TP de fin d'année : 2048

PTSI Lycée Eiffel

23 mai 2014

Pour ce dernier TP de l'année, nous allons tenter de faire quelque chose d'un peu plus ludique que d'habitude en programmant une version rudimentaire du jeu 2048 (bien sûr, ceux qui trouvent plus rigolo de résoudre des systèmes d'équations aux dérivées partielles d'ordre 12 peuvent améliorer notre programme Euler vu en cours s'ils le souhaitent). On essaiera dans un premier temps d'écrire des fonctions permettant de gérer les déplacements des chiffres dans la grille et l'apparition des nouveaux nombres, puis les plus rapides pourront essayer de se lancer à la découverte du module Tkinter de Python pour faire un peu de graphisme. Les questions *en italique* dans le sujet constituent des étapes facultatives sans lesquelles on peut créer un jeu qui tourne à peu près correctement, destinées à en avoir une version qui se rapproche le plus possible de l'original.

1 Gestion de la grille

1.1 Règles du jeu

Votre première mission consiste tout simplement à taper le mot-clé « 2048 » sur Google et à jouer jusqu'à ~~avoir fait un 4096~~ avoir bien compris les règles du jeu. Bon, pour ceux qui ne connaitraient vraiment pas, c'est simple : à chaque tour, le joueur choisit un déplacement parmi les quatre possible (haut, bas, gauche, droite), et chaque nombre dans la grille se déplace dans cette direction jusqu'à rencontrer un nombre différent de 0, en fusionnant avec ce nombre si et seulement si il lui est égal. On peut avoir deux fusions sur une même ligne (ou colonne) lors d'un même déplacement, mais pas deux fusions successives du même nombre. Pour être plus clair, si on se déplace vers la gauche, la ligne $2, 2, 4, 4$ deviendra $4, 8, 0, 0$ (deux fusions de deux paires de cases distinctes), mais $4, 2, 2, 1$ deviendra $4, 4, 1, 0$ (le deuxième 4 venant d'être créé ne va pas fusionner à ce tour avec son voisin).

1.2 Gestion des déplacements

On représentera désormais la grille de jeu comme une liste de quatre listes contenant quatre entiers chacune. Chaque ligne sera ainsi une liste Python de quatre éléments (mais pas les colonnes, qui sont constituée de quatre éléments pris dans quatre listes différentes).

- Écrire une fonction **deplaceligne** prenant comme argument une liste, et retournant la liste déplacée vers la gauche (en suivant les règles du 2048, bien entendu). On fera attention à bien vérifier ensuite que tous les cas possibles sont pris en compte.
- En déduire une fonction **deplacementgrille** prenant comme arguments une grille et une chaine de caractères (pouvant prendre les quatre valeurs 'haut', 'bas', 'gauche' et 'droite' par exemple), et ressortant la grille déplacée dans la direction indiquée par la chaine de caractères (sans ajouter de nombre dans la grille pour l'instant). Soit vous écrivez trois autres fonctions sur le modèle de deplacementligne pour gérer les déplacements dans les autres directions, soit (c'est mieux) vous vous ramenez à toujours utiliser deplacementligne en manipulant intelligemment les listes.
- *Modifier la fonction deplacementgrille pour qu'elle détermine si un déplacement (au moins) a vraiment été effectué dans la grille ou non (si ce n'est pas le cas, il ne faudra pas rajouter de nouveau chiffre dans la grille mais attendre que le joueur effectue un autre déplacement).*

1.3 Ajout de nouveaux chiffres

Après chaque déplacement, le jeu doit ajouter un nouveau chiffre égal à 2 ou à 4 dans une case vide (contenant un zéro) aléatoire de la grille.

- Écrire une fonction **ajoutechiffre** prenant comme argument une grille, et effectuant cette opération. On aura recours à la fonction randint du module random pour choisir la case à remplir (par exemple en créant une liste contenant les coordonnées de toutes les cases vides de la grille et en piochant un élément au hasard de cette liste). Si aucune case n'est vide, le jeu doit s'arrêter et afficher un message au joueur (du style « Espèce de gros nullard, la partie est finie »).
- *Modifier la fonction précédente pour que le nouveau chiffre intégré dans la grille ait plus de chances d'être un 2 qu'un 4 (du genre probas respectives de 0.75 et 0.25, je ne sais pas quelles sont les valeurs exactes dans le vrai 2048).*

1.4 Test de la grille

- Écrire une fonction **verifiegrille** qui teste si la grille contient un 2048, et affiche un message de victoire au joueur si c'est le cas (mais n'arrête pas forcément la partie ; si vous décidez de la faire continuer, il faudra faire attention à ce que le message ne s'affiche pas ensuite après chaque déplacement tant qu'il y a un 2048 dans la grille).

2 Aspect graphique

Pour faire un peu de graphisme avec Python (comme avec n'importe quel autre langage de programmation), il faut utiliser une bibliothèque graphique (en anglais Graphical User Interface ou GUI), la plus commune étant Tkinter. Il s'agit d'un module qui s'importe comme n'importe quel autre module sous Python, et qui contient toute une quantité de fonctions et de méthodes permettant bien évidemment de créer des objets graphiques. Tkinter est très orienté objet, ce qui signifie en gros que toutes les variables graphiques créées avec Tkinter seront vus comme des objets sur lesquels vont s'appliquer des méthodes, certains objets pouvant être des sous-objets plus importants (typiquement, l'objet principal sera une fenêtre dans laquelle on insèrera des widgets, composants graphiques situés à l'intérieur de la fenêtre comme des boutons ou des zones de texte).

- Essayer de se familiariser avec le fonctionnement de base de Tkinter (on peut faire quantité de choses avec, ne vous lancez pas non plus plus dans des choses trop compliquées !), et notamment tenter de créer une fenêtre contenant un bouton avec du texte dedans.
- Créer avec Tkinter une grille quatre lignes quatre colonnes contenant les nombres d'une liste de listes Python telle qu'utilisée pour modéliser notre grille plus haut.
- *Remplacer l'affichage des nombres dans la grille par des images pour rendre le tout moins laid.*
- Comprendre comment gérer les événements clavier avec Tkinter, pour que la grille puisse se mettre à jour quand le joueur appuie s'une des quatre touches de déplacement.
- Finaliser votre programme 2048.
- *Ajouter au programme le calcul du score : à chaque fois que deux nombres identiques fusionnent, on additionne au score la valeur du nouveau nombre obtenu.*

3 Amélioration de mon programme

J'ai réussi à bidouiller un programme Python qui marche à peu près, avec une interface graphique aux confins de l'immonde. Le programme se trouve ici, votre mission est de l'améliorer :

http ://www.normalesup.org/ glafon/eiffel13/2048.txt

- Comprendre le principal bug du programme, à savoir pourquoi l'ancienne grille ne disparait pas au moment où s'affiche la nouvelle.
- Modifier le programme pour qu'il tienne compte des coups où on n'a rien déplacé (pour l'instant, le programme ajoute quand même un nombre dans ces cas là !.

- Ajouter la prise en compte des probas différentes d'apparition du 2 et du 4, ainsi que le calcul du score.
- Venir faire les dernières séances consacrées à ce TP à ma place.