**Projet : 2048**

Ce sujet est composé de quatre sections. La section 1 décrit le jeu et ses règles. La section 2 décrit en détail les consignes de base du projet pour obtenir une note correcte (12/20). La section 3 décrit les extensions possibles pour viser le 20. L'une d'entre elle est de créer un joueur automatique de 2048 le plus performant possible. Enfin, la section 4 (facultative) donne les consignes pour faire participer votre joueur automatique au tournois qui sera organisé en fin de semestre et dont les résultats seront rendus publics !

## 1. Description et règles du 2018

2048 est un jeu vidéo de type puzzle, variante du *jeu de taquin*. Il a été développé par Gabriele Cirulli en 2014 (19 ans à l'époque) et publié en ligne sous licence libre.

**Les règles du jeu.** Le jeu se joue sur un plateau $4 \times 4$ où chaque case est soit vide, soit contient une puissance de 2, inscrite sur une tuile. Le joueur peut déplacer les tuiles en les faisant glisser toutes ensemble dans une même direction (haut, bas, droite, gauche). Les tuiles ne peuvent dépasser les bords du plateau. Si deux tuiles de même valeur ($2^k$) sont adjacentes pendant le glissement, alors elles se combinent en une unique tuile étiquetée par la somme des valeurs ($2^{k+1}$). Chaque combinaison de tuiles rapporte au joueur un nombre de point équivalent à la valeur de la tuile après la combinaison. Après chaque déplacement, une nouvelle tuile apparaît aléatoirement sur un des emplacements vides. Cette nouvelle tuile a pour valeur soit 2, soit 4, avec probabilités respectives $\frac{9}{10}$ et $\frac{1}{10}$. Le jeu débute avec deux tuiles posées sur le plateau, tirées selon les probabilités mentionnées ci-dessus.

Le but du jeu est de créer une tuile portant le numéro 2048. Cependant, on pourra continuer à jouer après avoir atteint le but, en créant des tuiles avec des numéros plus grands et ainsi améliorer indéfiniment son score. Le jeu se termine lorsque toutes les tuiles sont occupées et que plus aucun mouvement ne permet de combiner de tuiles.

## 2. Consignes de base du projet

Après le démineur et le yams, vous allez implanter 2048 ! Vous devrez rendre un programme respectant les consignes décrites plus bas et permettant de jouer à 2048 et rédiger un rapport. Enfin, vous devrez présenter votre travail lors d'une soutenance orale incluant une démonstration. Pour les modalités pratiques, consulter :
`https://nicolas.thiery.name/Enseignement/Info111/projet.html`

**Niveau 0 : Pour obtenir jusqu'à 12/20.** L'objectif ici est de modéliser et d'implanter le jeu 2048 en mode console en C++. Le joueur saisira chaque déplacement (*haut, bas, gauche* et *droite*) par l'intermédiaire de caractères de votre choix. (par exemple `z`, `s`, `q` et `d` ou encore `h`, `b`, `g` et `d`). Le plateau sera affiché en début de partie et après chaque tour de jeu. Vous trouverez en figure 1 un exemple d'exécution après quelques tours de jeu.

Le programme devra **impérativement** respecter la structure suivante. Un fichier `2048.cpp` contiendra une fonction `main` qui lancera le jeu. On modélisera un plateau de jeu par un type `Plateau` (alias du type `vector<vector<int>>`). Les règles et les actions du jeu seront implantées dans un fichier `modele.cpp`, par les fonctions déclarées dans le fichier `modele.h`. Il est néanmoins possible d'ajouter des fonctions au fichier `modele.h` si besoin est, notamment pour gérer le score de la partie.

```
*************************
* 128 *  4  *     *     *
*************************
*  2  * 16  *  4  *  2  *
*************************
*     *     *     *     *
*************************
*  2  *     *     *     *
*************************

Entrer commande: h

*************************
* 128 *  4  *  4  *  2  *
*************************
*  4  * 16  *     *     *
*************************
*     *     *     *     *
*************************
*  2  *     *     *     *
*************************

Entrer commande: g

*************************
* 128 *  8  *  2  *     *
*************************
*  4  * 16  *     *     *
*************************
*     *     *  4  *     *
*************************
*  2  *     *     *     *
*************************

Entrer commande: ...
```

FIGURE 1. Quelques tours du jeu 2048 en mode console

**Barême.** La note 12/20 pourra être obtenue si

(0.1) les fichiers de votre projet compilent ;

(0.2) le jeu est *fonctionnel* ;

(0.3) le jeu respecte les règles du 2048 ;

(0.4) le score est mis à jour à chaque mouvement ;

(0.5) l'implantation respecte la structure donnée ;

(0.6) toutes les fonctions sont *spécifiées*, *documentées* et *testées*.

(0.7) lors de l'affichage, les colonnes sont bien alignées, quelles que soient les valeurs de chaque tuile :

```
Correct                      Pas correct
*************************     ****************
* 128 * 128 *     *     *     * 128 * 128 *   *   *
*************************     ****************
```

```
*  2  * 16  *  4  *  2  *       * 2 * 16 * 4 * 2 *
************************       *****************
*     *     *     *     *       *   *   *   *   *
************************       *****************
*  2  *     *     *     *       * 2 *   *   *   *
************************       *****************
```

(0.8) le rapport est bien rédigé (orthographe...) et comprend :

— une *présentation du jeu*,

— une *documentation* de votre application

(0.9) la soutenance est soignée et inclut une brève démonstration de votre jeu

**Remarque :** Vous perdrez des points en cas de non respect de la structure montrée ci-dessus. Le plagiat sera sévèrement sanctionné comme indiqué dans `https://nicolas.thiery.name/Enseignement/Info111/projet.html#plagiat`.

## 3. Extensions pour améliorer sa note

**Niveau 1 : « gagner des points avec un minimum d'efforts ».** Ce niveau propose différentes améliorations (faciles) :

(1.1) Comme sur le jeu original, rajouter un peu de couleur à notre triste console... ($\sim$ 1pt)

(1.2) On aimerait jouer en utilisant directement les flèches (*haut, bas, gauche* et *droite*) du clavier, sans avoir à appuyer sur entrée à chaque fois. Étudier différentes solutions envisageables et ajouter cette fonctionnalité. ($\sim$ 2pt)

(1.3) Encore une fois pour des raisons de jouabilité, on aimerait que l'affichage d'un plateau se fasse à la place du plateau précédent, plutôt que en dessous. Autrement dit, on souhaiterait que l'écran se rafraîchisse. Étudier différentes solutions et ajouter la fonctionnalité. ($\sim$ 2pt)

(1.4) L'utilisation de variables globales dans un code est déconseillé car elle réduit la modularité et la flexibilité du programme. Calculer le score du jeu non pas avec une variable globale mais avec une structure de données qui associe un plateau `Plateau` à un score . ($\sim$ 2pt)

Pour les trois premiers points, une solution envisageable est d'utiliser la bibliothèque `ncurses`. Un exemple minimal d'utilisation est fourni dans le fichier `ncurses_min.cpp`, à compiler avec le paramètre de compilation `-lncurses` pour l'éditeur de lien [1].

**Attention :** L'utilisation de certaines librairies nécessite d'ajouter des paramètres de compilation. Si tel est le cas, le rapport devra expliciter la procédure de compilation.

**Niveau 2 : « passer la barre des** $16 - 17$ **».** Ce niveau propose des améliorations plus difficiles de l'ordre des bonnes pratiques de programmation :

(2.1) Votre projet comportant un certain nombre de fichiers, la compilation manuelle – en appelant directement `g++` – commence à être peu pratique. Une solution classique pour automatiser cela est d'utiliser un `Makefile`. Mettez cette solution en œuvre pour votre projet. ($\sim$ 2pt) **Attention :** Deux choix s'offrent à vous, créer un (petit) `Makefile` (simple) adapté à votre projet ou trouver un `Makefile` générique qui compile tout. Pour mériter les points de la question, l'important est de comprendre ce qui se passe (n'oublions pas la soutenance !).

1. Pour de la documentation, voir par exemple `http://invisible-island.net/ncurses/ncurses-intro.html` ou `http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/`

(2.2) Utilisez de plus un gestionnaire de version comme `git` pour gérer les sources de votre projet [2] ($\sim$ 1pt)

**Niveau 3 : « Aller plus loin ! ».**

(3.1) Cette fois-ci, c'est au programme de décider comment déplacer les tuiles ! Implanter une IA ou un joueur automatique de 2048 à votre manière. Même des règles simples peuvent permettre à votre IA de marquer des points. Les IA les plus élaborées seront davantage récompensées. Si vous choisissez de faire cette extension, vous pourrez aussi participer au tournois organisé en fin de semestre. Pour cela, merci de lire attentivement et de respecter les consignes décrites dans la section 4.

(3.2) Vous pouvez par exemple implanter des variantes du jeu, comme l'une de celles décrites sur `https://phenomist.wordpress.com/2048-variants/` ou, encore mieux, la vôtre. Et créer un menu en début de jeu pour choisir la variante.

(3.3) La console est un peu triste, vive les graphiques ! En utilisant la `SFML`, ou directement `SDL` ou `GTK+`, proposer une version de 2048 avec une interface graphique. ($\sim$ 2pt)

(3.4) Vous en voulez encore ? Une application *android* ou *iOS* pour votre portable, c'est possible ? ($\sim$ 2pt)

La note maximale est sûrement de 20, mais des points se cachent encore dans la soutenance de votre projet. Ceux qui atteignent le dernier niveau assurent (presque) le 20, mais tout n'est pas perdu pour les autres. Soutenance bien préparée et travail soigné seront récompensés !

## 4. Pour participer au tournois d'IA

Cette section ne concerne que les groupes ayant décidé d'implanter un joueur automatique à leur jeu. Afin de faire concourir vos programmes sur un pied d'égalité, nous utiliserons notre implantation du jeu. Votre programme (le joueur automatique) se contentera de lire un fichier `configuration.txt` contenant la configuration du jeu à une itération donnée k, et écrira dans un autre fichier `mouvements.txt` le mouvement à effectuer pour la configuration à l'itération k.

Notre programme se chargera de lire le mouvement choisi par votre programme et de mettre à jour le fichier `configuration.txt` contenant la nouvelle position de jeu.

Voici les consignes à respecter.

- Le programme principal de votre joueur automatique devra s'appeler `2048_IA.cpp` et votre projet devra documenter clairement la commande de compilation. Idéalement, le programme sera accompagné d'un makefile, dont l'utilisation sera documenté dans un fichier `README.md` ou dans le rapport.
- La compilation de votre programme `2048_IA.cpp` ne doit pas faire intervenir de librairie externe qu'il faudrait installer (pas de librairies graphiques par exemple). Votre commande de compilation doit fonctionner dans le terminal de votre serveur jupyterhub.
- Le fichier `configuration.txt` sera écrit par notre programme. Votre joueur automatique devra le lire et en extraire la configuration du jeu à l'itération k. Le fichier se présente sous la forme suivante :

```
0 0
0,0,0,0;
2,0,0,0;
0,2,0,0;
0,0,0,0;
```

2. Vous trouverez un tutoriel à cette adresse : `https://openclassrooms.com/courses/gerez-vos-codes-source-avec-git`

La première ligne contient le numéro de l'itération (k) puis le nombre de points (n). Les autres lignes représentent le plateau de jeu. Ici c'est l'état initial, donc k et n valent 0 (ligne 1) et seulement deux tuiles sont placées. Après 6 mouvements, le fichier `configuration.txt` sera actualisé de la manière suivante :

```
6 20
0,0,0,0;
0,2,0,0;
2,0,0,0;
8,4,0,0;
```

Le joueur automatique a marqué 20 points.

— Votre programme devra attendre que le nouveau fichier `configuration.txt` soit actualisé avant de lire la configuration du plateau. Pour cela, vous pourrez utiliser un compteur permettant de compter le numéro de l'itération k écrit au début du fichier. Votre programme devra attendre que le nombre écrit dans le fichier coïncide avec celui que vous comptez.

— Le fichier de `mouvements.txt` que vous devrez écrire commencera par un trigramme (trois lettres) faisant office de pseudonyme pour votre joueur automatique. Ce trigramme sera écrit uniquement sur la première ligne. Puis, les lignes suivantes comprendront le numéro de l'itération k et le mouvement correspondant (H, B, G et D pour Haut, Bas, Gauche et Droite). Les lignes seront ajoutées au fur et à mesure de l'exécution de votre programme. Par exemple, à la 5ème itération, le programme joueur automatique BOB aura lu 5 fois le fichier `configuration.txt` et aura écrit le fichier `mouvements.txt` suivant :

```
BOB
0 D
1 B
2 G
3 B
4 B
```

— Si un mouvement est invalide (non reconnu ou bien ne changeant pas l'état du jeu), alors le numéro de l'itération k est incrémenté sans changer le plateau de jeu ou le score.

— Dans le dossier du projet, vous trouverez un notebook `tests_tournois.ipynb`. Suivez les instructions pour exécuter un exemple de simulation et puis tester si les entrées sortie de votre programme sont conformes aux contraintes du tournois listées ci-dessus.

Toutes les améliorations de votre IA sont bonnes à prendre. On peut commencer par une IA qui essaie de ne pas perdre, qui marque les points quand c'est possible ou bien qui anticipe plusieurs actions pour combiner un maximum de tuiles ! Arriver dans le haut du podium en fin de semestre vous garantira quelques points bien mérités... Bonne chance !