Projet 3
Aidez MacGyver à s'échapper !

Le code source du jeu est disponible à l'adresse suivante, sur mon repo GIT : https://github.com/pshop/p3_labyrinthe.

Au tout début du projet j'ai commencé par imaginer un pseudo diagramme de classes dans lequel allaient se trouver tous les objets destinés à interagir lors du lancement du programme. Ce diagramme n'avait pas vocation à être définitif ni rigoureusement exacte, mais plutôt à être modifié au cours de l'avancement du programme ce qui me permettait d'avoir toujours un œil sur sa structure. Dans l'idée le diagramme a influencé le code et le code s'il évoluait au-delà du diagramme, venait le mettre à jour.

En complément de ce diagramme j'ai dessiné le labyrinthe à la main et tenté d'imaginer la meilleure façon de le retranscrire. L'énoncé du problème nous conseillant de commencer à coder le jeu sans interface graphique. La première question a donc été de savoir comment afficher le labyrinthe dans la console. Le labyrinthe étant donc une simple grille je l'ai naturellement codé sous la forme d'un tableau à double entrée ainsi deux boucles imbriquées ont fait l'affaire pour afficher son contenu.

Le labyrinthe devait être contenu dans un fichier externe qui puisse être modifié sans avoir à modifier le code. J'ai donc opté pour l'utilisation d'un fichier json dont la structure est semblable à celle du python et facilement exploitable.

Pour la version text du jeu je n'ai pas implémenté l'utilisation des flèches directionnelles car je n'ai pas trouvé de bonne manière de capter le pressage d'une touche en mode console sans qu'il ne faille valider l'entrée utilisateur en appuyant sur enter. Il se trouve qu'avec pygame la question est bien plus simple à régler avec la gestion des évènements. Cette fonction a donc été implémentée lors de l'intégration de pygame dans mon code.

Pour ce qui est de la répartition aléatoire des objets, j'ai créé une fonction qui parcourt le labyrinthe et renvoie un tableau contenant toutes les coordonnées des cases libres dans le labyrinthe. Lors de l'initialisation de l'objet je lui donne ce tableau et utilise la fonction choice de random pour lui assigner un emplacement. Ainsi à chaque lancement et peu importe le labyrinthe, les objets apparaîtront de manière aléatoire sur des emplacements libres.

Pour ce qui est des déplacements de Macgyver en général, le programme prend l'entrée de l'utilisateur et vérifie ce qui est situé à l'emplacement demandé par l'utilisateur et stock cette valeur dans une variable next_item. Si cette variable s'avère être un mur où être hors du tableau, le programme ignorera l'entrée utilisateur et attendra simplement la suivante. Si next_item s'avère être une case libre alors le programme va afficher Macgyver sur cette nouvelle case, il en sera de même si la case contient un objet ramassable, Macgyver le remplacera immédiatement donnant l'impression qu'il ramasse l'objet. Ensuite le programme effacera Macgyver de son précédent emplacement. Et voilà, Macgyver avance, pour le meilleur comme pour le pire.

De plus lorsque le next_item est un objet que Macgyver doit ramasser il est directement ajouté au sac de Macgyver. Le contenu de ce sac sera vérifié lorsque Macgyver se rendra à la sortie du labyrinthe et déclenchera la condition de victoire, soit le sac est plein et la partie est gagnée soit il ne l'est pas et c'est la défaite.

Une fois le jeu fonctionnel, il était temps d'intégrer pygame et lui donner une interface graphique digne de ce nom. Pour l'affichage je suis parti sur un choix arbitraire d'une fenêtre de 600*600 px, simplement parce que j'ai décomposé mon tableau en cases de 40*40 qui étaient la taille d'un mur de brique dans les fichiers fournis pour le projet. Ainsi tous les objets affichés sont des png de 40*40 et rien ne dépasse d'un pixel.

De manière générale je n’ai pas rencontré de grandes difficultés avec pygame, je me souviens cependant que l’implémentation du déplacement de Macgyver a été une longue prise de tête. Je voulais un peu trop factoriser mon code et stocker de valeurs d’attributs de Macgyver dans des variables utilisées elles-mêmes dans des méthodes de sa propre classe. Je sais pas si c’est très clair mais je n’ai jamais compris pourquoi la modification des variables modifiait aussi les propriétés. Le problème a fini par être réglé mais j’ai dû abandonner mon idée, car Macgyver marchait à travers les murs sans que je ne puisse expliquer pourquoi.