Python_Projet3_Démarche_PierreSempéré

1 – Cahier des charges

La première étape dans la création du programme a été l'établissement du cahier des charges : ce que le produit final doit être capable de faire, à quoi il va ressembler etc., pour cela j'ai eu recours à des listes et schémas détaillant majoritairement l'interaction utilisateur – programme (l'aspect code a dans un premier temps été laissé de côté pour envisager au mieux mes attentes vis-à-vis du rendu final).

Cette phase m'a permis de savoir précisément ce que l'utilisateur – ou joueur devait être capable de faire, ce à quoi devait ressembler l'interface utilisateur etc.

Cela a été une phase essentielle puisqu'elle a déterminé les connaissances nécessaires à la réalisation du programme en lui-même et a permis la division du programme en étape bien définies.

2 – Connaissances nécessaires

Lors de cette étape j'ai pu mettre en place une sorte de liste contenant les connaissances dont j'avais besoin pour la réalisation du programme.

Les principales ont été :

- L'OOP qui semble être le point central du programme. En particulier la division des objets en classes auxquelles ont été greffées des attributs spécifiques nécessaires à l'exécution d'un code le plus léger et le moins répétitif possible. – Cela a probablement été le problème le plus compliqué à résoudre.
- La librairie PyGame, essentielle à la réalisation du programme puisqu'elle a permis de gérer les affichages et les inputs du joueur. Fort heureusement la documentation et les guides trouvés sur internet sont très clairs et l'acquisition des connaissances utilisées dans le programme ont été relativement simples même si des améliorations sont sans doute possibles.
- Les librairies Random et Pathlib ont également été essentielles dans la réalisation du programme : Random pour la génération aléatoire des positions des items au démarrage du programme et Pathlib pour structurer l'arborescence du dossier final de façon logique pour permettre la séparation des images et du code afin d'ajouter en visibilité (la librairie OS aurait pu être utilisée mais la lisibilité de Pathlib a été préférée)
- La division du programme en boucle principale et modules complémentaires, cela a permis d'alléger la structure globale du programme et a rendu les ameliorations et corrections de bugs beaucoup plus simples.

3 – Structure Globale

La structure du programme est relativement simple et, je pense, assez lisible : les classes sont déclarées et les instances et leurs attributs initialisés au début du programme, les images sont ensuite chargées et la boucle PyGame, dans l'ordre, vérifie les conditions diverses (on vérifie si le joueur est en vie, quels objets ont été ramassés et si les conditions de victoire ou défaite sont présentes). En fonction de ces conditions sont affichées la majorité des sprites (sols et murs exclus) et les contrôles bloqués en cas de « GameOver ». Les contrôles sont dépendants de la condition « GameOver » pour ne laisser que la possibilité « Exit » au joueur une fois le jeu terminé.

4 – Problèmes et solutions

Les problèmes rencontrés dans ce programme ont majoritairement été lié à l'utilisation de l'OOP dans le programme : je ne comprenais pas l'utilité de l'utilisation de l'OOP dans le projet et l'utilisation des classes pour alléger le programme. C'est majoritairement grâce à l'aide de mon mentor et à des cours trouvés sur YouTube (@Corey Schafer) que les problèmes relatifs à l'OOP ont été résolus (j'ai trouvé le cours d'OC sur le sujet peu clair et trop imagé pour moi)

La structure fichier du programme a également posé problème, mais l'utilisation de Pathlib conseillée par mon mentor a permis de comprendre d'où venait le problème et de le corriger.

La gestion des coordonnées a également été problématique puisque les coordonnées manipulables (celles du joueur) sont sous forme de liste alors que de nombreuses autres (celle de la carte ainsi que celles des objets) sont sous forme de tuples. Ceux-là n'étant pas comparables sans conversion, la solution au problème a été de convertir la position du joueur (macpos) en tuple au début de chaque boucle PyGame pour permettre la comparaison de coordonnées, pour pouvoir ramasser un objet par exemple. La liste « macpos » a néanmoins été gardée sous format liste pour permettre une modification simple lors de la gestion des inputs.

5 – Améliorations possibles

Le programme n'est bien sûr pas parfait et des améliorations pourraient être mises en place. Par exemple, le parti pris d'utiliser un appui constant de touche au détriment d'un input clavier enregistré quand le joueur aurait relâché la touche du clavier pourrait par exemple rendre le mouvement plus précis entre autres.

Lien du repo Git :https://github.com/Psemp/oc

La limite imposée à la taille de ce document empêche de développer certains points tels que l'initialisation de la classe « map » ou les interactions avec le document maze.txt, je serais ravi de répondre à toute question non soulevée par ce document.

Addendum sur les environnements virtuel. (J'utilise personnellement virtualenv)

Un environnement virtuel permet d'avoir un espace de travail isolé du reste du système. Cela est utile pour séparer les différents projets et s'assurer qu'il n'y aura pas de problème de compatibilité entre eux (si projet A ne fonctionne qu'avec la version 1 de Librairie X et que projet B utilise des spécificités ajoutées dans la version 2 de Librairie X, les deux peuvent être installées dans les environnements virtuels propres au projet pour éviter le conflit).

Formatted: Justified