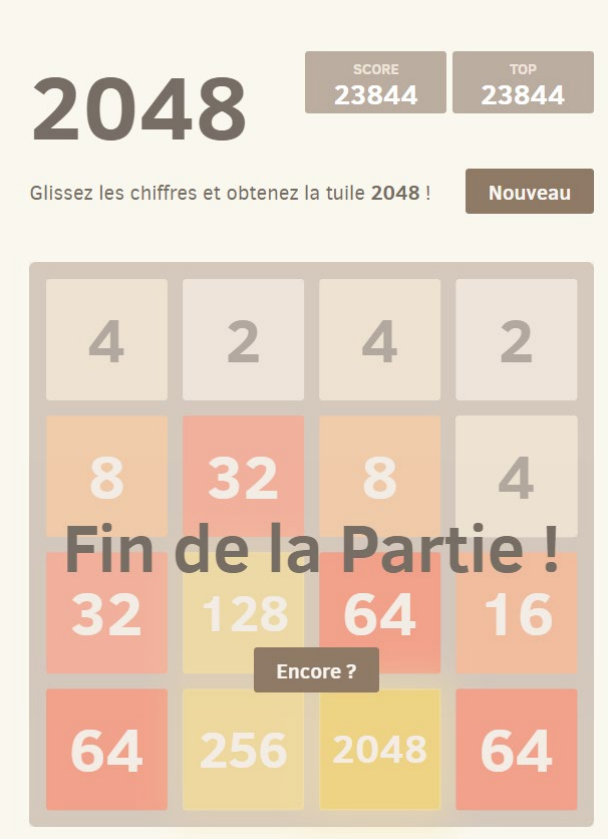
MA-20 Jeu programmé (2048)
(Pour Python)

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).

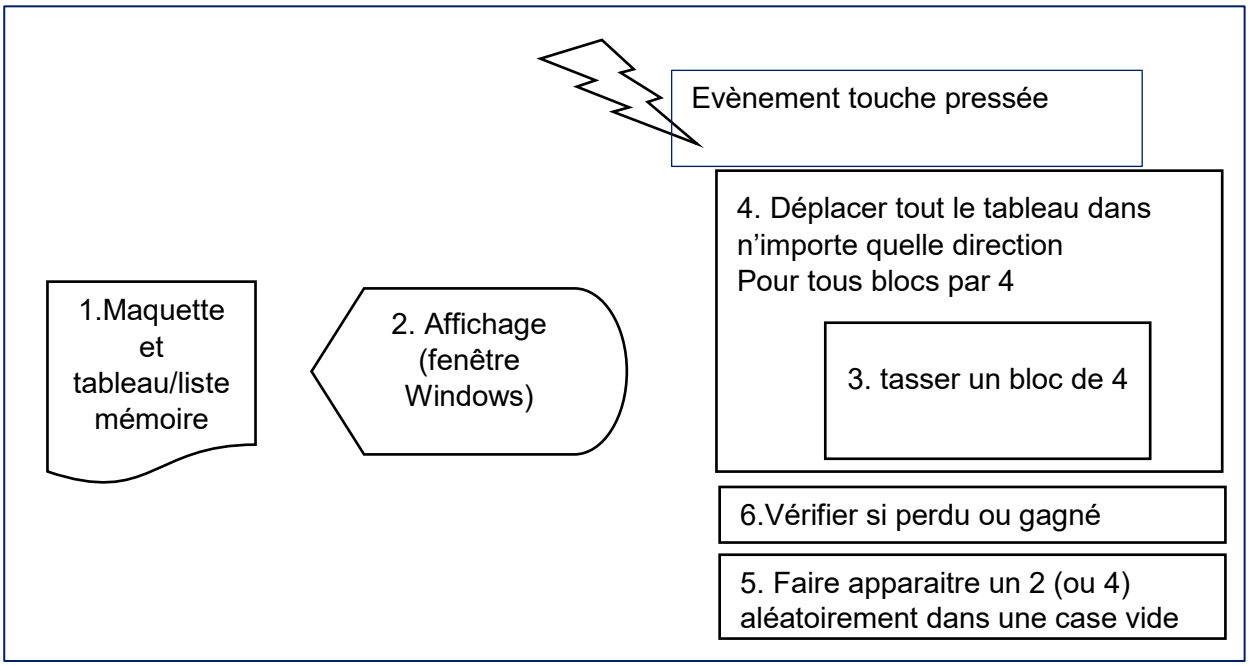
A chaque déplacement, lorsque les tuiles bougent vraiment, une nouvelle tuile (de 2 ou 4) apparait.

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…).

- Un code avec deux exemples de tableaux/listes en mémoire
  - Au début du jeu (quasi vide, avec 2 tuiles montrant un 2)
  - Contenant toutes les valeurs possibles comme si un jeu avait toutes les tuiles de 2 à 8192 et quelques tuiles vides.

Réflexion : on peut stocker les tuiles sous forme de nombres (2,4,8,16,…2048) ou stocker la puissance de deux (0,1,2,3,4,…11). Pour l'affichage des couleurs par exemple, la deuxième solution est peut-être plus simple.

2. Etape 2. Affichage d'un tableau/liste mémoire

A la fin de cette étape, le programme sera capable d'afficher le jeu conformément au tableau en mémoire et conformément à la maquette. 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 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	Nb déplacements
0/0/0/2	2/0/0/0	3
0/0/2/2	4/0/0/0	3
2/0/2/2	4/2/0/0	2
2/2/2/2	4/4/0/0	2
2/2/4/0	4/4/0/0	1
2/0/0/0	2/0/0/0	0
0/0/0/0	0/0/0/0	0

La fonction dira aussi (en 5° paramètre par exemple) combien de déplacements elle a dû faire. On peut compter 1 déplacement par 0 comblé (vers la gauche) et 1 déplacement par fusion.

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 (probabilité équivalente ou proche du jeu réel) dans une case encore vide, s'il en reste.

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

Délais de fin :

Sprint	Date de fin	Livrable
Sprint 1	Semaine 2 : 10 février 2023	v0.1 Etape 1 + 2 1 maquette (Balsamiq) Code 2048_v0.1 dans le disque perso.
Sprint 2	Semaine 5 : 10 mars 2023	v0.2 Etape 3 + 4 Code 2048_v0.2 dans le perso Les 2 premiers sprints dans icescrum
Sprint 3	Semaine 7 : 24 mars 2023	V0.3 Etape 5 + 6 Code 2048_v0.3 dans un github Les 3 sprints dans icescrum
Défense	Semaine 8 + 9 : 30-31 mars et 6 avril (pas de vendredi)	Démonstration devant la classe

Attention: toute aide autre que celle de votre enseignant du module MA-20 doit être mentionnée de manière précise dans les commentaires du code, y compris celle de vos camarades de classe ou de personnes extérieures à l'école.