PROJET 3: MACGYVER LABYRINTH

Démarche adoptée

En premier je me suis concentré sur la création de la logique du labyrinthe dans un fichier texte, puis je suis passé à la structure du programme: j'ai opté pour une division en trois scripts différents, un contentant les constantes du jeu (les images utilisées, les symboles de représentation du labyrinthe et les dimensions de la fenêtre du jeu), le deuxième contenant les classes nécessaires à la création des éléments et le troisième destiné au programme principal.
J'ai commencé en calculant les dimensions de la fenêtre à afficher, j'ai choisi un rectangle de 15x16 pixels: 15x15 étant la dimension demandée du labyrinthe et une ligne supplémentaire en dessous pour montrer le objets récupérés par MacGyver. Ensuite j'ai défini les images à utiliser, le titre de la fenêtre à créer et les symboles utilisés pour dessiner la structure du labyrinthe.
Je suis passé donc aux classes du jeu, la partie la plus compliquée à mon avis. Je vais détailler les passages selon le nom des classes utilisées.
1. Création de la classe «Position» pour définir le positionnement de tous les éléments dans la fenêtre du jeu, en calculant la position en pixels.
2. Création de la classe «Maze» grâce à l'initialisation d'une liste vide et une condition «*for*» pour parcourir le fichier texte concernant la structure de mon labyrinthe.
2a. Création d'une méthode «show»pour l'affichage à l'écran du labyrinthe et de l'inventaire dynamique de MacGyver. Dans la première version du programme j'y avais placé aussi l'affichage des objets nécessaires à la victoire (l'aiguille, l'éther et le tube en plastique) mais au final j'ai préféré les gérer à part vu l'implication de la fonction *randint* appartenant à la librairie *random*.
3. Création de la classe «Characters» pour la création des personnages du jeu et la définition des leurs caractéristiques.
4. Création du gardian et de son positionnement initial.
5. Création de MacGyver et de son positionnement initial.
5a. Création de la méthode «move» pour définir les mouvements de MacGyver et la gestion des possibilités de collision avec les murs.
6. Création de la méthode «take_objects» pour gérer la collision avec les objets nécessaires à la défaite du gardian à travers l'utilisation des structures conditionnelles *if*, *elif* et *else* et à l'initiation de la variable dédiée à les compter à 0. Si la position de MacGyver est égale à la case de l'objet à récupérer, l'objet est enlevé de l'écran et ajouté à la ligne de l'inventaire (+= 1).
7. Création de la classe «Item» pour générer l'aiguille, le tube en plastique et l'éther et en gérer leur positionnement au hasard dans le labyrinthe.
7a. Crèation de la méthode «place_random» pour positionner les objets toujours dans des endroits différents à chaque démarrage du jeu. Pour cela j'ai utilisé la fonction *randint* en lui donnant pour marge inférieure de l'intervalle le nombre 0 et pour marge supérieure la longueur de la liste de l'attribut de classe «LAB_STRUCTURE» (-1 à la fin car avec *randint* la marge supérieure est incluse).
8. Création des classes concernant les objets dans le labyrinthe pour en gérer le positionnement au hasard en appelant la méthode «place_random».
9. Création de la classe «Endgame» pour définir les conditions de victoire ou défaite et afficher les relatives images à la fin du jeu.
La partie finale a été consacrée à la structure principale du programme: j'ai défini les caractéristiques de la fenêtre du jeu, l'initialisation des personnages et des objets du labyrinthe, j'ai mis les variables du jeu à 1 pour gérer les événements avec des boucles (tant que la variable du jeu = 1 fais cela… sinon si la variable du jeu = 0 fais cela).
Enfin j'ai défini le contrôle des mouvements de MacGyver avec les touches directionnelles et les conditions de victoire à l'aide de la classe «Endgame».

Difficultés rencontrées

J'ai eu deux problèmes principales: 1) gérer les mouvements de MacGyver au début parce que je n'avais pas bien compris le fonctionnement du calcul de sa position dans le labyrinthe. J'ai passé deux jours à me documenter sur internet sur ce genre de situation avant de pouvoir continuer mon programme.

2) Gérer la collision entre objets. Là aussi je suis resté trois jours sur le problème en regardant des exemples et essayant différents solutions qui ne marchaient pas. Comment les retirer de l'écran et les visualiser dans l'inventaire? Au final j'ai compris qu'il fallait placer le bloc des éléments avant le rafraîchissement de l'écran (*display.flip*).

Ensuite j'ai eu des soucis mineurs comme la gestion du positionnement au hasard des objets, résolu grâce à la création d'une méthode à part par rapport à la première version du jeu, et la compréhension de certains erreurs (notamment «unexpected indent error»).

J'ai beaucoup apprécié toutes ces difficultés car m'ont permis de développer ma connaissance -encore très très très limitée- sur des problématiques réelles de la programmation orienté objet et sur la correcte manière d'écriture d'une programme (grâce à la lecture des conventions de la PEP8).

Propositions d'amélioration du jeu

1) Permettre au joueur de sauvegarder, quitter et reprendre le jeu plus tard;

2) Ajouter des niveaux en difficulté croissante (en créant différents fichiers texte et en les ouvrant à la fin du niveau précédent ou en les créant aléatoirement);

3) Gérer les mouvements de MacGyver à travers la pression continue des touches directionnelles.

4) Créer un compteur de score du jeu (exemple: chaque niveau complété équivaut à un certain score);

5) Ajouter une musique au jeu pour le rendre un peu moins triste.