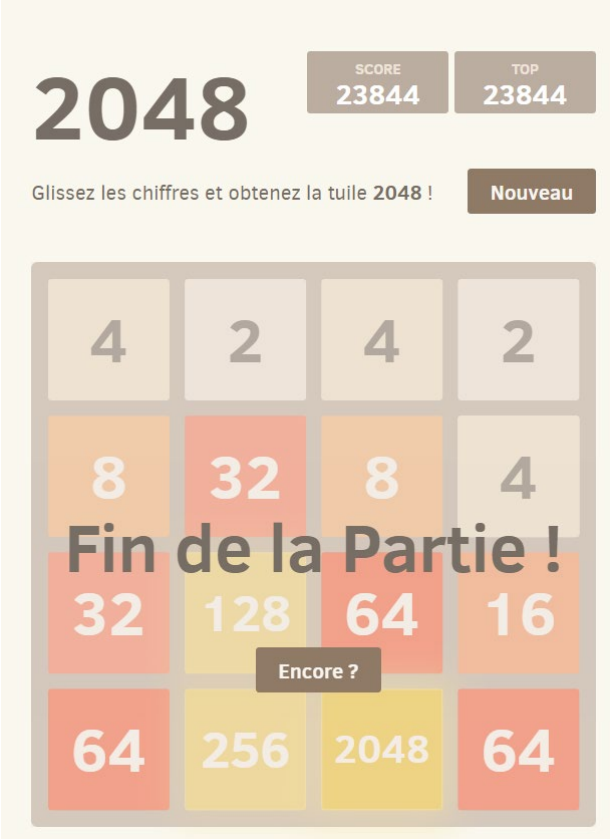
MA-20 Jeu programmé (2048)
(Pour C#, ou Python)

Introduction au jeu 2048

Sur le site 2048.fr, on trouvera en ligne le jeu qu'on se propose de programmer. Sur un ordinateur il se joue avec les flèches de direction (ou les touches asdw). A chaque déplacement on « tasse » les tuiles dans une direction. Lorsque 2 tuiles identiques collisionnent (exemple 2 tuiles de 8), elles fusionnent pour faire une tuile de niveau supérieur (exemple 16).

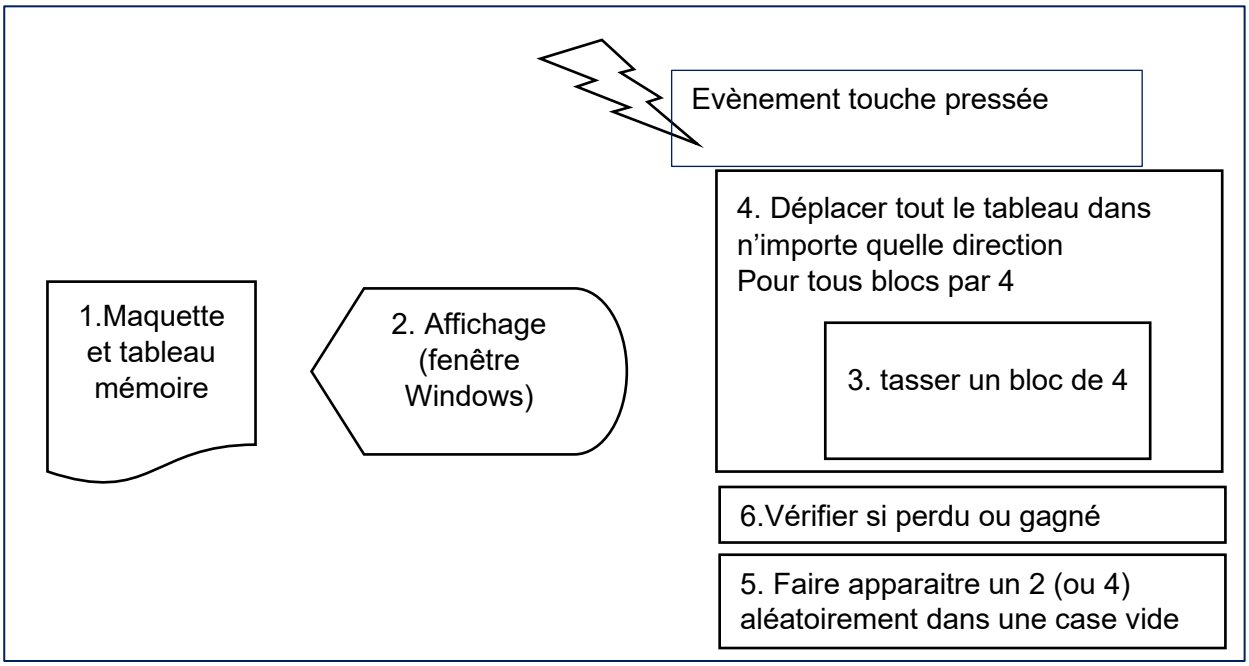
A chaque déplacement une nouvelle tuile (de 2 ou 4) apparait.

Sur les copies d'écrans ci-dessous on voit à gauche une partie en cours, et à droite une partie terminée (il n'est plus possible de jouer).



Proposition de découpage

Pour réaliser un programme d'une certaine importance, on doit le découper en parties, de manière à pouvoir réaliser, tester et livrer en un temps raisonnable (par ex. 4h ou 8h suivant la finesse de découpage). Ci-dessous une proposition de découper en 6 parties, à chaque fois avec un livrable qui peut être testé :



1. Etape 1. Maquette et « tableau » mémoire

A la fin de cette étape, on aura créé :

- Une maquette du jeu (fenêtre Windows) en utilisant des objets connus (labels, contrôles texte, boutons…) et la description des actions. La maquette montrera les couleurs de toutes les valeurs (de 2 à 8192 et aussi des cases vides). Les cases ne sont pas forcément carrées.
- Un code avec deux exemples de tableaux/listes en mémoire (ou autre structure pour mémoriser l'état du jeu)
  - Au début du jeu (quasi vide, avec 2 tuiles montrant un 2)
  - Au milieu du jeu (par exemple comme le jeu en cours à la page précédente.

Réflexion : on peut stocker les tuiles sous forme de nombres (2,4,8,16,…8192) ou stocker la puissance de deux (0,1,2,3,4,…13). Pour l'affichage des couleurs par exemple, si on choisit la deuxième solution, bien qu'en python on pourra utiliser un dictionnaire.

2. Etape 2. Affichage d'un tableau mémoire

A la fin de cette étape, le programme sera capable d'afficher le jeu conformément au tableau/liste en mémoire. Pour que cette étape soit validée, on remplira un tableau avec des valeurs fixes pour que l'affichage puisse montrer :

- Des tuiles de toutes les valeurs de 2 à 8192 (=13 cases, de 2^1 é 2^{13}) avec leur couleur.
- 3 tuiles vides.

3. Etape 3. Tasser un bloc de 4

A la fin de cette étape, on livrera une fonction qui reçoit 4 valeurs, et renvoie un tableau/liste de 4 valeurs « tassées » vers le début du tableau. Avec la possibilité de la tester (par exemple par menu)

test : vérifier avec plusieurs états du tableau, que le déplacement est conforme au jeu.

Avant tassement	Après tassement
0/0/0/2	2/0/0/0
0/0/2/2	4/0/0/0
2/0/2/2	4/2/0/0
2/2/2/2	4/4/0/0
2/2/4/0	4/4/0/0

La fonction dira aussi combien de déplacements elle a dû faire.

4. Etape 4. Tasser tout le jeu dans une direction

- A la fin de cette étape, on livrera un programme qui, à chaque pression sur la touche correspondante (asdw ou les flèches), tasse tout le tableau dans la bonne direction et le réaffiche. Elle dira aussi combien de déplacements ont eu lieu.

5. Etape 5. Apparition aléatoire de tuiles de 2 ou de 4

A la fin de cette étape, on livrera un programme qui, après chaque tassement de tout le tableau (si un déplacement a vraiment eu lieu), fait apparaître aléatoirement un 2 ou un 4 dans une case encore vide, si c'est possible.

6. Etape 6. Tests de fin de jeu (perdu/gagné)

A la fin de cette étape, on livrera un programme qui vérifie si on a gagné ou perdu :

- après apparition d'une nouvelle tuile, si le tableau est plein et qu'on ne trouve plus 2 cases identiques contigües, alors on a perdu. Dans ce cas, on finit le jeu en affichant un message de fin de jeu, et en ne donnant plus la possibilité de continuer de déplacer. Mais on doit voir encore le jeu.
- Après fusion de 2 tuiles, si on vient de faire 2048 (et qu'un autre 2048 ou supérieur n'existe pas dans la grille), on signale que le joueur vient de gagner tout en lui laissant la possibilité de continuer

7. Options et personnalisation (à choisir)

a. Menu « Recommencer »

Un menu « Fichier/Recommencer » permet de recommencer une partie sans quitter

b. Menu « Quitter »

Un menu « Fichier/Quitter » permet de recommencer une partie sans quitter

c. Afficher le « score »

Pour cette option, le programme mettra à jour, à chaque fusion de 2 tuiles, un score augmenté de la valeur de la tuile fusionnée (si 2 tuiles 4 font un 8, on ajoute 8 au score).

d. Meilleur score

A la fin de la partie, le programme doit écrire sur le disque le score. Et de cette manière on pourra indiquer le meilleur score réalisé sur cet ordinateur.

e. Pseudo

Dans ce cas il faut donner son pseudo en début de partie, et lors de la fin d'une partie (si on quitte précipitamment ou si on perd), le programme écrit le score associé au pseudo. De cette manière le « meilleur score » indiqué est propre au pseudo

f. Fonction « Annuler le déplacement »

Par un menu « Fichier/Annuler », on aimerait permettre à l'utilisateur d'annuler le dernier déplacement (un seul retour en arrière possible)

g. Personnalisation

Il est possible de proposer d'autres personnalisations du programme.

Délais de fin :

Sprint	Date de fin	Livrable
Sprint 1	Semaine 2 : 30 j / 2 fév	v0.1 Etape 1 + 2
Sprint 2	Semaine 4 : 20 / 23 février	v0.2 Etape 3 + 4
Sprint 3	Semaine 6 : 05 / 08 mars	V0.3 Etape 5 + 6
Sprint 4	Semaine 8 : 19 / 22 mars	V 1.0 Rendu final (avec options – personnalisation…)
Défense	Semaine 9 : 26 / 27 mars	Exposé, démonstration publique. (2h pour la classe b)