# Evaluation du groupe : Ugo SCHIAPPARELLI, Dylan BEDIN, Nicolas ETCHEVERRY

## Projet 2048

Guillaume ALMYRE Gaëtan CHAMBRES
Allan MAHAZOASY Chrystelle PETUREAU

30/03/2015

# Point de vue de l'utilisateur :

## Question 1 :

L'archive n'est pas trop surchargée mais il n'y a pas de séparation en différents dossiers, à l'exception des dossiers de l'interface graphique, que nous n'évaluons pas ici. Cependant il n'y a pas de fichiers superflux autres que ceux de l'interface graphique.

## Question 2 :

La compilation s'exécute sans aucune erreur et sans warning, cependant aucune bibliothèque n'est créée. Les executables sont générés correctement.

## Question 3 :

Présence d'un README.txt qui explique comment compiler et executer les différentes versions du projet. Le programme se lance sans difficulté, et le nom de l'exécutable est précisé dans le README.

# Point de vue fonctionnel :

## Question 4 :

Les règles du jeu sont respectées. Cependant il n'y a jamais d'affichage du score et on peut effectuer plusieurs déplacements à la fois. Par exemple en écrivant QD puis en validant, on effectuera un mouvement gauche, droite instantanément. L'absence de grille visible est à noter.

## Question 5 :

Nous n'avons pas trouvé de libgrid.a dans l'archive, même après un make.

## Question 6 :

Après utilisation de Valgrind on obtient : "total heap usage : 10 allocs, 2 frees, 313 bytes allocated" Une partie de la mémoire n'est donc pas libérée.

# Point de vue de l'utilisateur :

## Question 7 :

Une partie des fonctions est parfaitement indentée. Cependant, dans de nombreux blocs, (en particulier au niveau des boucles for), l'indentation est incorrecte, comme montré sur le screenshot suivant. Il y a aussi certains blocs très

```
for (int x = 0 ; x < GRID_SIDE ; ++x){
  for (int y = 0 ; y < GRID_SIDE ; ++y){
    if ((x + incX) < GRID_SIDE && (y + incY) < GRID_
  if (get_tile(g, x, y)==0)
    {
      if (get_tile(g, x + incX, y + incY) != 0)
        return true;
    }
```

FIGURE 1 – Problème d'indentation

compacts ce qui rend la lecture difficile, un saut de ligne aurait aéré le bloc, comme montré sur l'exemple qui suit. Le code reste cependant lisible.

```
grid new_grid(void){
  grid g = malloc(sizeof(*g));
  assert(g != NULL);
  g->tab = malloc(GRID_SIDE * sizeof(tile*));
  assert(g->tab);
  for(int i = 0; i < GRID_SIDE; i++){
    g->tab[i] = malloc(GRID_SIDE * sizeof(tile*));
    assert(g->tab[i]!= NULL);
    for (int j = 0; j < GRID_SIDE; j++)
      g->tab[i][j]=0;
  }
  g->score = 0;
  return g;
}
```

## Question 8 :

Pour les coordonnées des colonnes, x et y sont assez pertinents. Dans do_move et can_move, les variables incX et incY peuvent prêter à confusion, mais une ligne de commentaire explique leur sens. Le nom des fonctions est assez explicite.

## Question 9 :

Certaines fonctions ne sont pas du tout commentées. En règle générale, un commentaire explique le but d'une fonction ou d'un bloc, mais rarement son fonctionnement. Certains comentaires sont de "type Oxygen" avec le "brief" et certains non. Le fichier contenant le main ne possède aucun commentaire.

### Question 10 :

Il n'y a aucune constante non abstraite, le code utilise les constantes définies dès que possible.

### Question 11 :

Nous n'avons pas trouvé de duplication de code. Dans l'ensemble du projet, il y a de nombreuses fonctions qui prennent en charge les différentes actions requises dans le programme.

### Question 12 :

Les fonctions de base, définies dans le .h sont testées. Les fonctions rajoutées par le groupe sont utilisées lors des tests et donc testées "par défaut". Au moins 90% du code a l'air d'être testé.

### Question 13 :

La modification de GRID_SIDE, n'influe pas sur la stabilité et la jouabilité du programme. On ne peut cependant rien constater sur la modifications de l'affichage puisque la grille n'est pas complètement dessinée.