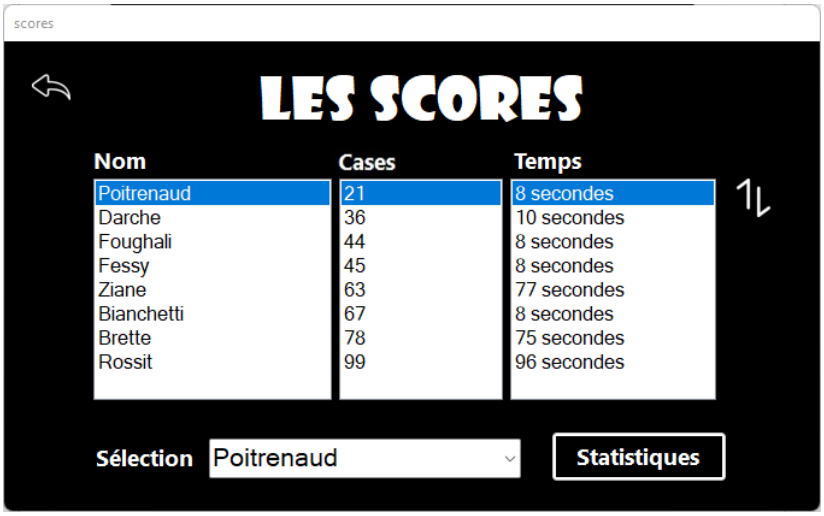


Figure 2bis. *Joueurs classés par ordre décroissant*



Ce formulaire n'a aucune option en dehors de celles comprises dans le sujet. Tous les joueurs enregistrés sont triés en fonction de leur meilleur nombre de cases découvertes. Les joueurs à égalités sont alors triés en fonction du temps. Nous avons également programmé un bouton qui permet d'afficher les joueurs par ordre croissant ou décroissant (figure 2). Nous avons dû créer une classe *JoueurComparer* qui permet de comparer deux joueurs. Cet objet va d'abord comparer le nombre de cases et en cas d'égalité, il va comparer les temps (comme indiqué dans le sujet). Nous utilisons des listes qui sont synchronisées entre elles afin de faciliter l'expérience de recherche de l'utilisateur.

c. Réglages

Ici nous avons essayé de laisser le plus possible le choix au joueur quant aux réglages de la partie. Pour chaque réglage nous avons utilisé l'outil le plus optimal selon nous. Par exemple pour le nombre de lignes et de colonnes la *scrollbar* nous a semblé être parfaite car elle permet de montrer clairement les limites de la grille. On note notre *checkbox* en bas du formulaire qui permet de désactiver ou non le minuteur en partie. Attention, si ce dernier est désactivé, les parties ne sont pas sauvegardé. Ce réglage est donc idéal pour les joueurs qui sont uniquement là pour s'amuser sans vraiment chercher la performance.

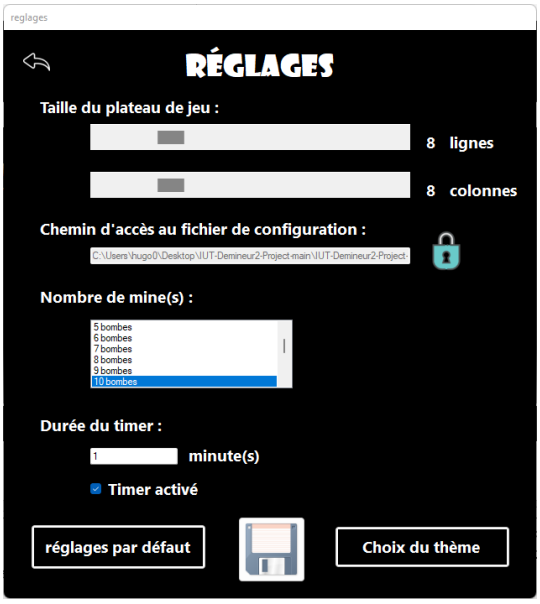
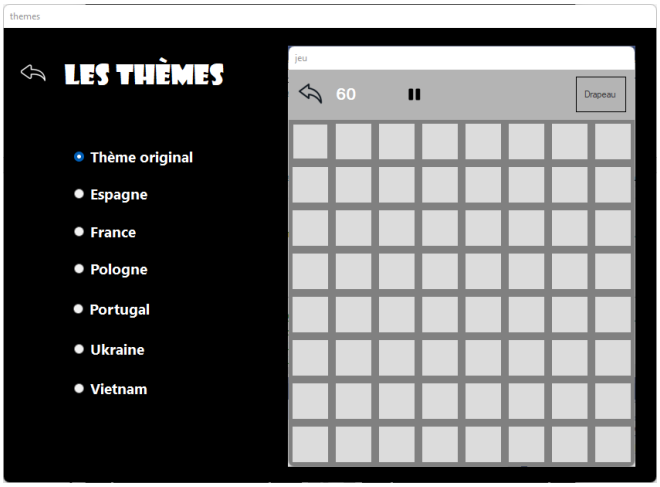


Figure 3. *Formulaire des réglages*

d. Thèmes



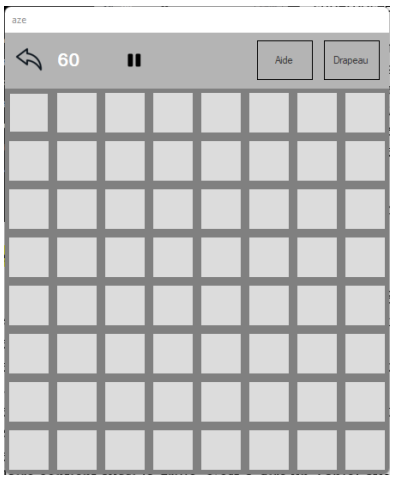
Ce formulaire est composé d'une série de *radiobuttons* avec des images superposées. Chaque *radiobutton* représente un thème et l'affichage des images est synchronisé en fonction du pays choisis par le joueur. Nous avons utilisé la propriété *tabindex* des *radiobuttons* pour pouvoir naviguer entre les différents thèmes avec les flèches directionnelles du clavier.

Figure 4. *Formulaire des thèmes*

e. Jeu

Figure 5. *Formulaire de jeu (Thème original)*

C'est le formulaire principal de l'application. Il est lancé lorsque le bouton "Nouvelle partie" de "Accueil" est cliquée. Il contient la grille et les options "in-game" du jeu :



- Le bouton retour vers l'accueil.
- Le bouton d'aide qui permet de jouer un coup (soit dévoiler une case soit marquer une case).
- Le bouton pour activer le mode "Drapeau" : cela permet de marquer les cases en cliquant dessus.
- Le *timer* pour montrer au joueur le temps qui lui reste

Le formulaire contient aussi la grille, c'est-à-dire un TableLayoutPanel composé de bouton ou de label. La taille du formulaire varie en fonction de la taille de la grille. Nous avons utilisé un TableLayoutPanel pour pouvoir y ajouter facilement des contrôles à l'intérieur et pour que ces contrôles s'organisent automatiquement. **Une partie se lance au moment où le joueur a cliqué sur une case**. De ce fait, il ne peut pas perdre dès son premier coup (comme dans le jeu original). La partie s'ouvre ainsi sur une case vide et place les bombes ensuite. Cela permet de rendre le jeu plus logique et sans la mauvaise surprise de perdre au premier coup.

II.2. Les modules

a. Données

Ce premier module stock toutes les données du jeu, c'est-à-dire les réglages de dimensions de la grille, les Box, le compte à rebours, le placement de chaque bombe, le thème choisi et la liste des boutons à découvrir. Ce module va être utilisé par le formulaire jeu pour lier les données aux actions du joueur sur la grille. Il permet d'enregistrer tous les paramètres d'une partie (la taille, le nombre de bombes, le temps limite, le thème) et ainsi de définir les valeurs de chacune des Box (Bombe, Nombre ou Vide). Le module calcule et répertorie les cases vides à dévoiler que le formulaire "jeu" va utiliser pour dévoiler les boutons. Le module va aussi servir à renseigner la meilleure case à marquer d'un drapeau et la meilleure case à dévoiler. Enfin, le module contient une méthode pour naviguer parmi la grille avec les flèches.

b. stockJoueur

Ce module va nous permettre d'une part, de stocker tous les joueurs mais aussi de les enregistrer. Donc une méthode va enregistrer un joueur s'il n'existe pas et le stocker avec tous les joueurs dans une Liste de Joueurs. Ce module va aussi actualiser les données des joueurs en fonction des parties jouées. Le module va également servir à vérifier l'existence d'un joueur, trouver un joueur rafraichir les listes de joueurs à afficher sur les formulaires accueil et scores et aussi de récupérer les joueurs dans le fichier de configuration au démarrage du jeu et d'écraser le contenu à la fermeture du jeu.

II.3. Les structures

a. Joueur

Cette structure est importante pour la gestion des joueurs et de la sauvegarde. Chaque joueur est créé au lancement d'une partie sauf si ce dernier est déjà présent dans la base de données du jeu.

Un joueur est caractérisé par :

- Un nom
- Un nombre symbolisant le meilleur nombre de bombes découvertes
- Un temps associé à cette performance
- Le nombre de parties jouées

- Le temps passé sur l'application

Toutes ces caractéristiques sont des attributs d'instance de la classe Joueur. Nous avons mis ces attributs en privé pour respecter le principe d'encapsulation. On a dû faire des *getter/setter* pour pouvoir récupérer et modifier les données. Ces informations sont initialisées par le constructeur. Les instances seront créées par le module stockJoueur (cf. II.2.b).

b. Box

Cette structure représente une case de la grille du démineur. Elle est créée au tout début de la partie et elle est contenue dans une case d'un TableLayoutPanel.
Une Box est caractérisée par :

- Une valeur : représente le type de valeur de la case, c'est-à-dire si c'est une case qui contient un nombre, une bombe ou rien ("vide").
- Un état : représente l'état dans lequel la case est visible par le joueur, c'est-à-dire si c'est une case qui est face cachée ("inconnu"), face visible ("connu") ou marquée par un drapeau ("flag").
- Un compteur de bombe : compte le nombre de bombe autour de la cette case.
- Un bouton : c'est sa représentation visuelle quand la case est face cachée.
- Un label : c'est sa représentation visuelle quand le case est visible avec en guise de texte la valeur de son compteur de bombe (lorsque ce n'est ni une bombe, ni une case vide).

La taille de la Box varie en fonction du nombre de cases dans la grille. Plus la grille contient de Box, plus les cases seront petites.

c. Thème

Cette structure représente le thème de la grille. Elle permet de personnaliser le jeu avec plusieurs couleurs. Elle est choisie par défaut avec un style sobre et neutre mais peut être changée dans les réglages à travers plusieurs associations de couleurs prédéfinies.
Un Thème est caractérisé par :

- Une couleur des Box de la grille
- Une couleur des bordures des Box de la grille
- Une couleur du texte des Labels des Box de la grille
- Une couleur qui représente le drapeau

Partie III : Gestion de la sauvegarde

Comme expliqué au II.2.b, toute la gestion des joueurs est effectuée par un module. Pour retrouver les informations des joueurs à partir d'un fichier texte, il suffit d'écrire les informations ci-dessous en respectant la syntaxe :

[*nom*] [*nb max de cases démasquées*] [*tps de la performance*] [*nb de parties jouées*] [*tps passé sur le jeu*]

Le module va automatiquement écrire dans cette syntaxe les nouvelles informations sur les joueurs pour pouvoir les récupérer à son prochain lancement.

Partie IV : Fonctionnalités rajoutées

Disclaimer : Nous avons réussi à implémenter toutes les fonctionnalités du sujet (incluant les fonctionnalités « bonus »).

Voici la liste des fonctionnalités en plus que nous avons imaginées et implémentées :

IV.1. Les images

Nous avons rajouté des images que nous avons importé dans notre solution pour rendre la navigation plus agréable dans notre application. Comme image il y a une disquette pour le bouton d'enregistrement, une flèche retour pour revenir en arrière, des rouages symbolisant les réglages et une série d'images qui permettent au joueur de choisir le thème. Le fait de créer un formulaire en plus pour les thèmes nous a permis d'avoir assez de place pour faire défiler les thèmes à l'aide de *radiobutton*. Nous avons utilisé tous les outils vus en cours. Nous sommes particulièrement contents de la superposition du cadenas fermé et ouvert qui permettent de verrouiller la zone de texte du chemin d'accès (figure 6 et figure 6bis). Cet ensemble d'images nous semblent rendre le jeu plus ergonomique car il diminue aussi le nombre de zones de textes qui pourraient être fastidieuses à lire. Nous avons essayé de garder une homogénéité avec les flèches qui marquent un retour en arrière en les plaçant sur tous les formulaires.

Figure 6. *Cadenas fermé*

Figure 6bis. *Cadenas ouvert*

IV.2. Les drapeaux et le clic droit

Cette fonctionnalité n'était pas spécifiée dans le sujet mais nous avons eu l'idée de reproduire les drapeaux, qui sont bien entendu propres au jeu d'origine. On a pris le soin de changer la couleur lorsqu'une case est marquée par un drapeau (figure 7). Pour améliorer l'expérience utilisateur, nous avons rajouter un évènement au clic droit de la souris qui permet de marquer une case d'un drapeau. La couleur de la case marquée par un drapeau sera l'une des couleurs secondaires qu'on peut retrouver sur le drapeau du pays. Pour réaliser cela, nous avons dû effectuer des recherches sur internet car il a fallu différencier les actions en fonction de la touche de la souris utilisée (clic gauche pour démasquer / clic droit pour marquer).

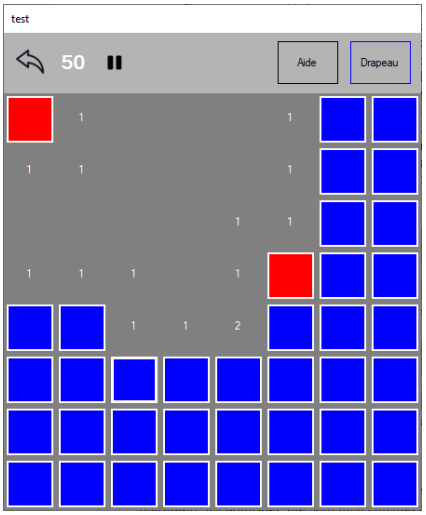


Figure 7. *Partie du démineur avec les drapeaux (cases rouges)*

IV.3. L'aide

Le bouton *aide* permet d'aider le joueur lorsqu'il se trouve dans des situations délicates. Attention, notre aide n'est pas une triche. C'est-à-dire qu'elle permet d'indiquer une case s'il y a une solution. Elle ne permet en aucun cas d'indiquer une case qui ne peut être trouvée par la logique. Donc si vous êtes dans une situation où vous devez user de la chance, l'aide ne vous aidera pas. En revanche elle le fera si une case peut être démasquée logiquement. L'algorithme à écrire a été assez complexe et nous avons passé beaucoup de temps pour qu'il ait cette « intelligence » de na pas aider robotiquement le joueur car notre application connaît l'emplacement des bombes. Pour aider le joueur, l'aide va juste marquer une case par un drapeau qui va permettre de signaler au joueur que cette même case détient une bombe.

IV.4. Explorateur de fichier

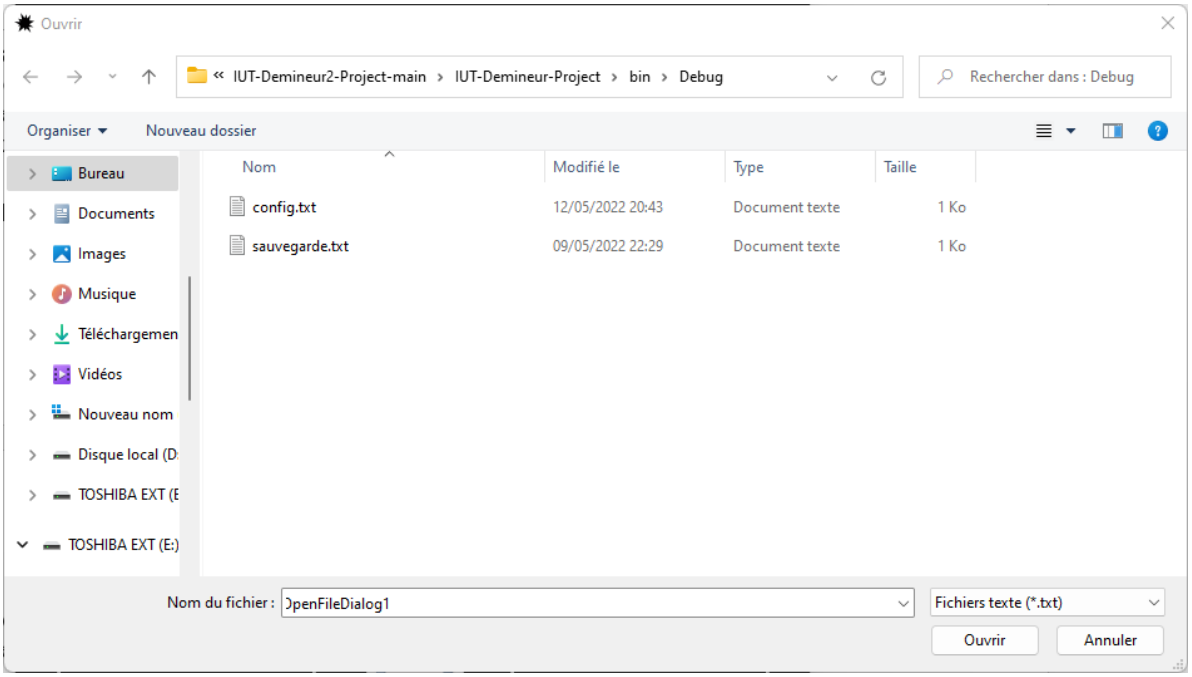


Figure 8. *Explorateur de fichiers pour le trouver le fichier de configuration*

Le bouton en forme de dossier permet à l'utilisateur de trouver plus facilement l'emplacement du fichier de config (figure 8). En effet, sans lui, le joueur ne peut pas lancer de partie. On laisse tout de même la possibilité au joueur de rentrer l'adresse du fichier mais l'option d'aller le chercher graphiquement nous semble plus ergonomique. Nous avons appliqué un filtre afin que la fenêtre affiche uniquement les fichiers de type texte (.txt).

IV.5. Jouer au jeu sans souris

Notre jeu est tout à fait jouable sans souris. Toutes les cases sont assez bien organisées pour pouvoir être pointé à l'aide du clavier. Les 4 flèches directionnelles permettent de facilement naviguer sur la grille. La touche entrée permet de dévoiler la case et on pourra toujours utiliser le clic droit pour masquer les cases.

IV.6. Le curseur

Nous utilisons un autre curseur que celui par défaut (figure 9). Une fonction a été créée pour appliquer le curseur à la page une fois que celle-ci est activée.

Figure 9. *Curseur de notre jeu*

Partie V : Diagramme de l'application

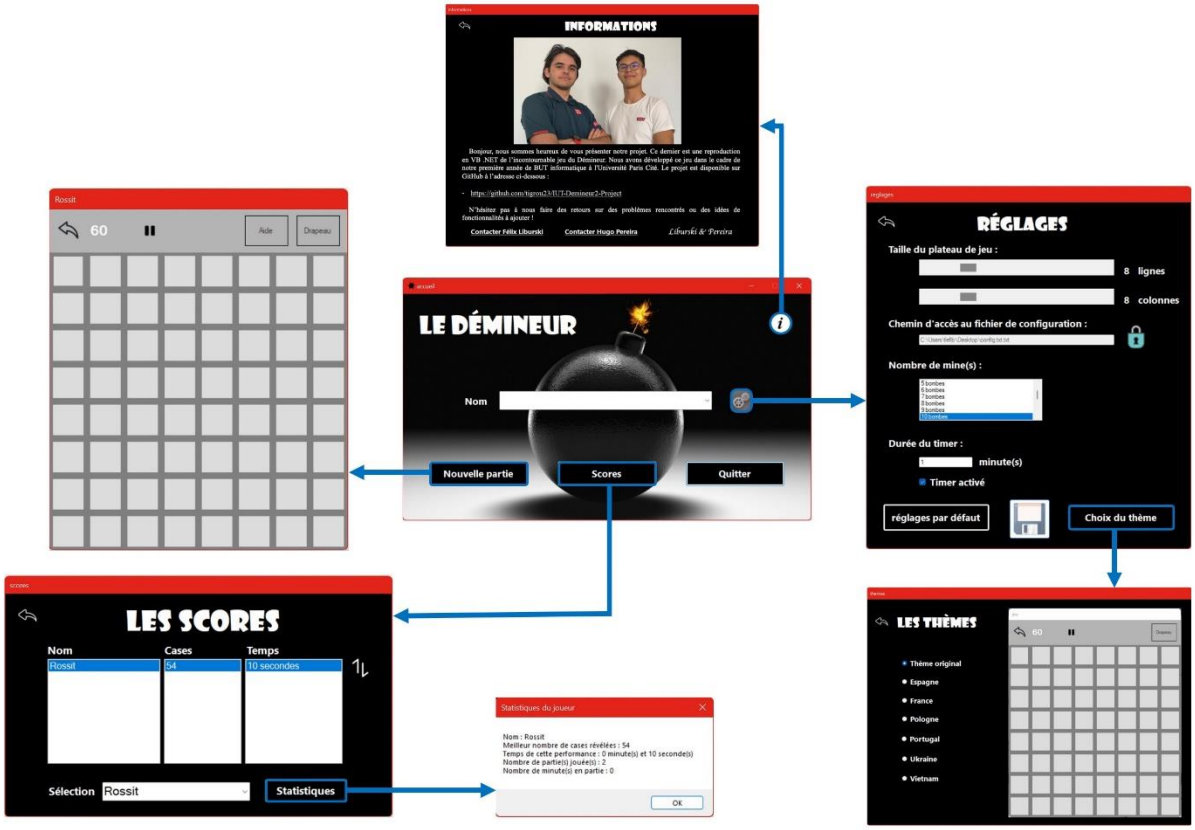


Figure 10. *Schéma*

Partie VI : Conclusion

Pour réussir à programmer ce jeu, nous avons séparé le travail en deux parties pour qu'on puisse chacun travailler de notre côté. Félix s'est occupé de programmer le jeu du démineur en lui-même et Hugo de faire le reste (réglages, scores, gestion des joueurs). Nous travaillons ensemble depuis maintenant 6 mois donc nous arrivons à avoir une très bonne cohésion. Cela nous permet d'être plus efficace car nous connaissons le style de l'autre donc on peut avoir un code de plus en plus homogène. Nos capacités de travail sont équivalentes donc cela facilite le travail en équipe. Nos connaissances sur le développement objet nous ont beaucoup aidé car le VB .NET est un langage orienté objet. Pour conclure, nous sommes très contents de notre travail car nous y avons investi beaucoup de temps. Nous avons essayé d'utiliser toutes les connaissances qu'on a pu acquérir cette année en développement.