

Evaluation du groupe :
Simon LEJEUNE, Yannis BALTUS,
Pierre Louis MARTIN, Alexis RICHARD
Projet 2048

Guillaume ALMYRE	Gaëtan CHAMBRES
Allan MAHAZOASY	Chrystelle PETUREAU

30/03/2015

Point de vue de l'utilisateur :

Question 1 :

Il y a énormément de fichiers. Je pense que même en utilisant cmake (qui a besoin de plus de fichier qu'un Makefile), il y en a beaucoup trop ici et certains fichiers auraient pu être factorisés en un seul. L'archive aurait gagné en lisibilité. De plus, cette abondance de fichier souligne pour moi une mauvaise connaissance de cmake.

On note aussi que la présence du fichier CmakeCache.txt bloque la compilation. L'archive n'aurait pas du la contenir.

Enfin, on constate la présence de control-grid.c qui est en fait le fichier pour terminer le jeu. (un début d'intelligence artificiel?)

FIGURE 1 – Erreur lors de la compilation en laissant CmakeCache.txt

```
petur001@aquarium:~/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD$ cmake .
CMake Error: The current CMakeCache.txt directory /net/cremi/petur001/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD/CMakeCache.txt is different than the d
LEJEUNE_MARTIN_RICHARD where CMakeCache.txt was created. This may result in binaries being created in the wrong place. If you are not sure, reedit the CMakeCa
petur001@aquarium:~/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD$ make
CMake Error: The source directory "/home/yannis/A1_BALTUS_LEJEUNE_MARTIN_RICHARD" does not exist.
Specify --help for usage, or press the help button on the CMake GUI.
Makefile:323: recipe for target 'cmake_check_build_system' failed
make: *** [cmake_check_build_system] Error 1
```

Question 2 :

Après avoir supprimer cmakeCache.txt et fait un "make clean" et il n'y a plus de soucis de compilation.

FIGURE 2 – Compilation

```
petur001@aquarium:~/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD$ cmake .
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /net/cremi/petur001/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD
petur001@aquarium:~/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD$ make
Scanning dependencies of target grid
[ 12%] Building C object CMakeFiles/grid.dir/grid.c.o
[ 25%] Building C object CMakeFiles/grid.dir/fonctions-sup.c.o
Linking C static library libgrid.a
[ 25%] Built target grid
Scanning dependencies of target 2048
[ 37%] Building C object CMakeFiles/2048.dir/control-grid.c.o
[ 50%] Building C object CMakeFiles/2048.dir/grid.c.o
Linking C executable 2048
[ 50%] Built target 2048
Scanning dependencies of target testFct1
[ 62%] Building C object CMakeFiles/testFct1.dir/test/test.c.o
[ 75%] Building C object CMakeFiles/testFct1.dir/grid.c.o
Linking C executable testFct1
[ 75%] Built target testFct1
Scanning dependencies of target testFct2
[ 87%] Building C object CMakeFiles/testFct2.dir/test/test2.c.o
[100%] Building C object CMakeFiles/testFct2.dir/grid.c.o
Linking C executable testFct2
[100%] Built target testFct2
```

Question 3 :

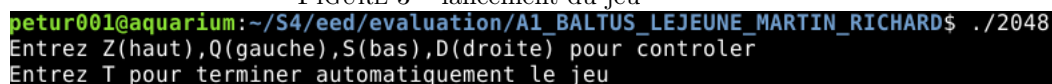
Le fichier exécutable est dans le dossier principal avec le nom 2048.

Il n'y a pas de fichier "readme.txt" qui m'aurait permis de comprendre les fichiers en plus. (fonction-sup.h, fonction-sup.c, control-grid, et les executables testFct1 et testFct2).

Un rappel des commandes "cmake ." "make" "makeclean" aurai été un plus.

Enfin, au lancement du jeu, on nous demande une direction sans voir la grille cela ne me semble pas être un bon choix. Mais le jeu se lance sans erreur.

FIGURE 3 – lancement du jeu



```
petur001@aquarium:~/S4/eed/evaluation/A1_BALTUS_LEJEUNE_MARTIN_RICHARD$ ./2048
Entrez Z(haut),Q(gauche),S(bas),D(droite) pour controler
Entrez T pour terminer automatiquement le jeu
```

Point de vue fonctionnel :

Question 4 :

L'option t(terminer) est assez étrange, je n'en comprends pas vraiment l'intérêt hormis pour les tests ou l'intelligence artificielle. Mais, ce n'est pas, en soit, un bogue.

Les tuiles fusionnent de façon cohérente et le calcul du score est bon.

Question 5 :

La bibliothèque libgrid.a doit être créée avec grid.c. Or, dans ce code, il faut grid.c + fonctions-sup.h + fonctions-sup.c pour créer la bibliothèque libgrid.a.

La fonction valeur est définie dans fonction-sup.c et sert pour la fonction show-grid définie dans le même fichier.

L'autre erreur (sur le scann écran figure 4) est surtout lié à la différence de nom de fonction show-grid pour ce groupe , afficher pour notre groupe.

FIGURE 4 – Erreurs

```

grid.o: dans la fonction « do_move »:
/net/cremi/petur001/S4/eed/testprojet/grid.c:297: référence indéfinie vers « valeur »
grid.o: dans la fonction « play »:
/net/cremi/petur001/S4/eed/testprojet/grid.c:377: référence indéfinie vers « show_grid »
collect2: erreur: ld a retourné 1 code d'état d'exécution

```

FIGURE 5 – Titles de 3

```

+-----+-----+-----+-----+
|       |       |       |       |
+-----+-----+-----+-----+
|       |  1  |       |  2  |
+-----+-----+-----+-----+
|       |       |       |  3  |
+-----+-----+-----+-----+
|       |       |  3  |  4  |
+-----+-----+-----+-----+
Votre score est de 42 points

```

En copiant le code de show-grid et valeur dans mon fichier jeu, cela crée une grille jouable même si elle s'affiche de base avec 3 tuiles remplies. Cependant, il y a un problème lié à l'utilisation ou non des puissances de 2. J'obtiens des tuiles de 1 ou de 3. Mais quand elles fusionnent, elles augmentent le score de 8 points!

Question 6 :

L'utilisation de valgrind ne révèle aucune fuite de mémoire.

FIGURE 6 – Début du programme

```

==5391== Memcheck, a memory error detector
==5391== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==5391== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==5391== Command: ./2048
==5391==

```

FIGURE 7 – Après avoir jouer

```

==4974==
==4974== HEAP SUMMARY:
==4974==   in use at exit: 0 bytes in 0 blocks
==4974==   total heap usage: 6 allocs, 6 frees, 120 bytes allocated
==4974==
==4974== All heap blocks were freed -- no leaks are possible
==4974==
==4974== For counts of detected and suppressed errors, rerun with: -v
==4974== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Point de vue du code :

Question 7 :

Dans le fichier grid.c, l'implémentation de can-move est assez aléatoire.

Dans le fichier fonction-sup.c :

availMoves : problème sur l'accolade finale.

show-grid : les commentaires à la fin du fichier ne sont pas alignés. Une ligne de code est commentée (un oubli?).

valeur : les 3 if ne sont pas indentés de la même façon.

Dans le fichier control-grid.c, les 2 while ne sont pas indentés de la même façon. Les normes d'indentation pour test.c et test2.c ne sont pas du tout les mêmes. Les commentaires sont pratiquement absents du code.

Globalement, il n'y a aucune unité dans l'indentation, la forme des commentaires ou le placement des accolades ouvrantes ou fermantes (début ou fin de ligne).

Question 8 :

Dans le fichier grid.c, dans la fonction can-move, les variables loni/sup et Tj/Tk ne sont pas claires.

Les variables loni et sup deviennent horizontal et sup dans do-move. Ce qui n'aide pas à la lisibilité du code. Surtout qu'on parle de loni (au lieu de horizontal) dans les commentaires de do-move. (grid.c, ligne 243)

Ce genre de ligne de commentaire ne devrait pas être présent "ln = ligne, cl = colonne" (grid.c, ligne 331). Pourquoi ne pas mettre directement les variables ligne et colonne ?

On note cependant que les nouvelles fonctions (non demandées par le .h) ont des noms cohérents (set-score, get-free-tiles, set-free-tiles, show-grid, valeur).

Question 9 :

On note d'abord qu'il n'y a pas d'unité dans les commentaires notamment dans les titres. Parfois, c'est le "on" qui est utilisé parfois le "nous" (grid.c ligne 333 et ligne 347)

Dans grid.c, une balise "/param" est mise en début de fonction mais la définition de ce paramètre est rarement utile. Par exemple, (grid.c ligne 363) un simple "dir : direction" n'est pas très explicite à mon avis. "direction choisie par l'utilisateur" ou "direction testée" aurai eu plus d'intérêt, je pense.

À la ligne 253, un mot n'est pas français "lico".

À la ligne 273, un code équivalent mais commenté est proposé pour comprendre le code de l'implémentation. Je pense que le code commenté aurai

été plus clair, donc pourquoi ne pas le mettre directement ?
 La plupart des fonctions ont en titre le nom de leurs "créateurs". Dans un projet réel cela peut être pratique pour demander à telle équipe d'expliquer le code. Dans notre situation, un projet fait à 4 personnes sans distinction, je ne sais pas si c'est réellement adapté.

Question 10 :

Ce code utilise les puissances de 2 pour les valeurs des tuiles ce qui est une très bonne chose. Cependant, l'utilisation d'une constante et non l'inscription en dur de "2" aurait permis une évolution rapide du code avec par exemple des puissances de 3.

Question 11 :

Le code n'est absolument pas dupliqué, bien au contraire. La fonction can-move (dans le fichier grid.c) est tellement condensée que sa compréhension en devient très fastidieuse.

Question 12 :

Le fichier testFct1 (créé à partir de test.c) test la fonction can-move en affichant les possibilités de déplacement. Seules les touches l,d,u,r et s(pour sortir) sont autorisées mais certaines touches comme a ou z ont des actions inattendus. On note cependant la prise en compte des majuscules/minuscules.

FIGURE 8 – comportement inattendu des tests automatiques

```
a
can_move:test RIGHT-----OK
can_move:test LEFT-----OK
can_move:test UP-----NOK
can_move:test DOWN-----OK
can_move:test RIGHT-----OK
can_move:test LEFT-----OK
can_move:test UP-----NOK
can_move:test DOWN-----OK
z
can_move:test RIGHT-----OK
can_move:test LEFT-----OK
can_move:test UP-----NOK
can_move:test DOWN-----OK
can_move:test RIGHT-----OK
can_move:test LEFT-----OK
can_move:test UP-----NOK
can_move:test DOWN-----OK
q
can_move:test RIGHT-----OK
can_move:test LEFT-----OK
can_move:test UP-----NOK
can_move:test DOWN-----OK
can_move:test RIGHT-----OK
can_move:test LEFT-----OK
can_move:test UP-----NOK
can_move:test DOWN-----OK
```

Le fichier testFct2 (créé à partir de test2.c) test surtout la fonction do-move avec quelques mouvements pré-définis. On sent que les mouvements

ont été choisis de façon précise et en dur. Il n'y a aucune place au hasard donc j'en vois peu l'intérêt. Le but des tests est de trouver les bogues potentiels, pas de montrer qu'on a testé un cas particulier. De plus, il y a très peu d'explication donc je ne comprends pas certains choix comme entre les lignes 55 et 70.

Question 13 :

La compilation ne donne pas d'erreur. La grille s'affiche avec la nouvelle taille sans aucun problème d'affichage. Cependant, l'exécutable `testFct2` n'est plus fonctionnel du coup. On note également que le `t` (terminer) est vraiment très long si on donne une taille un peu grande. Un "quitter" aurait été plus pratique à mon avis.