Présentation

Le but du projet est de réaliser un jeu de labyrinthe. La ou le joueu·r·se part d'un point de départ et doit arriver à la sortie du labyrinthe en ayant collecté un certain nombre d'objets sur son chemin afin de neutraliser le gardien situé à la sortie !

Le jeu tourne autour de l'univers de MacGyver. La ou le joueu·r·se prend le contrôle du héros de la série.

Code source

L'ensemble du code source est hébergé sur la plateforme GitHub. Le dépôt contenant ce code source est le suivant : https://github.com/GuillaumeOj/HelpMacGyver

Déroulement du jeu

Initialisation

Suite à l'exécution de **main.py**, les actions suivantes vont se dérouler :

1. Initialisation de la bibliothèque **pygame** essentielle à la création de l'affichage du jeu et la gestion des événements utilisateurs.
2. Génération et affichage du labyrinthe en se basant sur le fichier **.txt** préalablement écrit.
3. Affichage du panneau d'affichage sur la droite du labyrinthe
4. Création et placement des personnages à leurs places respectives (le gardien à la sortie et MacGyver au départ).
5. Ensuite des objets sont générés et placés dans le labyrinthe. Le placement se fait aléatoirement grâce à l'utilisation du module **random** fourni avec Python.

Mouvement

Une fois ces premières étapes effectuées, le programme va surveiller et "capturer" les événements créés par la ou le joueu·r·se.

- Lorsqu'elle ou il appuie sur **ESC** ou la croix de fermeture de la fenêtre, le jeu se ferme.
- Lorsqu'elle ou il utilise les flêches directionnelles de son clavier, la mise en mouvement de MacGyver s'enclenchera.

La phase de mouvement enclenche un certain nombre d'instructions :

1. Effacement de MacGyver de son ancienne position
2. Calcul de la nouvelle position du héros dans le labyrinthe et vérification qu'il est autorisé à se déplacer dans cette direction (que ce ne soit pas un mur par exemple).
3. Sur la nouvelle position du personnage, vérification de la présence d'un objet. Si il y en a un, il est ajouté à la liste des objets du héros.
4. MacGyver est affiché à sa nouvelle position.

Fin du jeu

Pour marquer la fin du jeu, le programme vérifie que la ou le joueu·r·se à atteint la sortie du labyrinthe. Deux cas de figure se présentent alors :

1. Tous les objets sont ramassés au cours de la partie, dans ce cas, la ou le joueu·r·se gagne.
2. Ils restent des objets dans le plateau de jeu, la ou le joueu·r·se perd.

Dans les deux cas, un message de victoire / défaite apparait dans le panneau de droite ainsi qu'un menu composé de deux boutons. Ce menu propose de continuer à jouer ou non.

Bilan

Au cours de ce projet, j'ai surmonté plusieurs obstacles pour arriver au terme. Je liste par la suite trois d'entre eux, ceux qui m'ont paru les plus importants à retranscrire.

Pygame

Le premier obstacle a été l'apprentissage de Pygame.

Mon mentor m'a aiguillé au début sur un tutoriel permettant de faire bouger une image dans une fenêtre. Cet exercice m'a donné une bonne base de départ pour le labyrinthe.

Cependant je me suis rapidement aperçu que les bases ne suffisaient pas pour, par exemple, gérer la collision d'un objet avec un autre, utiliser la souris pour agir sur l'écran, prendre en compte une touche de clavier enfoncée en continu, etc.

Pour remédier à cela j'ai pris le temps d'approfondir la lecture de la documentation de la bibliothèque, je me suis inspiré de parties de codes réalisées par d'autres utilisateurs et j'ai fait des recherches sur des points précis (comment réaliser un bouton par exemple).

Limite de prestation

Même si les instructions de l'énoncé sont assez claire, j'ai eu beaucoup de mal à m'arrêter dans la réalisation du jeu. En effet, je suis parti dans une sorte de folie des grandeurs et j'ai voulou ajouter beaucoup plus de fonctions que demandées.

J'ai donc pris le temps de mettre à jour la To Do liste écrite dans le 'Readme.md' en m'accordant l'ajout de certaines fonctionnalités, telles que l'affichage des objets ramassés sur le côté et le menu de fin permettant de continuer à jouer.

Cohérence et propreté du code

Pour finir, je suis en pleine phase d'apprentissage dans le développement en Python. Entre le début du projet, le milieu et la fin, j'ai appris et appliqué beaucoup de nouvelles connaissances et pratiques. Par conséquent, ma façon d'écrire et de commenter mon code à beaucoup évoluée en cours de projet.

J'ai dû prendre des pauses pour relire les parties de codes déjà produites afin de m'assurer que l'ensemble est une certaine cohérence pour en faciliter la lecture ultérieure.