Rapport sur la construction et l'évolution du développement du jeu Saving MacGyver dans le cadre du Projet 3 d'Openclassrooms.

1/ Planification :

Avant de commencer à coder, j'ai établi un plan de route afin de ne pas me perdre dans ma programmation. Pour cela, j'ai d'abord identifié les classes et les méthodes/fonctions qui me seraient nécessaires par rapport aux exigences du cahier des charges du jeu.

J'ai ensuite fait au brouillon un algorithme mi-textuel mi-code pour avoir une vision de mon programme.

J'ai ensuite réparti mon programme en plusieurs fichiers : class_map, class_player, class_object et saving_macgver, le fichier principal.

2/ Début de la programmation

A partir de là, j'ai commencé à transformer l'algorithme textuel en vrai code. Ma procédure aura été d'écrire tout d'abord le code dans sa plus simple expression (mais aussi la plus longue finalement) pour ensuite la traduire en classes et en méthodes de classes.

Pour la class_map par exemple, j'ai commencé par écrire deux listes contenant l'une les sprites de ma map (mur, sol, départ, sortie…) et l'autre ma grille avec des coordonnées (x, y) pour chaque valeur de la liste. J'ai ensuite fusionné les deux listes pour obtenir un dictionnaire avec comme clé les coordonnées en tuple et comme valeur les sprites.

J'ai ensuite cherché à l'afficher à l'aide de pygame et une fois réussi, mon but a été de générer automatiquement ces listes. C'est là que sont véritablement nés les fichiers « level1 » contenant le niveau et « variables », ainsi que la classe Map et ses méthodes permettant de générer le niveau et de l'afficher.

J'ai procédé de la sorte pour chaque classe en incorporant au fur et à mesure mes améliorations à mon code principal et en le testant.

3/ Amélioration et implémentations des fonctions

Par cette méthode, le programme a donc évolué au fur et à mesure que j'implémentais et remplaçais du code. Au début, il ne s'agissait que d'afficher la map : Classe Map

Puis il a été question d'intégrer le personnage principal et son ennemi : Classe Player. Les placer sur la carte. Créer la méthode pour déplacer MacGyver. Mettre à jour la position. Clôturer le programme avec conditions de victoire simple.

Puis au tour des objets : Classe Object. Générer les objets. Les afficher sur la map. Générer aléatoirement leur position sur la map. Les faire ramasser et mettre dans l'inventaire. Mettre à jour leur affichage. Les intégrer dans des conditions de victoire.

Les différentes boucles du programme (program, home_page et game) se sont créées quant à elle au fur et à mesure pour répondre aux besoins du jeu et de l'algorithme.

4/ Debuggage et affinage :

La dernière phase aura donc été d'améliorer le jeu, logiquement et graphiquement et d'y apporter quelques fonctions supplémentaires, pour le fun et pour voir ce qu'il est possible de faire.

Bien sûr, chaque phase aura rencontré son petit lot de problèmes puisque j'ai le sentiment d'avoir passer plus de temps à debugger mon propre code et à l'ajuster qu'à véritablement concevoir des fonctions et les traduire en code.

C'est bien sûr une impression et cela fait partie du processus j'imagine. Car, au final, on apprend beaucoup en revenant sur les erreurs et en comprenant pourquoi on les a faite et pourquoi ça ne fonctionnait pas. C'est aussi ce qui m'amuse le plus finalement et la joie procurée quand on trouve enfin la solution me pousse à vouloir aller toujours plus loin dans l'amélioration et l'implémentation de nouvelles fonctions. Quelques heures de sommeil auront été sacrifiées mais je n'aurais jamais fini ma journée sur un échec.

5/ Choix de l'algorithmique :

L'algorithme de mon programme a été défini pour les besoins du jeu et selon le cahier des charges imposé. Il se compose de :

- 1 boucle principale, démarrant le programme et contenant l'entièreté de l'algorithme et les 4 boucles annexes. Celles-ci contiennent les principaux états du jeu et se définissent chacune par un affichage particulier :
    - La boucle home_page, qui ouvre le jeu et présente le menu au joueur pour qu'il choisisse son niveau et démarre une partie.
    - La boucle game qui comme son nom l'indique démarre la partie et contient la quasi-totalité de l'algorithme finalement. Elle exécute toutes les fonctions principales du jeu et ne se referme que lorsque le joueur a gagné ou perdu.
    - La boucle win qui s'éxécute quand le joueur a gagné et lui propose de rejouer ou de quitter
    - La boucle loose qui s'éxécute quand le joueur perd et lui propose de rejouer ou de quitter.
- Une « boucle » intermédiaire entre les boucles home_page et game et qui ne s'exécute que si le joueur a fait le choix du niveau souhaité. C'est elle qui va générer le niveau, les joueurs, les objets et les afficher.

Un petit mot enfin sur le choix du dictionnaire pour la génération de la map :

Celui-ci m'a semblé tout indiqué pour cette situation où j'avais à établir une liste de coordonnées et de sprites. Plutôt que deux listes séparées, le dictionnaire me permettait de mettre en corrélation les deux données et ainsi définir chaque case avec ses coordonnées et son sprite. Par la suite cela m'a permis de pouvoir faire appel à l'un grâce à l'autre pour en modifier le contenu (dans le cadre du mouvement du personnage ou de la génération/récupération des objets par exemple) et afficher la map.

Pour retrouver mon code :

https://github.com/valld27/Project3-OC---McGyver