Léo Pezard

Joachim Deschodt

Jules Illiano

Rapport de projet Poo :

Notre projet comporte 16 points de complexité, les éléments que nous avons choisi de programmer sont :

- G.I (interface graphique) 4pts
- G.II (points d'expériences) 1pt
- G.III(inventaire limité)1pt
- A.II (Repos) 1pt
- A.III (Magie) 2pt
- O.I (Nourriture) 1pt
- O.IV (Armure) 1pt
- O.VI (Solidité) 1pt
- S.II (Pièges) 1pt
- S.IV (Trésor) 2pts
- M.I (Poison) 1 pt

Bonus : Lorsqu'on lance le jeu, un bouton demande au joueur s'il souhaite jouer ou non et , lorsque le joueur perd, un bouton lui demande s'il veut rejouer.

(Nous avons eu un problème sur les appareils windows et macOS pour le focus du jeu avec ces boutons même avec la fonction focus_get() . Après avoir répondu au askyesno, nous devons cliquer sur le shell puis revenir sur le jeu pour pouvoir bouger)

Expliquons nos méthodes et fonctions :

1/ Gameplay I : Interface graphique :

A l'aide de tkinter nous avons créé une interface graphique qui nous plonge dans le monde de Mario (décor + personnages), pour cela nous avons ajouté 2 canvas afin de mieux répartir les effets graphiques.

- Nous avons modifié le __init__() de class Game afin de lancer le bouton de départ du jeu (debut), importer les éléments (bind_elements) et définir des polices de caractère avec tkFont (define_fonts).

- Grâce à bind_elements(), nous attribuons à chaque variable des éléments du jeu une image avec tk.PhotoImage.

  Chaque variable est placée dans un des dictionnaires (dfloors, dautres, delement, dinventory) et une abréviation leur est attribuée.

- Dans la classe Game, nous avons modifié les dictionnaires déjà présents (« equipments », «actions », « monsters ») et avons placé nos personnages, nos actions et nos équipements utiles pour la suite du jeu.

- La méthode gdraw() permet de parcourir la liste du sol du jeu et dessiner sur le 1er canvas les images en fonction de leurs abréviations sur floor._mat.

- La méthode draw_report() donne de la vie au jeu, on y dessine l'inventaire en parcourant hero._inventory , on dessine sous forme de barre de pourcentage les XP, la satiété et la magie.

- La méthode debut() permet de demander si le joueur veut jouer avec une fenêtre qui s'ouvre. (cette méthode est exécutée dans __init__ de Game)

- Modification de buildFloor() pour placer sur chaque « map » un trésor et une clef (même manière que pour placer Stairs)

- La méthode actions() combine toutes les fonctions et affiche les modifications à chaque déplacements, notons que les déplacements peuvent s'effectuer avec « zqsd » ou avec les flèches du clavier.

- La méthode initialize() permet de lancer le jeu lorsqu'elle est exécutée dans play()

- la méthode play() permet de mainloop notre fenêtre de jeu (problème de focus ici avec le askyesno du début)

2/ <u>Gameplay II : Points d'expérience :</u>

Dans la classe Creature, la méthode meet permet de rencontrer un monstre, si le héros tue le monstre, il gagne un certain nombre d'XP en fonction du monstre.
Au bout d'un certain nombre d'XP gagnés le héros passe au niveau supérieur, il regagne des points de vie et sa force augmente.

3/ <u>Gameplay III : Inventaire limité :</u>

Le nombre d'éléments de l'inventaire est limité à 10. (meet dans la classe Equipment et take dans la classe Hero ont été modifiées)

Fonction Game().deleteinvent() qui, lorsqu'on appuie sur 'o' demande quel objet supprimer de l'inventaire.

4/ <u>Actions II : Repos :</u>

Lorsque le héros prend le tuyau vert (escalier) pour changer de niveau de jeu, une fenêtre apparaît et celle-ci lui demande s'il veut se reposer, c'est-à-dire gagner 5hp, si c'est le cas les monstres bougent aléatoirement sinon ils restent comme ils étaient initialement.
Nous avons utilisé la méthode meet de la classe Stairs qui fait que lorsque le héros rencontre l'escalier la fonction askyesno fait apparaître une fenêtre avec marqué "Voulez-vous vous reposer ?". Si oui alors 5hp en plus et les monstres bougent, sinon rien.

5/ <u>Action III : Magie :</u>

Dans la classe Creature on initialise 20 points de magie que le héros peut utiliser soit avec la fonction teleportmagie qui va téléporter le héros aléatoirement dans la Map, (lorsque teleportmagie est utilisé on perd 4 pt de magie), soit healmagie qui va soigner le héros de 2 hp (en perdant 2 pt de magie)
Ces fonctions s'utilisent avec les touches "m" et "g".

6/ Objet I : Nourriture :

Dans la classe Hero, le héros possède un niveau de satiété de 20, à chaque déplacement, le héros perd 1 point de son niveau de satiété , lorsqu'il n'a plus de points dans son niveau de satiété, il perd 1 hp (avec la fonction satiété dans Map().move()) un élément appelé poulet 'p' permet de régénérer totalement le niveau de satiété.

7/ Objet IV : Armures :

Au début du jeu le héros ne possède pas d'armure, dans la classe héros, le niveau d'armure vaut 0.
Si notre héros Mario récupère une armure et l'utilise, son niveau d'armure passe à 10. Ainsi, lorsqu'il rencontre une créature il ne perd pas directement des hp mais des points d'armure (lorsque l'armure est à 0, elle est détruite et il perd des points de vie).
(fonction armure)

8/ Objet VI : Solidité :

La solidité est valable pour l'objet armure et l'objet étoile. Une fois que leur fonction est finie, ces objets se suppriment et le héros perd les attributs de ces objets.

9/ Salles II : Pièges :

Sur la carte, des pièges ressemblant au sol (map.ground) sont présents.
Nous avons décidé de mettre le piège dans les éléments, ainsi, une fois que le héros a rencontré le piège, il perd des hp et le piège est découvert (il ne peut plus marcher dessus).

10/ Salles IV : Trésor :

Sur chaque "map" se trouve un trésor et une clef (placées avec la même méthode que Stairs).
Chaque clef permet d'ouvrir un seul coffre (si le héro change de map il perd sa clef).
Chaque coffre contient un objet aléatoire (armure, up, fleur…) qui est ajouté à l'inventaire du héros lorsqu'il le rencontre.

11/ Magie I : Poison :

Le héro à un état (hero.poison = True / False) si cet état est à True, à chaque mouvement (Map.move()) il perd des hp (fonction Map().poison()) S'il utilise un champignon rouge, il n'est plus empoisonné (fonction stoppoison(hero))