Table des matières

[1. Introduction 2](#_Toc105492162)

[2. Choix du sujet 3](#_Toc105492163)

[3. Structure du projet 4](#_Toc105492164)

[4. Développement 4](#_Toc105492165)

[1) Préparation de l’interface de lancement 5](#_Toc105492166)

[2) Codage de l’interface de lancement 5](#_Toc105492167)

[3) Génération de la grille 5](#_Toc105492168)

[4) Recherche d’un mot dans la grille 6](#_Toc105492169)

[5) Vérification d’un mot français 7](#_Toc105492170)

[6) Mise en place du timer 7](#_Toc105492171)

[7) Les scores 7](#_Toc105492172)

[8) Menu des scores 8](#_Toc105492173)

[5. Conclusion 9](#_Toc105492174)

[6. RÉFÉRENCES 9](#_Toc105492175)

# 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.

# 

# 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.

# 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

# Développement des modules

## 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 décider 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.

## 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.

## 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 généré 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 autours. Ce nouveau procédé est répété jusqu’à la fin de la génération de la grille.

## 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.

## Recherche d’un mot dans la grille

Calendar

Description automatically generated

*Image 5 : Illustration de la recherche d’un mot dans la grille*

Afin de pouvoir vérifier que le mot saisi par le joueur / utilisateur, nous avons appliqué le raisonnement suivant, en réalisant parfois des adaptations nécessaires (position de la lettre dans la grille (milieu, coin ou bord), dernière lettre, mauvaise première lettre, …).

Afin de rechercher un mot dans la grille, nous avons créé un nouveau fichier (*mots.c*) afin de pouvoir trouver nos erreurs de code plus simplement. Dans ce fichier nous commençons par acquérir le mot que l’utilisateur a saisi. Ceci fait, nous avons créé une fonction qui parcours la grille afin de rechercher le mot. Nous effectuons cette recherche lettre par lettre jusqu’à ce que le mots soit entièrement trouvé.

Notre réflexion a été modélisé tel un algorithme, une série d’instructions :

1. Quelle est la 1ère lettre ?
2. Combien de fois est- elle présente dans la grille ?
3. 1ère fois
   1. Quelles sont les lettres autour d’elle ?
   2. Est-ce que la 2e lettre est présente ?
      1. Si non alors on re-parcours à partir de la lettre suivante dans la grille
      2. Si oui alors prendre les coordonnées de la 2ème lettre
   3. ième lettre en partant avec les coordonnées de la (i-1)ème lettre
4. On répète le programme tant que nous n’avons pas parcouru le nombre d'occurrence de la première lettre dans la grille.

## Vérification d’un mot 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 que nous l’avons scindé 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.

## Mise en place du timer

Afin que la partie ne dure pas indéfiniment, nous avons mis en place le timer (également demandé dans le sujet). Paul l’a réalisé et testé. Nous avions alors un timer fonctionnel. Le problème c’est qu’il fonctionnait uniquement sur son ordi (OS Mac). Nous avons alors procédé à des tests pour qu’il fonctionne sur les deux ordis mais cela n’a rien changé. J’ai alors refait un timer sur mon ordinateur (OS windows). Il fonctionnait alors sur mon PC mais plus sur le sien. C’est alors que nous avons compris que le timer dépendait de l’OS et avons donc conservé chacun le nôtre en tant que solution temporaire.

Par la suite, nous avons cherché une autre solution, aidé par Vincent HILAIRE. Nous avons pu trouver le problème généré par les deux OS et refaire un timer commun.

## Les scores

Après avoir saisi le mot, avoir contrôlé sa présence et son existence, nous avons procédé au calcul du score.

Afin de calculer le score, nous avons suivi la formule donnée dans le sujet. Nous avons choisi de faire le calcul en 1 bloc à part et à la fin de la partie afin de ne pas avoir le score au milieu du code et risqué d’avoir une erreur lors du calcul.

Le score est calculé une fois tous les mots trouvés grâce au parcours du tableau dans lequel on stock les mots trouvés par le joueur.

Une fois le score calculé, nous demandons au joueur son pseudo. Une fois acquis, nous le stockons dans un fichier contenant les scores. Lorsque nous le stockons, nous y associons le temps et les dimensions de la grille.



*Image 9 : Illustration du stockage d’un score dans le fichier des scores*

Nous réalisons ensuite le tri du fichier à l’aide d’un algorithme de tri à bulle et d’un tableau temporaire pour simplifier la manipulation.

## Menu des scores

Pour le menu des scores, nous avons une fois de plus suivi les indications du sujet.

Le menu des scores permet à l’utilisateur de pouvoir rechercher les meilleurs scores par grille ou bien par pseudo. Nous avons donc créé 2 fonctions, une par cas.

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.

# Conclusion

Notre projet est terminé et fonctionnel. Cependant, il reste des parties du code dont on peut encore parler. En effet, certaines parties telles que la génération de la grille ne semblent pas optimisées dans la façon dont nous avons procédé. De plus, nous ne sommes probablement pas passé par les méthodes les plus simples afin de réaliser le jeu.

Nous sommes cependant 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.

# RÉFÉRENCES

1. Le site : Open Classroom

CONSIGNES

- Page de garde, avec la liste des membres du projet et le sujet choisi

- Introduction brève (moins de 0.5 pages) dans laquelle vous reformulez les objectifs du projet, et annoncez la structure de votre rapport

- Sommaire (automatique)

- Présentation générale de la structure de votre code : décomposition en différents modules, si des dépendances inter-modules sont présentes il faut les détailler (par un schéma éventuellement), description brève des fonctionnalités de chacun des modules

- Présentation des fonctionnalités effectivement implémentées : présentation du menu général, présentation de l'écran de configuration du jeu, présentation du jeu, présentation de la sauvegarde, etc. N'hésitez pas pour chaque grande fonctionnalité à illustrer avec une capture d'écran (proprement commentée) ; si des difficultés ou des détails d'implémentation pertinents concernent cette fonctionnalité, vous pouvez les détailler.

- Conclusion : faites le bilan de ce qui a été effectivement implémenté (vous pouvez même faire une checklist), et ce qui aurait pu être implémenté. Ensuite, vous pouvez par exemple ouvrir la conclusion sur une prise de recul dans laquelle vous critiquez les choix réalisés dans la conception du projet, et indiquer quelles modifications vous feriez si vous deviez refaire ce projet.

Attention, ne mettez pas de capture d'écran du code, ne copiez-coller par votre code dans le rapport. Le code sera fourni à côté et nous le consulterons directement.

Bien entendu, le code devra être correctement commenté et documenté.