Rapport Projet:
Logimage

Groupe: QUACH Kevin, RAJON Lionel
TDc TP5
2019-2020 à Université Paris-Est Marne-La-Vallée en L1

Le but de cet exercice est d'écrire un programme python permettant de jouer au jeu Logimage à l'aide du module python "upemtk". Il s'agit d'un jeu de type casse-tête dans lequel le joueur doit reconstituer une image avec pour seul indice le nombre de cases colorées sur chaque ligne et colonne. Nous somme de plus demandé d'y faire des améliorations telles que: ajouter plusieurs grilles, ajouter une bonne interface graphique, ajouter un solveur automatique pour les grilles, etc...

Code de couleur:
En rouge: fonction
En vert: liste

Manuel d'utilisateur

Pour lancer notre programme, il suffit de l'ouvrir avec un éditeur python et de l'exécuter. Vous vous retrouverez alors sur le menu du jeu Logimage. Cliquez sur "Choix de la grille" pour choisir une grille (Chat par défaut) , sur "Quitter" pour fermer le programme, et enfin sur "Jouer" pour commencer une partie sur la grille sélectionnée.

Durant le jeu, cliquez sur une case blanche pour la colorier, sur une case colorée pour la rendre blanche. Le bouton retour (celle avec une flèche) permet d'annuler votre action précédente. Le bouton Menu vous permet de retourner au menu du jeu. Enfin, Cliquer sur l'ampoule activera le solveur automatique.

Votre but est de reconstituer l'image avec pour indice les nombres affichés à coté de chaque ligne et au dessus de chaque colonne. Ces nombres indique le nombre de case colorée sur chacune de ces lignes et colonnes. Bonne chance !

I - Ecriture du code pour le Logimage

A - Création du plateau

Pour faire le logimage ,nous avons tout d'abord commencer par la création du plateau. Pour se faire on a défini en premier lieu la taille des cases ainsi que le nombre de case composant le plateau qui est de (8 x 9) et créer une fenêtre qui contiendrait ces éléments. La fonction "*plateau"* permet ensuite, via deux boucle for, de dessiner les cases du plateau.

L'état du plateau est décrite par la liste *board*, qui contient une liste pour chaque ligne du plateau, elles même contenant une valeur pour chaque case de la ligne. Ces cases peuvent avoir deux valeur: 0 ou 1. Pour changer cette valeur, il suffit de cliquer sur une case, la fonction "*case_select*" détecte alors les coordonnées en pixel de la case sélectionnée, les convertissent en coordonnées cases et change la valeur de cette case dans la liste *board*.

C'est la fonction "remplissage_plateau" qui permet le coloriage des cases, celle-ci va lire la valeur de chaque case dans la liste *board* et colorie les cases de valeur 1 et laisse blanche les cases de valeur 0.

B - Mise en place du menu

Le menu est construit à l'aide de fonctions du module "upemtk" tel que "rectangle", "image" ou "texte". À l'aide de ces dernières, nous avons dessiné les boutons du menu, "Jouer", "Choix de la grille","Quitter" et nous avons ajouté l'image de fond ainsi que les différentes icônes pour les boutons.

C - Grilles et mécanisme du jeu

Chacune de nos grilles nécessite trois listes, une première décrivant l'état des cases de la grille si elle était complétée, une deuxième contenant le nombre de case colorée de la grille sur chaque ligne, et une troisième faisant de même pour les colonnes.

Lorsqu'on lance une partie, à chaque sélection de case, on vérifie l'état de la liste *board* et la compare à la liste de la grille contenant l'état des case qu'on doit obtenir, à l'aide de la fonction "check_board". Lorsque les deux listes sont identiques, la fonction retourne *True*, mettant fin à la partie.

II - Les améliorations

A - Ajout de fonctionnalité

1 - Plus de grilles

Suivant la méthode décrite plus haut, nous avons ajouté 2 grilles supplémentaires en plus de "canard", ces dernières étant "*chat*" et "*note_musique*". Bien que différentes, le nombre de case reste le même , en effet la grille reste sous le format (8 x 9) et fonctionnent de la même façon que la grille chat.

2 - Le bouton retour

À chaque sélection de case via la fonction "case_select", on stocke ses coordonnées en case dans la liste *history*. Lorsqu'on appuis sur le bouton retour, la fonction va inverser la valeur de la case dont les coordonnées sont celles en dernière position de la liste dans la liste *board*. Cette valeur est ensuite retirer de la liste *history*.

3 - Le bouton menu

Le bouton "Menu" permet comme son nom l'indique de revenir au menu principal et donc d'accéder de nouveau à "Jouer" , "Choix de la grille" et "Quitter" , il est accessible depuis le plateau de jeu.

4 - Le bouton rejouer

Le bouton rejouer permet de recommencer la grille que l'on vient de terminer , celui-ci varie donc en fonction de la grille sélectionnée et s'affiche uniquement quand vous terminer la grille.

B - Le solveur

Nous avons décidé de coder un solveur qui essayera chaque possibilité du plateau jusqu'à trouver la bonne. Cette fonction va colorier le nombre de case indiqué par l'indice de cette ligne de la gauche vers la droite, puis vérifie pour chaque colonne si le nombre de case coloriée ne dépasse pas leurs indices. Si non, elle passe à la ligne suivante et réitère l'opération décrite en haut. Si oui, les case qui viennent d'être coloriée redeviennent blanche et on recolore le même nombre de case après s'être décaler d'une case vers la droite. Si on arrive au bout des possibilités de la ligne, alors la fonction retourne à la ligne précédente, réinitialise les cases de cette ligne et les recolorie à nouveau après s'être décaler d'une case vers la droite.

Ce solveur devrait, en théorie, marcher. Cependant il semble ralentir considérablement après un moment.

III - Les fonctions importantes

Les fonctions suivantes sont celles qui sont les plus sollicitées dans le code et ont un rôle important pour son fonctionnements:

case_select: Cette fonction permet de changer la valeur de la case sélectionnée dans la liste *board*, de 0 à 1 ou de 1 à 0. Elle récupère les coordonnées du clic gauche, les convertissent en coordonées en case sur le plateau (si on clique sur une case) , et change la valeur de cette case. Les coordonnées en case étant les index de sa valeur dans la liste *board*.

remplissage_plateau: Lis les valeurs de chaque case dans la liste *board* et colorie ou non la case en fonction de sa valeur. Si la valeur est 1, la case est colorié, si la valeur est 0, est laissée blanche (ou est recolorié en blanc dans le cas où on désélectionne une case) .

check_board: Cette fonction vérifie l'état actuel du plateau via la liste *board* et la compare à la liste de la grille sélectionnée. Elle retourne True si les deux listes se correspondent.

solver: (Voir l'explication du fonctionnement de cette fonction plus haut.)

IV - Organisation

A - Partage et méthode de travail en période de confinement

Durant cette période de confinement due au COVID-19, nous n'avons pas eu l'occasion de se rencontrer physiquement pour travailler ensemble. De ce fait, nous avons opter pour un travail "au tour à tour". Lorsque l'un de nous avance sur le projet, on envoie le code à notre binôme via Discord, et lui explique ce qu'on a ajouter, ce dernier relie et, si besoin, corrige et/ou optimise le code.

Une estimation du pourcentage de répartition du travail serait de 40% pour Lionel et 60% pour Kevin.

Lionel Rajon	Kevin Quach
• Plateau, Sélection et remplissage des cases • Affichage des numéros sur chaques lignes et colonnes • Grilles supplémentaires et bouton "Choix de la grille" • Bouton "Rejouer"	• Optimisation de la méthode de sélection, d'affichage du plateau, de remplissage des cases, et de l'affichage des numéros • Menu et écran de fin • Boutons "Jouer, "Menu", "Quitter", "Retour", "Solveur" • Désélection et historique des mouvements • Solveur

B - Etat actuel du projet

Le projet est terminé bien que des améliorations sont possibles et qu'il soit toujours à optimiser afin de réduire le temps du solveur par exemple.

C - Difficultés rencontrées

Les difficultés rencontrées ont été en grande partie pour le solveur, ce qui est sans surprise. Ils a été en effet difficile de planifier les étapes du solveur puis de les écrire sous forme de code python.