Description du code du labyrinthe McGyver

Github: https://github.com/GreyDaxx/McGyver/commits?author=GreyDaxx

Dans un premier temps, il m'a fallut créer la boucle de gameplay qui me servira pour toute la suite du jeu:

J'ai donc créé le Fichier main.py qui sera le fichier à lancer afin de profiter du programme. Cette boucle se sert de Pygame afin de prendre en compte les differents inputs de l'utilisateur ainsi que leur direction correspondantes (la direction faisant appel à une fonction que nous verrons plus bas), mais également le rafraichissement de l'affichage de l'aire de jeu.

Dans un second temps il faudra génerer l'aire de jeu:

C'est ansi que le fichier laby.py fut créé. Il va se baser sur un fichier.txt créé au préalable avec l'architecure du labyrinthe sous forme de chiffre en lignes et de colonnes correspondant chacunes à une ligne et une colonne dans lesquelles se déplacera McGyver. Chacune d'entre elles ayant une taille de sprite définie dans le fichier constantes.py

Selon la valeur des chiffres nous aurons une fonction qui viendra attribuer ces valeurs à des murs/sols/ou protagonistes à l'interieur du labyrinthe. Ces derniers seront affichés selon des images préalablement placées dans un fichier Pics grâce à une fonction appelée draw dans chaque case correspondante.

La gestion des objets:

Afin de "pimenter" un peu les parties, il faut que les objets aient des positions différentes entre chaque partie. C'est ainsi que dans le fichier qui sera nommé objet.py nous utiliserons l'import de random qui nous aidera à génerer de nouvelles positions pour chaque objet.

Ainsi les objets que nous avons choisis, à savoir un champignon, un pied de biche et une seringue seront générés aléatoirement, oui, mais surtout uniquement sur les parties du labyrinthe sur lequel McGyuver peut se déplacer, à savoir le sol (qui a une valeur numérique particulière comme nous l'avons vu plus haut).

La gestion de McGyver et son utilisation:

Maintenant que nous avons notre boucle de gameplay, l'aire de jeu, et les objets mis en place, il nous

faut gérer les déplacements de McGyver ainsi que sa gestion des objets.

Dans un fichier appelé character.py, nous allons generer son inventaire ainsi que récuperer la position du joueur, qui va nous servir à chaque input; c'est ici que McGyver va acquérir la possibilité de se déplacer dans le labyrinthe.

Dans la fonction déplacement nous allons attribuer à chaque input le déplacement adéquat (haut, bas, gauche, droite), cependant nous ne voulons pas que McGyver se transforme en Casper le fantome et traverse tous les murs. Nous allons donc créer une condition qui empèchera McGyver de se déplacer dans la direction que le joueur lui a donné si sa prochaine position est un mur.

Il ne reste plus qu'à vérifier à chacun de ses déplacements si McGyver est à la même position qu'un objet placé dans le labyrinthe, si c'est le cas, McGyver va le récuperer, l'objet va donc disparaître du labyrinthe et se retrouver dans son inventaire (aussi bien logique que graphique)

Finalisation de la boucle de gameplay:

Nous pouvons désormais définir les conditions de réussite ou de défaite du joueur. Pour ce faire nous allons tout d'abord verifier que la position de McGyver est bien identique à celle du gardien, si c'est le cas nous allons vérifier le deuxieme facteur de résussite: a-t-il les 3 objets nécessaires?

Ces facteurs seront verifiés à chaque fin de boucle entre chaque input de joueur; si le joueur ne remplit pas le premier, le deuxième ne sera pas verifié. En revanche, si le joueur est bien à l'emplacement du gardien, MAIS n'a pas récuperé l'ensemble des objets c'est une défaite et une image en conséquence s'affichera.

Si le joueur remplit les deux conditions la victoire est à lui, et une image s'affichera le lui indiquant.

Optimisations, finitions:

Il faut désormais faire en sorte que le code respecte les normes pep-8 ainsi qu'une documentation des differentes classes et fonctions grâce aux docstrings.

Le travail n'est cependant pas fini, il me faut encapsuler les données afin qu'elles ne soient pas accessibles directement par l'utilisateur, ce qui oblige de créér de nouvelles fonctions dans les differents fichiers.py afin de pouvoir récuperer ces dites valeurs lorsqu'elles sont appelées.

Enfin réunir mes fichiers.py utilisés par mon fichier main.py en packages et modules qui seront directement appelés par main.py.