**Rapport de projet 2048 :**

**Présentation du jeu :**

Le jeu 2048 est un jeu conçu par un développeur indépendant Gabriele Cirulli, ayant été très populaire sur les plateformes mobiles. En effet, son aspect addictif et le fait qu’il soit gratuit ont grandement favorisé le développement de sa notoriété. Le but du jeu est de combiner les tuiles de même valeur en les faisant glisser sur une grille et de créer au final une case contenant le nombre 2048.

Dans ce projet, un jeu « 2048 » a été implémenter et est fonctionnel sur un terminal. Certaines options tel une grille colorée, des déplacements avec les flèches directionnelles ou encore un calcul du score égaillent le programme tournant sur le terminal.

**Auteurs**

* RANAIVO-HARISOA Mitia, mitia.ranaivo-harisoa@u-psud.fr, MI/4

**Résumé du travail effectué**

* *Niveau 0 :*
  + 0*.*1 réalisé
  + 0.2 réalisé (pour tout type de plateau avec le même nombre de colonnes sur chaque ligne.)
  + 0.3 réalisé
  + 0.4 réalisé
  + 0.5 réalisé
  + 0.6 réalisé
  + 0.7 réalisé (ne marche que pour les plateaux de taille 4x4)
  + 0.8 réalisé
  + 0.9 réalisé
* *Niveau 1 :*
  + 1.1 réalisé en utilisant la bibliothèque ncurses
  + 1.2 réalisé en utilisant la bibliothèque ncurses
  + 1.3 réalisé en utilisant la bibliothèque ncurses
  + 1.4 réalisé grâce aux structures de données ; une structure alliant un plateau et un score s’actualisant lors de l’appel des fonctions combineCases\_haut et combineCases\_gauche.

* *Niveau 2 :*
  + 2.1 réalisé en utilisant un Makefile simple adapté au programme.
  + 2.2 réalisé en créant un repository local et en utilisant git.
* *Niveau 3 :*
  + 3.1 non réalisé
  + 3.2 non réalisé
  + 3.3 non réalisé
  + 3.4 non réalisé

**Démonstration**

Les actions de jeu en générale (déplacement) marchent pour tous types de plateau sauf pour les plateaux avec des lignes possédant des tailles de colonnes différentes.

Nous trouverons dans les tests 2 types de plateaux (3x2 et 3x5) sur lesquels les fonctions de déplacement marchent. Le programme devrait ainsi fonctionner pour les plateaux non carrés et avec un nombre de colonnes différents de 4 colonnes.

Cependant, en dépit de l’adaptation du programme à plusieurs types de plateau, l’affichage quant à lui se fait mal sur les plateaux avec un nombre de colonnes différent de 4. En effet, le plateau peut afficher un plateau avec 4 colonnes sans problème mais si le nombre de colonne varie, la taille du séparateur de ligne ne s’adapte pas au nombre de case ayant augmenté ou baissé.

La compilation ne marche pas non plus sur le jupyterhub.

Note : Vous retrouverez les tests dans le fichier test\_jeu.cpp

**Organisation du travail**

Pour le projet j’ai travaillé seul. En raison de cette particularité dans mon travail, j’ai décidé de m’y prendre très tôt pour débuter l’implémentation du programme. Ainsi j’ai commencé à implémenter le jeu le Mardi suivant l’annonce de la disponibilité des sujets en ligne.

Les séances de « code » ont duré environ 3 à 5h par séance et ont été espacé d’environ 2 jours. En effet, j’alternais les séances de code et de documentation sur certain aspect du langage qui m’étais encore un peu flou, et les nouvelles bibliothèques utiles au bon fonctionnement de mon jeu. Les plus grosses séances se sont faites les Mardis, les Jeudis et les Samedis ; et autres jours ont été consacré à la documentation et au débogage.

Pour une grande partie du projet, j’ai utilisé le JupyterNotebook afin de tester mes fonctions et voir comment mon programme réagissait vis-à-vis de certain cas ; cependant, la majorité de mon programme a été implanté directement sur VS code.

Total d’heure consacré au projet : environ 40h (en comptant les heures de documentation)

Note : Hongfei et Quentin m’ont initié au concept de structure de données ; ils m’ont appris l’existence de ceux-ci et leur utilisation.

**Prise de recul**

L’une de plus grande difficulté que j’ai rencontré lors de la réalisation de mon projet a été la gestion du temps. En effet, je devais jongler entre les différents devoirs et révisions dans les autres matières et le projet. Je devais alors me créer du temps pour faire face à ce problème, une partie du code que j’ai implémenté a été fait dans le métro avec l’application note de mon téléphone. Je pouvais alors le copier sur mon ordinateur une fois rentrer et commencer à résoudre les bugs.

Un autre aspect sur lequel j’ai eu quelques difficultés a été l’apprentissage de l’utilisation de certaines bibliothèques et la compilation de mon code. En effet, les articles sur la bibliothèque ncurses sont assez vieux et pas très lisible. Ensuite j’ai rencontré beaucoup de problèmes lors de la compilation au fur et à mesure que j’avançais dans mon projet. En effet, j’ai pris du temps avant de connaitre et d’appliquer l’option -std=c++11 lors de la compilation car sur les ordinateurs de l’école et le jupyterhub, le programme compilait sans problème ; or sur mon ordinateur personnel, une erreur se présentait lors de mes tentatives de compilation. Ainsi, le mise en parallèle entre le programme sur les ordinateurs de l’université et le miens a été une des difficultés les plus marquante de la réalisation du projet.

Nouvelles fonctionnalités apprises (non traité en cours) :

* + La bibliothèque random
  + Les options de compilation
  + Les structures de données
  + La bibliothèque ncurses
  + La compilation grâce à un Makefile
  + Utilisation de git et installation de git, Homebrew, Conda et gh (commande github) sur le terminal.

Si je devais refaire le projet, je le referais en groupe. En effet, le travail seul présente certains bons aspects ; notamment la compréhension de l’intégralité du code, et la flexibilité vis-à-vis du temps de travail ; cependant, être livré à soit même avec une limite de temps et l’impossibilité de savoir si son code est logique ou présente certaines limites d’un autre point de vue ont été des facteurs ayant rendu la réalisation du projet plus compliqué que prévu.

**Documentation de l’application :**

Pour fonctionner, le dossier comportant les fichiers de l’application doit présenter :

* 5 fichiers .cpp et 1 fichier .h
  + BasePlateau.cpp (avec les fonctions qui seront utile majoritairement à la création d’une partie)
  + EtatPlateau.cpp (avec les fonctions gérant l’état actuel du plateau)
  + OpperationPlateau.cpp (avec les fonctions permettant les déplacements et autres actions mener sur le plateau)
  + test\_jeu.cpp (les tests de toutes les fonctions)
  + my2048.cpp (avec le code gérant le déroulement d’une partie et les fonctionnalités liées à l’ajout des couleurs)
  + Jeu2048.h
* **Fonctions Implémentées, documentées et testées** :

Voir fichier Jeu2048.h

* Liste des bibliothèques utilisées :
  + String
  + Iomanip
  + Sstream
  + Vector
  + Cstdlib
  + Random
  + Assert.h
  + Ncurses

**Pour lancer le jeu : (utilisation d’un Makefile)**

1. Ouvrir un terminal
2. Se placer dans le dossier où sont situé les 6 fichiers cité précédemment à l’aide de la commande « cd »
3. Taper la commande suivante : « make » et appuyer sur « entré »

Le programme compilera tout seul.

Note : utilisation des option -std=c++11 et -lncurses dans le Makefile

Contenu du Makefile :

output: my2048.o BasePlateau.o EtatPlateau.o OpperationPlateau.o test\_jeu.o

g++ -std=c++11 BasePlateau.o EtatPlateau.o OpperationPlateau.o test\_jeu.o my2048.o -lncurses -o my2048

my2048.o: my2048.cpp

g++ -c my2048.cpp

BasePlateau.o: BasePlateau.cpp Jeu2048.h

g++ -c -std=c++11 BasePlateau.cpp

OpperationPlateau.o: OpperationPlateau.cpp Jeu2048.h

g++ -c -std=c++11 OpperationPlateau.cpp

EtatPlateau.o: EtatPlateau.cpp Jeu2048.h

g++ -c -std=c++11 EtatPlateau.cpp

test\_jeu.o: test\_jeu.cpp Jeu2048.h

g++ -c -std=c++11 test\_jeu.cpp

clean:

rm \*.o my2048

1. Pour lancer le jeu ; entrer la commande suivante : ./my2048

**Méthode de sauvegarde des versions sur Github :**

* Création d’un repository (privé) github sur le profile meethyrh.
* Mise en lien du nouveau repository avec le dossier contenant les fichiers en local.
* Commande pour sauvegarder ses fichiers :
  + git add .
  + git commit -m « label donné au commit »
  + git push origin main