# Petit Rapport – Projet 2048

## Liste des fonctionnalités du jeu

*Fonctionnalités basiques :*
-Affichage du plateau de jeu 4x4 tel que demandé
-Demande le mouvement via *cin* en c++ (Z,Q,S,D), nécessite de presser Entrer
-Déplacements qui respectent les règles du jeu basique 2048
-Affichage d'un message lorsque le jeu est gagné ou perdu (bloqué)
-Fonctions testées dans un fichier annexe
-Apparition aléatoire sur la grille d'un 2 ou 4 avec les probabilités correspondantes
-Affichage d'erreur lorsque la commande entrée n'est pas reconnue

*Premières extensions :*
-Ajout du système de score, affichage du score
-Ajout de la librairie Ncurses, soit :
    -Affichage qui s'actualise au lieu de se réécrire en dessous
    -Ajout de couleurs
    -Prise de décisions via les flèches au lieu des (4 touches +Entrée)

*Mes extensions :*
-Possibilité de relancer le jeu via la touche R
-Enregistrement du Highscore (dans un fichier externe) et actualisation à chaque coup si celui-ci est battu au cours d'une partie.
-Possibilité de revenir 1 coup en arrière mais entraîne une perte de 5% de score (via Espace)

*Ma variante :*
-Diminution ou Augmentation d'une case aléatoire du jeu avec des probabilités respectives à chaque coup de 2% et 4%. Cette modification se fait en fonction de la valeur de la case (sur une augmentation, 2 passe à 32, 128 à 256, 1024 n'augmente pas).
Un message s'affiche indiquant le changement, il est coloré en fonction de la nouvelle (Vert Augmentation, Rouge Diminution, Blanc neutre).
*Règles précises dans le rapport complet page 43.*
-Possibilité de supprimer une case de son choix avec E après 70 coups (réutilisables). Tant que la suppression n'est pas disponible, la ligne de commande s'affiche en rouge, puis en vert une fois disponible (soit, après 70 coups). Si la touche E est pressée avant les 70 coups, un message s'affiche annonçant le nombre de coups restants avant d'atteindre 70.
-Changement des touches pour qu'elles soient toutes accessibles facilement d'une main (Z, E, R, Q, S, D, Espace).

# Complexité des fonctions :

Mes fonctions de déplacements sont divisées en 2 fonctions. Les **déplacements partiels**, qui décalent toutes les cases du côté choisi. Et les fonctions de **Fusion**, qui fusionnent 2 cases si celles-ci ont la même valeur, puis relancent une fonction de Déplacement Partiel. De plus, mon extension du Highscore est intégrée à la fonction mouvement.
La complexité ne sera donc pas la même si une fusion a lieu ou non.

Le nombre d'opérations va dépendre de chaque plateau de jeu (dépend du nombre de cases déplacées, du nombre de cases qu'elles parcourent, ainsi que du nombre de fusions).

## Fonctions Déplacements :

Je choisi comme opération élémentaires :
-Ecriture d'une nouvelle valeur de case du tableau 2D 4x4
-Test si une case est égal à une certaine valeur

La fonction déplacement Partiel :
-Scan chaque case du tableau pour voir si celle-ci est vide. Donc 16 opération.
-Si une case est vide, elle prend la valeur de la case d'à côté, ce qui prend 2 écritures. Si on considère que **n** cases se déplacent par coup en moyenne, cela rajoute **n** opérations. Donc 16+2**n** au total.
-La fonction Déplacements Partiel fait toutes ces opérations 3 fois, donc la complexité est de l'ordre de **3(2n+16)** opérations.

## Fonctions fusions :

Je choisi comme opération élémentaires :
-Ecriture d'une nouvelle valeur de case du tableau 2D 4x4
-Test si une case est égal à une certaine valeur
-Changement de valeur d'un entier

La fonction Fusion :
-scan chaque case du tableau pour voir si celle-ci est égal à sa voisine, Donc 16 opération.
-Si c'est le cas, elle réécrit 2 cases du tableau, change le score, et modifie la variable locale. Donc 4 opérations supplémentaires pour chaque fusion. Pour une moyenne de **m** fusions par coup, on aura donc environ **16+4m**.
-Si une fusion a lieu, la fonction Déplacements Partiel est relancé, mais celle-ci déplacent moins de cases en moyenne après une fusion. On fait l'approximation qu'elle en déplace 3 fois moins. La fonction fait un test pour savoir s'il faut lancer cette fonction, donc on rajoute 1. La complexité est donc de **(17+4m) + (2n+16)** pour la fonction Fusion.

## Fonction Highscore :

Cette fonction permet de stocker le meilleur score enregistré. Or la plupart du temps, il n'est pas battu au cours de la partie, on considère donc ici que le score ne sera pas battu.

Je choisi comme opération élémentaires :
-Ouverture d'un fichier et fermeture
-Test d'inégalité
-Récupération d'un entier provenant du fichier externe

La fonction Highscore :
-Ouvre le fichier et le ferme une fois la valeur récupérée, 2 opérations
-Récupère la valeur du fichier
-Test si cette valeur est plus petite que le score actuel
On a dit qu'on faisait l'approximation que le test serait faux tout au long de la partie, donc la fonction s'arrête là.

La fonction Highscore fait donc 4 opérations élémentaires.

## Fonction Déplacement totale :

On additionne les complexités de chaque fonction qu'elle appelle.
On a fait les approximations suivantes :
**m** le nombre de fusion en moyenne par coup
**n** le nombre moyen de cases qui bougent par coup
Le Highscore ne sera **pas battu**.

Alors la complexité sera de :
3(2n+16) + (17+4m) + (2n+16) + 4
= **8n + 4m + 85**

Pour des valeurs de m=2 et n=3, cela fait **117 opérations élémentaires par coups** en moyenne.

## Fonctions eventProba1surN :

Cette fonction prend en entrée un Float N et renvoie True si la simulation de l'évènement de probabilité 1/N s'est produit.

Je choisi comme opération élémentaires :
-Division et multiplication
-Test d'inégalité

La complexité sera de 3 opérations.

### Fonctions BaisseOuAug :

Je choisi comme opération élémentaires :
-Opération élémentaire (addition, soustraction, multiplication, division)
-Test d'égalité ou d'inégalité
-Affichage d'un message à un emplacement donné (avec ncurses)
-Entier (ou case d'un tableau 2D) qui change de valeur

Dans le cas où rien ne se passe :
La fonction effectue 2 tests. Ces 2 tests font appel à la fonction event1surN, cela nous fait 2*3= **6 opérations.** Cela se produit dans **92%** des cas (en théorie).

Si la fonction conduit à une baisse :
-Exécute la fonction eventProba1surN, donc +3 opérations
-Effectue un test. +1
-Fait 5 opérations et 4 entier qui change de valeur. +9
-Fait 4 tests, dont 3 négatifs. +4
-Fait un changement de valeur pour une case et affiche 2 messages. +3
On a donc 20 opérations dans 2% des cas.

Si la fonction conduit à une augmentation :
On a la même chose à 2 tests près. Soit 22 opérations dans 4% des cas.

On arrondi ces 2 cas à 21 opération dans 6% des cas.

On a donc 6 opérations à 92% et 21 opérations à 6%, soit une complexité de **7 opérations en moyenne**.

## Degré de confiance

Avant de rajouter les premières extensions, toutes les fonctions étaient testées via un fichier annexe au format cpp. Il utilisait son propre fichier .H.
Donc toutes les fonctions d'affichages, de probabilité et de déplacement avaient un grand degré de confiance.
Malgré cela, une erreur se situait dans une fonction déplacement, indécelable avec la plupart des tests effectués.

Par la suite, les fonctions ont été modifiées et d'autres ont été ajoutées. Elles ont été testées au fur et à mesure du codage. A présent, elles fonctionnent toutes correctement et l'idée que l'une soit défaillante et peu envisageable.

Toutes les documentations des fonctions se trouvent dans les fichiers .H.