

Table des matières

1. Introduction	1
2. Structure du code	2
3. Choix du sujet	3
4. Structure du projet	3
5. Développement des modules	4
5.1. Interface de lancement	4
5.2. Acquisition des paramètres de la partie	5
5.3. Génération de la grille	5
5.4. Saisie d'un mot	6
5.5. Vérification d'un mot	6
5.5.1. Vérification dans la grille	6
5.5.2. Vérification dans la liste des mots français	7
5.6. Mise en place du timer	7
5.7. Les scores	8
5.8. Menu des scores	9
6. Conclusion	9

1. Introduction

Dans le cadre de notre UV (Unité de Valeur) d'informatique IF2B, nous avons réalisé un projet. Les sujets proposés étaient un jeu d'échecs modifié (Really Bad Chess) et le célèbre jeu de lettres Boggle.

Nous avons choisi le deuxième jeu, le Boggle, pour des raisons que nous expliquerons plus loin. Ce jeu consiste à trouver un maximum de mots dans une grille de lettres aléatoires dans un temps imparti.

Ce projet a pour but de mettre en œuvre nos connaissances acquises au cours du semestre de printemps 2022. Notre travail a été axé sur la communication grâce aux 2 outils présentés plus loin.

2. Structure du code

- ***main.c*** :

Fichier contenant les différentes parties du code nécessaire au jeu (menus, paramètres, création de la partie...).

- ***fonctions globales.c / fonctions globales.h*** :

Fichiers contenant des fonctions utilisées par plusieurs modules du code.

Appelé par *main.c*.

- ***menu.c / menu.h*** :

Fichiers contenant les fonctions relatives aux différents choix que peut faire l'utilisateur en lançant le jeu.

Appelé par *main.c*.

- ***menu score.c / menu score.h*** :

Fichiers contenant le menu dont l'utilisateur se sert pour observer les scores en fonction de la grille ou du pseudo d'un joueur.

Appelé par *main.c*.

- ***generation grille.c / generation grille.h*** :

Fichiers contenant plusieurs fonctions afin de générer la grille par sous carré 3x3.

Appelé par *main.c*.

- ***mots.c / mots.h*** :

Fichiers contenant l'ensemble des fonctions nécessaires au parcours de la grille lors de la recherche d'un mot. Contient également la vérification de l'existence du mot dans la langue française.

Appelé par *main.c*.

- ***calcul score.c / calcul score.h*** :

Fichiers contenant la fonction permettant de calculer le score d'un joueur lorsque la partie se termine.

Appelé par *main.c*.

- ***traitement score.c / traitement score.h*** :

Fichiers contenant la fonction permettant de trier le fichier *score* par score décroissant.

Appelé par *main.c*.

- ***affichage des scores.c / affichage des scores.h*** :

Fichiers contenant les fonctions permettant d'afficher les meilleurs scores à l'utilisateur selon son choix (par joueur (recherche par pseudo) ou par grille).

Appelé par *main.c*.

3. Choix du sujet

Afin de choisir le sujet du projet nous avons choisi d'établir un argumentaire. Florian représentait le Really Bad Chess et Paul le Boggle. Au cours d'un dialogue entre nous, nous avons déterminé que notre sujet de projet serait le Boggle car au premier abord, le principal obstacle était le calcul de l'échec et mat.

4. Structure du projet

La communication est importante dans ce type de projet. Notre environnement de développement (IDE) est CLion de JetBrains. Pour travailler ensemble, sans se mettre des bâtons dans les roues, nous avons choisi d'utiliser un outil professionnel de développement, Github. Nous n'avons pas choisi un outil présent par défaut pour travailler en simultané. De plus, l'intégration de Github à CLion est plus que simple.



Image 1 : Illustration Github

Paul, utilisateur plus expérimenté de Github, a créé un dépôt (dossier de travail sur Github) et a paramétré les deux machines pour que tout fonctionne correctement.

Après avoir installé Github sur nos machines respectives (Florian sous Windows, Paul sous MacOS), nous avons mis en place quelques règles à respecter impérativement pour ne pas se perdre. Les voici :

- Toute fonction commence par une majuscule
- Tout fichier .c est accompagné d'un .h (sauf le main)
- Toute ligne de code doit être commentée quand elle est écrite
- Dès que l'un de nous s'arrête de travailler, il écrit des commentaires sur son travail et l'envoie sur notre dépôt
- Avant d'effectuer toute modification, on doit mettre à jour le projet sur notre machine, on récupère le code sur Github
- On ne travaille jamais sur le même processus en simultané sous peine de se mélanger les pinceaux

5. Développement des modules

5.1. Interface de lancement

Au début du projet, nous avons commencé par créer l'interface de lancement du jeu. Le menu se découpe en 3 parties :

- Jouer (Pour commencer une partie)
- Score (Afin de voir le meilleur score pour les différentes configurations de jeu possible en fonction du temps)
- Quitter (Pour sortir du jeu)

Nous avons donc décidé de créer une fonction dans un fichier *menu.c* accompagné de son header *menu.h* afin de ne pas surcharger le *main.c*. De plus, cette segmentation nous a permis de trouver nos erreurs avec plus de simplicité.

5.2. Acquisition des paramètres de la partie

Le premier paramètre d'une partie est la dimension de la grille souhaitée. Nous avons donc créé une fonction *Dimension_grille* permettant d'acquérir la dimension souhaitée par l'utilisateur. Toutes les fonctions qui sont relatifs à la génération grille sont écrites dans le fichier *generation grille.c* accompagné de son header *generation grille.h*. Pour terminer l'acquisition des paramètres de la partie, nous avons créé une fonction *Temps_de_la_partie* permettant à l'utilisateur de choisir le temps de sa partie. Cette fonction est écrite dans le main avec son prototype.

5.3. Génération de la grille

Concernant la génération de la grille nous avons opté pour découper cela en plusieurs parties. Tout d'abord, nous avons une fonction générant une lettre aléatoire selon des probabilités données en annexe dans le sujet. Ensuite, un sous-carré de 3x3 est généré avec des lettres aléatoires, ce sous carré ne comporte pas de doublons et est assigné à la grille juste après sa génération à l'aide d'une fonction. Le sous carré 3x3 est décalé vers la droite où sont générés 3 nouvelles lettres. Ce nouveau sous carré généré est vérifié pour ne contenir aucun doublon puis est assigné à la grille. Ce processus est répété jusqu'à la fin de la longueur de la grille.

Le sous-carré est revenu à l'origine et descend d'une ligne pour permettre de générer les 3 nouvelles lettres. A présent, pour effectuer le décalage vers la droite, une seule nouvelle lettre doit être générée donc nous avons mis en place une fonction pour cette génération. La vérification de cette nouvelle lettre générée est une vérification particulière car la lettre doit être différent non pas du sous carré mais aussi de toutes les lettres dans un rayon de 2 lettres autour. Ce nouveau procédé est répété jusqu'à la fin de la génération de la grille.

5.4. Saisie d'un mot

Pour que l'utilisateur puisse saisir un mot, nous avons décidé de créer un tableau dynamique à deux dimensions qui sera donné en paramètre dans une fonction pour la vérification d'un mot. Nous sommes partis sur le principe que l'utilisateur ne saisira pas plus de 2 mots à la seconde.

Lors de la saisie d'un mot, nous l'incrémentons dans le tableau prévu à cet effet puis nous vérifions que ce mot n'a pas été déjà saisi, si c'est le cas, on l'efface du tableau et on demande un nouveau mot.

5.5. Vérification d'un mot

5.5.1. Vérification dans la grille

```
Nombre de mots valides : 0
S  I  H  C  I
R  L  O  A  T
A  T  E  P  L
I  V  N  R  C
L  M  K  I  S
Saisir un mot :
cris
```

Image 2 : Illustration de la recherche d'un mot dans la grille

Ce processus est décomposé en plusieurs fonctions, nous allons en décrire le fonctionnement global. Tout d'abord, pour vérifier qu'un mot est dans la grille, nous voulons savoir combien de fois la première lettre du mot est présente dans la grille. Nous allons répéter le procédé suivant le nombre de fois que la première lettre du mot saisi est présente dans la grille.

Nous récupérons les coordonnées matricielles de la lettre (la première au début) en commençant par la ligne puis la colonne. Nous sauvegardons ces coordonnées dans des variables prévues à cet effet. Ensuite, nous récupérons les lettres autour de cette lettre pour savoir si la lettre suivante est présente dans les lettres autour. Si c'est le cas, on continue le procédé pour valider toutes les lettres sinon on cherche des nouvelles coordonnées de la première lettre dans la grille.

Si on valide toutes les lettres alors le mot est valide dans la grille sinon il n'existe pas dans la grille.

5.5.2. Vérification dans la liste des mots français

Pour savoir si un mot est français nous avons pris sur Internet une liste de plus de 336 000 mots présents dans la langue française. Notre liste pesant plus de 4 MB, nous l'avons scindée en 2 parties puisque CLion ne gère pas les fichiers .txt de plus de 2.56 MB. Nous avons ensuite demandé au programme de parcourir les 2 fichiers à la suite.

Dans la fonction *Verification_français* on donne en paramètre le mot à vérifier et on parcourt chaque ligne de la première moitié de la liste et on la compare au mot à vérifier. On fait la même chose avec l'autre moitié de la liste de mots. La fonction renvoie 1 si le mot est valide et 0 sinon.

5.6. Mise en place du timer

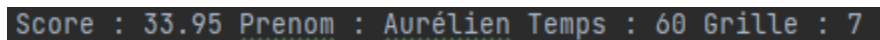
Pour que la partie ne dure pas indéfiniment, un timer a été mis en place. On prend une mesure de temps avant l'affichage de la grille puis une deuxième mesure une fois un mot saisi. On demande la saisie d'un mot tant que le temps d'exécution total ne dépasse pas le temps limite saisi

en paramètre. Ce timer a été mis en place à l'aide de M. Hilaire lors des séances de TD car nous avons rencontré quelques problèmes du fait que Paul travaille sous MacOS et Florian sous Windows.

5.7. Les scores

Afin de calculer le score, nous avons suivi la formule donnée dans le sujet. Le score est calculé d'une traite à la fin de la partie.

Une fois le score calculé, nous demandons à l'utilisateur son pseudo. Une fois acquis, nous stockons dans un fichier son score, son pseudo, le temps limite ainsi que la dimension de la grille, le tout sous forme d'une seule ligne.



```
Score : 33.95 Prenom : Aurélien Temps : 60 Grille : 7
```

Image 3 : Illustration du stockage d'un score dans le fichier des scores

Pour le tri décroissant du score, nous avons décidé d'extraire chaque ligne du fichier score et de les mettre dans un tableau dynamique. Ensuite nous trions ce tableau via un tri à bulles.

Nous réalisons ensuite le tri du fichier à l'aide d'un algorithme de tri à bulle et d'un tableau dynamique à deux dimensions pour simplifier la manipulation. On ré-ouvre le fichier en écrasement (mode 'a') et on réécrit le tableau dans le fichier avec une ligne indiquant la fin de fichier.

5.8. Menu des scores

Le menu des scores permet à l'utilisateur de pouvoir rechercher les meilleurs scores par grille ou par pseudo. Pour la recherche par pseudo, nous réalisons un parcours de la ligne jusqu'à trouver le pseudo recherché. Si tel est le cas, nous affichons alors la ligne.

Pour la recherche par grille, connaissant la position du caractère relatif à la dimension de la grille utilisée pour réaliser le score, nous récupérons ce caractère puis le comparons à la dimension recherchée. Si elles concordent, nous affichons la ligne.

6. Conclusion

Notre projet est terminé et fonctionnel. Cependant, il reste des parties du code optimisables. En effet, certaines parties telles que la génération de la grille ne semblent pas optimisées. De plus, nous ne sommes probablement pas passé par les méthodes les plus simples afin de réaliser le jeu mais les plus logiques pour nos personnes respectives.

Nous sommes très contents de ce que nous avons fait et nous avons pu apprendre comment gérer un projet, se répartir le travail ou encore, comment développer un mini jeu. Ce fut donc une expérience très intéressante et enrichissante. C'est dans ce cadre de travail que l'on voit l'importance de la communication.