PROJET 3 : Aidez Mac Gyver à s'échapper !

Soutenance de projet

Bastien Renaud
(Exodrum : GitHub & Open Classrooms)
Code Source : https://github.com/Exodrum/PROJET_3_MAC_GYVER

Introduction :

Cette documentation présentera le ***« Projet n°3 : Aidez Mac Gyver à s'échapper ! »*** réalisé dans le cadre de ma formation de **développeur d'application Python** sur **Open Classrooms**.
Grâce à mon journal d'apprentissage, je peux avoir une vision globale de l'avancer de mon programme durant ces dernières semaines et pouvoir y mettre quelques mots sur mes choix.
Ce document aura pour objectifs d'expliquer les contraintes, difficultés rencontrés, choix technique et modules choisis durant le projet.

Structure Programme :

Ci-dessous un aperçu de l'organisation du programme.

Fichiers :

- lvl.txt
- main.py
- classe.py
- constants.py
- readme.txt

Modules :

- Random
- Pygame

Classes :

- Level
- MacGyver
- Guardian
- Item

Gestion du projet :

Le projet à été développé sous **SublimText3**, sous **Windows 10** en versionnant le code avec **GitBash/GitHub**.
J'ai passé la majorité de mon temps à étudier les cours et à les appliqués, rechercher des tutoriels sur **Youtube**, les **forums** et analysé la documentation **Pygame.**

L'avancement du projet par rapport aux points suivants :

Points positifs :
Diverses connaissances en **HTML5/CSS3/PHP/JS**.
Connaissances de **Photoshop CC**.
Organisations des étapes à effectuer selon un planning.

Point à améliorer :
Approfondir mes connaissances en python.
Envoyer plus souvent mon code sur **Github**.

Analyse de programmation :

Premièrement, j'ai créé la structure logique du labyrinthe dans un fichier texte.

Je me suis intéressé à la structure du programme que j'ai programmé en trois parties, cela me semblé le plus approprier pour éviter d'avoir trop de scripts pour un programme simple.

Voici les scripts :

constants.py : *contient les images utilisées dans le jeu, les dimensions de la fenêtre d'affichage et les données de positionnement de certaines variables.*

classe.py : *contient les classes concernant les objets du jeux (objets, personnages, labyrinthe).*

main.py : *contient les paramètres principaux du jeux (boucles et gestion d'affichage du jeux).*

Je me suis penché sur le calcul des dimensions de la fenêtre, j'ai choisi un rectangle de 15x25 (étant un projet personnel, je me suis permis de modifier les dimensions pour afficher un background plus agréable à l'œil).

J'ai créé un dossier « img », puis ajouter ma galerie d'images au sein de celui-ci, j'ai choisi de généré des images invisibles pour utiliser les arbres de l'image de fond d'écran comme textures de murs.

Viens après la partie des constants, la création des classes :

- **Ecriture de la classe Level :**

  Initialise une liste vide et ajoute une condition pour parcourir le fichier « lvl » et gère l'affichage du labyrinthe.

  Ecriture d'une méthode **generate**, créer une liste vide et ajoute une condition pour parcourir le fichier « lvl » et gère l'affichage du labyrinthe.

  Créer les lignes et les cases où seront placés les images dans la fenêtre et enregistre le labyrinthe.

  Ecriture d'une méthode **toggle** pour ajouter la texture des murs et le garde au bon endroit grâce à la structure précédemment chargée.

- **Ecriture de la classe MacGyver :**

  Initialise le personnage Mac Gyver et lui donne ses caractéristiques (images, position, nombre d'objets).

  Ecriture d'une méthode « **move** » pour définir les mouvements ses mouvements et gère les collisions.

  Ecriture d'une méthode « **take_items** » pour prendre un objet quand Mac Gyver rentre collision avec celui-ci, le rajoute à son inventaire. Si la position de MacGyver est égale à la position de l'objet, supprime l'image de l'objet et ajoute à la méthode « **inventory** » +=1.

  Ecriture d'une méthode « **inventory** » pour qu'à chaque objet que Mac Gyver obtient rajoute le nombre d'items en sa possession grâce à la méthode « **take_items** ».

- **Ecriture de la classe Guardian :**
  Initialise le personnage guardian dans labyrinthe et lui donne 1 point de vie.

  Ecriture d'une méthode « **damage** » pour si dommage sur gardien enlève 1 au point de vie.

- **Ecriture de la classe Item :**
  Initialise les objets, leurs donnes des caractéristiques et leurs positions random.

  Ecriture d'une méthode « **randpos** » pour donner aux objets une position au hasard à chaque fois que le boucle du jeu est relancé. J'ai utilisé la fonction « **randrange** **»** en donnant le nombre de case max (largeur et longueur) entre 1 < > X, lui donner une position à condition, « if » si les objets apparaissent sur les cases « 0 ».

  Ecriture d'une méthode « **damage** » et « **display** » si Mac Gyver se situe sur la même case que l'objet, enlève 1 point de vie et donc enlève l'objet de sa position « 0 » si **health** > 0.

Pour finir j'ai écrit le script principal où sont importés les constantes des caractéristiques de la fenêtre du jeu, les classes. Puis programmer les variables : **continu_game, continu_home, continu** pour activer ou désactiver les boucles principales : du menu principal et du jeu avec leurs conditions.

Définitions du contrôle des mouvements de Mac Gyver avec les touches directionnelles et conditions WIN/LOOSE à l'aide du système d'inventaire.

Difficultés rencontrées :

J'ai eu quelques problèmes principaux lors de la programmation :

1) **La génération d'objet random** :
   J'ai mis beaucoup de temps avant de trouver comment généré les objets de manière aléatoires dans la structure du labyrinthe, j'ai finis par utilisé **random.range** qui permet à la

méthode de parcourir les cases de la fenêtre, y déposé un objet de manière aléatoire, donc sur l'emplacement « 0 » et de retourner sa position pour permettre à la classe de l'afficher.

2) Les méthodes pour récupérer la valeur de l'objet et l'ajouter à l'inventaire :

Ce problème-là n'a pas durée très longtemps, j'ai cherché quelques indices sur le net, il m'a suffi de définir une classe pour prendre un item lorsque qu'il perd sa vie et de l'ajouter via une autre méthode inventaire « inventory » qui ajoute +1 à celui-ci.

Pour l'amélioration du jeu :

Musiques d'ambiance et sonores pour l'immersion.

Différent niveau dans la structure du labyrinthe, par exemple passer sur une autre zone via un portail, pour ajouter de la difficulté à trouver les objets et rendre le temps jeu plus long. Salle secrete

Personnages Non Joueur avec une direction prédéfini, qu'il faut esquiver pour éviter que Mac Gyver meurt.

Rajouter une barre de vie et une barre armure pour avoir plus de chance face au gardien.