Rapport de projet AP1 : 2048

HANNA Emile-Malek
PROUVOST Laureline
Peip A groupe 13
2016-2017

Sommaire :

1. But du projet :

2048 est un jeu de combinaisons de tuiles comprenant des puissances de 2 positives supérieures à 1 dans le but de créer une tuile portant le nombre 2048. Dans le contexte de ce projet, nous nous limiterons à une grille carrée de taille 4x4 tuiles. Pour cela, il suffit à l'aide des différents mouvements qui nous sont proposés (droite, gauche, haut, bas) de fusionner les tuiles de valeur égale en les additionnant. Celles-ci se déplacent dans la direction donnée jusqu'à rencontrer le bord du plateau ou une tuile de la même valeur. De plus, à chaque mouvement, une nouvelle tuile de valeur 2 ou 4 apparaît de manière aléatoire.

2. Procédure utilisée :

Afin de réussir la programmation du jeu, nous avons créé plusieurs fonctions qui, les unes à l'aide des autres, nous ont permis d'aboutir à sa réalisation.

Tout au long de ce projet, nous assimilons la grille a une liste de listes. Chaque liste correspondant à une ligne, Ainsi l'élément grid [i] [j] correspond à l'élément le ligne i et de colonne j.

Les différentes étapes du projet sont:

- Définir une fonction donnant les différentes cases vides de la grille
- Définir la fonction donnant la valeur de la nouvelle case à ajouter
- Définir une fonction donnant la grille à laquelle on a rajouté une nouvelle case
- Définir une grille initiale aléatoirement
- Définir une fonction imprimant la grille
- Définir une fonction donnant la valeur maximale des tuiles présentes dans la grille
- Définir une fonction qui effectue un mouvement de toutes les cases dans une direction souhaitée
- Définir une fonction indiquant si la partie est terminée

Définir une fonction donnant les différentes cases vides (get_new_position):

Get_new_position est une fonction renvoyant les coordonnées de toutes les cases vides de la grille de jeu. Pour cela, nous parcourons chaque case de la grille passée en paramètre, puis, nous ajoutons dans une liste annexe initialement vide, les coordonnées i,j sous forme de liste, de toutes les cases contenant la valeur 0.

Définir une fonction donnant la valeur de la case à rajouter (get_new_tile):

Le jeu original ajoutant de manière aléatoire un 2 ou un 4 après chaque mouvement, nous avons désiré qu'il apparaisse de manière aléatoire avec une probabilité de 80% d'obtenir un 2 et de 20% d'obtenir un 4.

Pour cela, à l'aide de la fonction randint de python donnant un entier aléatoirement, nous avons cherché à obtenir des valeurs d'entiers aléatoirement entre 0 et 100. Ainsi, si la valeur obtenue était comprise entre 0 et 80, la valeur renvoyée était un deux et dans le cas contraire, un 4.

Définir une fonction donnant la grille à laquelle on a rajouté la nouvelle case (grid_add_new_tile):

Grid_add_new_tile est fonction prend en paramètre une grille et renvoie une nouvelle grille dans laquelle elle ajoute une case différente de 0 à la place d'une case contenant la valeur 0.

Pour cela, elle utilise la fonction get_new_position précédemment définie en lui passant en paramètre la grille passée elle aussi en paramètre. Après avoir récupéré la liste de doublets de coordonnées, on en choisit une aléatoirement dans laquelle on implémente la valeur choisie grâce à la fonction get_new_tile.

Définir une grille initiale aléatoirement (grid init):

Grid_init est une fonction qui renvoie une grille qui sera considérée comme la grille initiale. C'est à dire constituée uniquement de 0 à l'exception d'une seule case, choisie aléatoirement, et qui prendra aléatoirement la valeur de 2 ou de 4.

Pour cela, on fait référence à la fonction grid_add_new tile définie précédemment en passant en paramètre une grille ne contenant que des 0.

Définir une fonction imprimant la grille (grid_print):

Grid_print est une fonction qui imprime une grille passée en paramètre. C'est à dire que les lignes sont séparées par un enchaînement de 20 "-" et les colonnes sont séparées par des "|", offrant un visuel de grille constituée de cases .

Pour cela , nous avons commencé par définir deux constantes qui sont tiret et colonne. Tiret correspond à l'enchaînement de 20 "-" alors que colonne correspond au caractère "|".

Nous avons alors successivement imprimé la constante tiret puis la ligne de jeu appelée colo_imp. Nous avons obtenu colo_imp en alternant successivement le caractère colonne et la valeur de la case que nous avons centrée grâce à la méthode format centrant la valeur dans un espace de 4 caractères.

Nous avons alors pu distinguer deux cas : celui où la valeur était différente de 0, où la valeur est affichée normalement et celui où elle était égale à 0. Dans ce cas, pour permettre une meilleure visibilité de la grille, il était inutile d'afficher un 0, nous avons donc remplacé la valeur de la case par des espaces à la variable colo_imp avant d'ajouter la colonne.

Définir une fonction donnant la valeur maximale présente dans la grille(grid_get_max_value):

Grid_get_max_value est une fonction qui parcourt la grille passée en paramètre et qui en renvoie la plus grande plus grande valeur présente dans celle ci.

Pour ce faire, on définit une variable appelée max qu'on initialise à 0. Ensuite on parcourt la liste, si la valeur de la case est supérieure à max, alors max prend la valeur de la case évaluée.

Définir une fonction qui effectue un mouvement de toutes les cases dans une direction souhaitée (grid_move).

Cette fonction effectue un décalage des cases de la grille différentes de 0 dans une direction demandée, et additionnant les cases qui ont la même valeur.

Pour ce faire, on fait appel à une fonction annexe nommée mvt.

La fonction grid_move donne alors en paramètre à la fonction mvt d'une part, le score de la grille et d'autre part, la ligne ou la colonne sur laquelle on souhaite effectuer le mouvement de droite à gauche. Dans cette liste, nous ne donnons que les valeurs différentes de 0 pour simplifier le traitement (les valeurs manquantes seront ajoutées dans la fonction mvt). Ainsi, pour mouvement left, on lui donne chaque ligne à analyser sans effectuer de modification. Pour le mouvement right, on inverse le sens de la liste qui représente la ligne. Pour le mouvement up, on met dans une liste les éléments de chaque colonne de haut en bas. Enfin, pour le mouvement down, on procède comme pour le mouvement up en inversant la liste au dernier moment. Puis, on ajoute au fur et à mesure dans la nouvelle grille appelée newgrid les colonnes ou lignes (selon le mouvement) dont le mouvement a été effectué avec la fonction mvt.

La fonction move est définie comme tel :

Elle prend en paramètre une lsite sur laquelle elle va effectuer un mouvement des cases de droite à gauche, puis elle affiche le score des cases additionnées. Dans un premier temps, on définit une liste vide nommée ligne_finale et une variable score étant le score (cf extension) . Ensuite, la fonction parcourt la ligne (qui correspond à une liste).

On distingue alors deux cas:

- L'élément n évalué et celui d'après sont égaux, on additionnes alors ces deux termes et on les ajoute dans "ligne_finale". On ne passe pas à l'élément suivant mais à l'élément n+2 étant donné que nous avons traité le cas de deux cases
  De plus, le score est incrémenté de la valeur de la tuile ajoutée à "ligne_finale"
- Dans le cas contraire, on ajoute simplement l'élément évalué à la liste "ligne_finale"

Enfin, il ne reste plus qu'à compléter ligne_finale en y ajoutant suffisamment de 0 pour que "ligne_finale" ait une longueur suffisante (soit constituée de 4 éléments).

Définir une fonction indiquant si la partie est terminée(is_grid_over):

Cette fonction évalue si un mouvement peut encore être effectué sur la grille,et qui renvoie la valeur de vérité de la proposition "La grille n'est pas pleine", sous forme d'un booléen (Vrai, Faux)

Pour ce faire, on effectue différents mouvements dans les 4 dimensions. Si la grille reste inchangée pour les 4 mouvements possibles, alors aucun mouvement n'est valide et la fonction renvoie False, autrement, elle renvoie True.

3. Extensions

a. Calcul du score

Nous avons calculé le score de deux manières différentes :

- La première (celle qui est demandée dans le sujet) consiste à effectuer la somme des valeurs des cases présentes dans la grille. Pour cela, on utilise un compteur qu'on initialise à 0 et auquel on ajoute les différentes valeurs des cases au fur et à mesure que l'on parcourt la grille.
  Nous l'appelons score attendu.
- La seconde méthode (celle qui est utilisée dans le jeu original) consiste à calculer le score de manière à ce que ne soient comptabilisées que les cases s'étant additionnées. Pour cela, on ajoute directement la valeur des cases qui s'additionnent quand on effectue le mouvement, c'est à dire dans la fonction mvt. Pour cela, nous avons rajouté un nouveau paramètre appelé score dans les fonctions concernées (grid_move, is_grid_over, mvt)
  Nous l'appelons score jeu.

Enfin, pour communiquer au joueur son score en temps réel, nous avons, dans le fichier textual comme dans le graphical (l'extension graphique), imprimé au fur et à mesure le score de la grille dans l'interpréteur Python.

b. Extension graphique

C'est une extension , qui nous a été fournie et programmée par un professeur de l'université de Lille 1.

Cette extension offre une interface graphique à notre programme, c'est à dire qu'elle donne les valeurs de la grille dans une grille et elle affecte a chaque valeur une couleur de case. Cette extension fait notamment appel à la fonction grid_get_value que nous avons dû définir. C'est une fonction qui, en fonction des paramètres qui lui sont passés, donne la valeur de la case correspondante. Les paramètres étant la grille dans laquelle se trouve la case dont les coordonnées sont les deux autres paramètres.

De plus, nous avons ajouté une condition dans la fonction key_pressed dans le but de vérifier qu'une fois le mouvement effectué, la grille était bien modifiée. Ainsi, si ce n'est pas le cas, aucune case ne vient se rajouter dans une des cases vides.

Nous avons également modifié le code pour que le module graphique affiche dans l'interpréteur le score de la grille comme expliqué ci-dessous.

Cette extension, dit également à l'utilisateur, une fois que sa partie est terminée si il a gagné ou pas.

c. Sauvegarde et chargement de partie

Cette extension permet à l'utilisateur de sauvegarder sa partie à tout moment avec le fichier textual, mais aussi lorsqu'il commence une partie de démarrer avec une grille vierge, ou bien de charger une partie précédemment jouée.

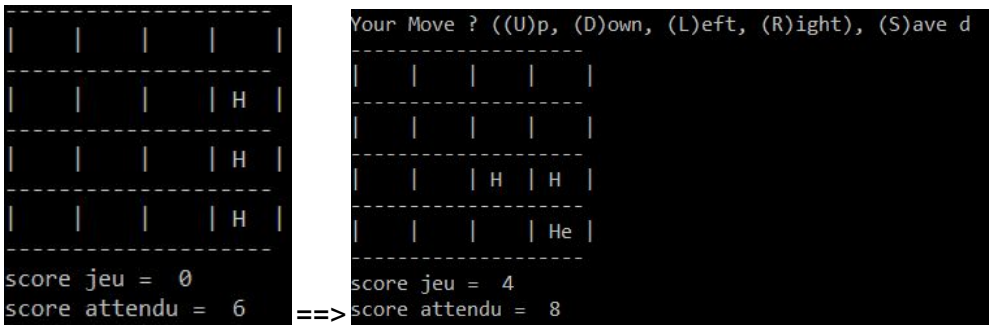
Dans ce but, nous avons créé deux nouvelles fonctions.

Tout d'abord, nous avons permis à l'utilisateur de sauvegarder sa partie grâce à la fonction grid frame. C'est une fonction qui sauvegarde la partie en cours. C'est à dire qu'elle enregistre dans un fichier texte la grille telle que est quand on l'imprime, le score ainsi que le thème de la partie (voir ci dessous pour le thème). Pour cela, nous avons rajouté une option dans les différents mouvements appelée Save qui demande ensuite à l'utilisateur sous quel nom le fichier doit être enregistré. Après avoir effectué cette action, l'utilisateur a le choix de continuer sa partie, on lui rappelle alors la grille ainsi que le score qu'il avait, ou de quitter définitivement la partie. Dans ce cas, nous interrompons le jeu avec un break.

Pour lui permettre de charger une ancienne partie au début du jeu, nous avons créé une nouvelle fonction appelée start donnant à la fonction principale le thème de la grille, la grille initiale ainsi que le score, notamment si on charge une ancienne partie. Pour cela, nous avons ajouté une condition permettant au programme de connaître son choix. Si l'utilisateur décide de charger une ancienne partie, la fonction grid_load va récupérer l'ancienne grille de jeu, le score ainsi que le thème du fichier dont le nom aura été renseigné. Dans le cas contraire, une grille est définie aléatoirement selon le thème que l'utilisateur aura choisi.

d. Thème Chimie

Cette extension inspirée du jeu atomas revient à remplacer les valeurs de la grille par des noms d'atomes (H, He, Li, Be, B, C, N, O, F, Ne …). Ainsi, une tuile de valeur 2^i est l'élément de numéro atomique i.



Pour faire cela, nous avons créé dans un premier temps un dictionnaire qui, à chaque puissance de 2, affecte de manière croissante un élément du tableau périodique. Nous avons ensuite créé une fonction qui imprime une grille passée en paramètre en remplaçant chaque valeur par l'élément chimique lui correspondant.

Par conséquent, nous avons modifié dans le fichier textual la fonction start en ajoutant une option en début de partie qui est d'avoir le choix entre une partie “classique” ou avec le thème “chimie”. Ainsi, nous avons ajouté en fonction de ce choix une condition dans la fonction play qui fait que la grille est imprimée avec des nombres ou des noms d’atomes.

e. Modifications sur textual (non citées ci-dessus)

Dans l’optique d’améliorer le jeu et corriger un maximum de bugs, nous avons effectué plusieurs modifications sur textual en plus de celles citées ci-dessus.

Nous avons alors permis à l’utilisateur, lorsqu’il répond à l’interpréteur d’utiliser des lettres majuscules comme des minuscules. Puis, nous lui avons permis de se tromper de commande (donc de lettre) sans lever d’erreur à l’aide d’une vérification faite par une boucle. La commande est donc demandée une nouvelle fois tant que celle-ci n’est pas connue.

Nous avons également corrigé un bug qui, par exemple, lorsque nous voulions effectuer un mouvement vers la droite alors que ce mouvement n’était pas possible, le considérait comme valide et rajoutait une case aléatoirement dans une des cases vides de la grille. Ainsi, nous avons mis une condition sur la commande de mouvement, c’est à dire que nous vérifions que la grille avant et après le mouvement souhaité n’est pas inchangé. Autrement, le mouvement souhaité est à nouveau demandé tant qu’il n’est pas valide.

f. Grille différente

Dans l’optique de développer notre jeu dans différents formats autre qu’une grille 4x4, nous avions créé au début du projet deux fonctions : sont_coordonnees_valides et est_une_grille. Elles vérifient que les coordonnées de la grille passées en paramètre peuvent lui appartenir et que la grille passée en paramètre est bien valide. Dans la suite du projet, nous n’avons travaillé qu’avec des grilles de format 4x4 pour simplifier le traitement toutefois, c’est une option que nous aurions aimé réaliser si nous avions eu plus de temps.

4. Conclusion

Tout au long de ce projet, nous avons pu mettre en pratique les différentes méthodes et raisonnements développés jusqu’alors en cours. Au cours du temps, nous avons modifié notre code dans l’optique de l’améliorer en le simplifiant tout en lui apportant un plus grand nombre d’options que possible. Réaliser ce projet aura été très long de réalisation toutefois,

il nous aura donné un but. Nous avons trouvé intéressant de découvrir l’arrière plan d’un jeu, aussi simple soit-il.