PROJET 3

Lien GitHub: https://github.com/emmanuelfd/Project3

Décision initiale :

Un fichier py qui fait tourner le jeux et un fichier contenant les classes.
Je me suis orienté rapidement sur l'idée de 3 classes :

- Une pour la création du jeux lui même.
- Une pour gérer McGyver.
- Une pour gérer le mouvement.

Il m'a paru assez intuitif d'utiliser un fichier csv pour le format du labyrinthe. Une ligne par liste avec chaque case représentée par un chiffre suivant sa nature (mur, couloir, objet,..).

```
[[5, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
[0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0],
[0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0],
[0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],
[0, 1, 0, 1, 0, 1, 0, 1, 4, 0, 1, 0, 1, 0],
[0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0],
[0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0],
[0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 2, 0],
[0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 3, 0],
[0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0],
[0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0],
[0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0],
[0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 9]]
```

Difficultés rencontrées

- **<u>Le module class_board</u>**

    - Class BoardGame:

1. comment générer le damier en 2D. J'étais parti sur un dico ayant la forme cle(x,y) = case. Trop compliqué, voire impossible (les clés ne peuvent pas être un tuple).
Mon mentor m'a orienté sur les listes de liste.

2. Pas de difficulté rencontrée avec Pygame, j'ai commencé en chargeant un fond de jeux et rajouté des murs et objets. J'ai changé en cours pour gérer les cases de la liste une par une pour plus de flexibilité.

3. Au niveau du code Python lui même, j'ai mis du temps a trouver comment faire marcher la fonction load_board. Pour moi toute fonction définie dans une classe devait contenir un «self».

Avec le self, l'erreur «1 argument missing» était levée. Sans le self, impossible de faire référence a mes attributs de class (window/floor, ..).
Le problème a été réglé avec le @classmethod avant la fonction.

- Class McGyveret et Class Position :

Rien a signaler.

- **Fichier Game.py**

La gestion des mouvements (+1 sur x ou y) est assez intuitive quand vous avez sous les yeux les listes imbriquées.

J ai essayé d'alléger le plus possible en définissant un maximum de méthode au niveau des class.
J'ai laissé la gestion du «next_move» dans le fichier Game.py, pour le déplacer au niveau de la class il fallait passer en arguments tous les «gyver.position...». Cela rendait le tout un peu confus.

- **Autres difficultés.**

Les principales difficultés rencontrées n'ont pas été en rapport avec le code lui même.
Mais :

- J'ai eu du mal a utiliser virtualenv.

Une fois installer, impossible de créer un environement virtuel. Pour finir par réussir a lancer mon Game.py, j'ai passé de long moment a chercher les commandes sur le net.

- GitHub

Pour des questions de logistique, j ai travaillé sur le projet d'un endroit n'ayant pas accès a GitHub. J'avais copié mes fichiers en local et les modifiais. Lorsque j'ai recopié mes fichiers sur mon répertoire Git, impossible de les synchroniser avec mon GitHub (message : Everything up-to-date). J'ai recréé un repository.

- Notepad++

Beaucoup de temps perdu en utilisant Notepad++ comme éditeur. J'utiliserai PyCharm pour le prochain projet.

Améliorations envisagées

J'ai envisagé plusieurs axes d'amélioration.

- créer des niveaux avec des csv differents

- simplifier le fichier Game.py avec des méthodes plus nombreuses dans les class (gestions des évènements des touches)