NOM: Andjelovic
PRENOM: Lazar
N° étudiant: 22210685

Université Paris Cité

Projet: FANCY FENCING

Master 1 Informatique VMI 2022-2023

1/ Required

Tout d'abord avant de parler des implémentations qui ont été réalisé, je voudrais parler de l'idée que j'ai eue pour implémenter le jeu. Mon jeu est constitué d'une matrice qui représente la scène et les joueurs, cette matrice est rafraîchie toutes les 60 secondes.

a) Scène (fichier. ffscene)

Mon fichier « .ffscene » doit contenir exactement 50 éléments, la scène en elle-même sans les joueurs et les obstacles doit comporter le même caractère du début jusqu'à la fin. Les fonctions pour l'implémentation de la scène sont dans la classe jeu.

b) Création de la matrice

Pour créer la matrice j'ai utilisé la manière la plus simple c'est-à-dire 2 for imbriqué mais j'ai aussi utilisé la méthode qu'on avait vu en cours celle avec les deux crochets. Tout le jeu se passe dans cette même matrice. Dans le main est écrit la fonction pour les évènements des touches et un while pour faire tourner le jeu, ensuite une classe jeu où sont implémenté les fonctions qui ont un lien avec la scène du jeu mais aussi du menu puis une classe joueur 1 et une classe joueur 2 avec leurs fonctions.

c) Rafraichissement de ma matrice pour avoir un jeu plus fluide

J'utilise un module de pygame qui est plus précis que le time.sleep, quand le jeu se lance je crée un objet qui me permet d'avoir le temps précis ensuite dans le while en utilisant « clock.tick(fps) » je sais que cette méthode est appelée une fois par frame et elle calcule combien de millisecondes se sont écoulées depuis l'appel précédent. Donc le programme ne s'exécutera jamais à plus de « fps » images par seconde. J'utilise aussi une fonction clear qui me permet d'effacer le chemin qui apparait sur le terminal pour ne laisser place qu'à un écran noir. Sur Vscode j'ai un petit problème si on compile plusieurs fois sur le terminal quand on remonte le terminal on aperçoit le jeu plusieurs fois je pense que c'est parce que clear n'est pas optimisé et à un certain moment il faut relancer un autre terminal.

d) Avancer (move_left) et Reculer (move_right)

Les fonctions ont été implémenter dans les classes joueur 1 et joueur 2, ces fonctions ont le même but que je préciserai juste en dessous et sont presque identique à un détail près qui sont les coordonnées de l'un et de l'autre.
Premièrement pour avancer il suffit juste d'incrémenter la variable coordonnée des 2 joueurs et enfin de supprimer l'emplacement du joueur précédemment. Les joueurs n'ont pas le droit de se traverser.
Deuxièmement pour reculer il suffit de décrémenter la variable coordonnée des 2 joueurs et de supprimer l'emplacement du joueur précédemment et lorsque les joueurs dépasse la limite de la scène ils sont repoussés de 10 cases vers l'avant.
Et enfin dans la fonction « avancer_reculer » dans la classe du joueur 1 et du joueur 2 est implémenté la vitesse à laquelle le déplacement se déroulera par seconde,

si « movement_speed = 1 » et qu'on joue à 60 frames par seconde alors le déplacement du joueur prendra 1/60 seconde.
Les fonctions utilisaient sont les suivantes :

- Avancer_reculer
- Supprimer_avancer
- Supprimer_reculer

e) Sauter à droite (jump_right) et Sauter à gauche (jump_left)

Pour sauter à droite le joueur doit d'abord monter d'une case dans la matrice ensuite avancer d'une case dans la matrice puis redescendre d'une case dans la matrice tout cela en 3* « mouvement_speed ». Entre chaque action qui sont appelé il faut aussi supprimer l'emplacement du joueur pour ensuite utiliser par exemple la fonction « sauter » qui permettra de faire monter le joueur d'une case dans la matrice.
Dans la fonction sauter il y a aussi la détection des obstacles. S'il y a un obstacle on doit sauter d'une certaine manière, regarder à hauteur - 2 s'il y a un obstacle pour ne pas tomber sur lui mais directement avancer encore une fois pour retomber un peu plus loin. Le joueur 1 et le joueur 2 saute différemment lorsqu'il y a un obstacle et ils ne peuvent pas sauter en même temps, pour rajouter un peu plus de difficultés dans le jeu.
Et enfin comme pour avancer et reculer, quand on saute en dehors de la scène on est repoussé de 10 vers l'avant, les 2 joueurs ne peuvent pas se rentrer dedans lorsqu'il saute et il y a aussi un time.sleep qui a le même but que pour avancer et reculer.
Les fonctions utilisaient sont les suivantes :

- Sauter
- Supprimer_avancer_sauter_gauche
- Supprimer_avancer_sauter_droite
- Supprimer_sauter_reculer
- Supprimer_sauter_avancer
- Supprimer sauter
- Avancer_reculer

f) Attaquer

L'attaque des joueurs est la même que dans le sujet du projet. Je change juste dans la matrice la case ou il y a l'épée, ce que je fais exactement est la chose suivante je supprime la case où il y a l'épée, je la remplace par une épée horizontale puis après je remets l'épée comme elle était, j'utilise un time.sleep pour la vitesse de l'attaque mais aussi pour laisser un temps d'attaque pour que l'on puisse voir visuellement que l'épée est horizontal. On peut attaquer simultanément grâce à la variable « mode_attaque » que j'utilise dans la fonction « attaque_meme_temps » définit dans la classe jeu et que j'utilise dans le while du main, de même pour savoir si l'attaque n'est pas bloqué j'utilise la fonction « attaque_defense ». A chaque fois que l'attaque a bloqué le joueur qui attaque recule de 10 cases. A chaque fois que le joueur touche la personne sans qu'elle soit en mode défense, le joueur qui attaque gagne 1 point et le jeu se reset.

Les fonctions utilisaient sont les suivantes :
- supprimer_states
- rétablir_states
- attaquer

g) Défense

J'ai implémenté la défense comme demander dans le sujet. La défense est faite de la même façon que l'attaque, je remplace juste la case ou il y a l'épée par une épée verticale. Il y a aussi un temps d'attente pour que l'on puisse voir visuellement que l'épée est verticale. On peut défendre lors d'une attaque grâce à la variable « mode_defense » que j'utilise dans la fonction « defense_attaque » et « attaque_defense » définit dans la classe jeu et que j'utilise dans le while du main. Quand un joueur défend alors que l'autre attaque, le joueur qui attaque recule de 10 cases. Aucun des joueurs ne gagne de point.
Les fonctions utilisaient sont les suivantes :
- Supprimer_states
- Rétablir_states
- Défense

h) Utiliser les 2 touches en même temps (attaque et défense)

Comme on ne pouvait pas attaquer en même temps et défendre en étant attaqué mon idée était de faire une liste ou lors de l'interaction avec les touches d'attaque et défense, les fonctions sont insérées dedans et lancer dans le while comme une sorte de fonction, je pense que nous avons vu cette méthode dans le cour, ensuite grâce au variable qui précise l'état dans lesquelles sont nos 2 joueurs permet de attaquer en même temps et de pouvoir défendre lorsque que l'on est attaque. Les fonction que j'utilise sont les « fonction_interract_.... » dans la classe jeu

i) Pynput

J'ai décidé t'utiliser pynput pour l'interaction avec le clavier, car il est simple d'utilisation mais ne permet pas d'utiliser les touches en même temps mais comme dit précèdent j'ai trouvé une solution pour y remédier. Pynput utilise des threads et donc on doit attendre qu'une action se termine pour ensuite exécuter l'autre. Les fonctions que j'utilise est implémenter dans le main « on_press » et « on_release ».
Toutes les fonctions préciser ci-dessus sont utiliser dans les fonctions « on_press » et « on_release » car lorsque l'on appuyer sur une touche elle doit exécuter une action.
J'ai aussi eu un problème concernant pynput lors de la compilation de mon code sur Ubuntu que je n'ai pas su régler cependant sur mac j'ai su régler le problème en donnant la permission de regarder les entrées à Python et au terminal dans les paramètres de confidentialité et sécurité du mac.

2/ Improvements

a) Menu Pause

Lorsque l'on appuye sur « ESPACE » du clavier le menu pause se lance ensuite tout est indiqué dans le menu on peut quitter le jeu mais aussi revenir en arrière et aussi voir la liste des commandes pour le joueur 1 et le joueur 2. J'ai implémenté le menu assez facilement en créant une variable pause et commande dans la classe jeu qui permet de savoir respectivement si les joueurs ont appuyé sur pause, si les joueurs regardent la liste des commandes disponibles dans le menu pause et en changeant les valeurs de la matrice et bien sûr en supprimant les valeurs précédentes. Tout est implémenter dans la classe jeu et dans la fonction « on_release » du main pour l'interaction avec les touches du clavier.

Les fonctions utilisaient sont les suivantes :

- Pause_menu
- Reset
- Options
- Effacer
- La variable pause_menu et commande

b) Save et Load

Pour save on utilise la touche « w » qui écrit dans un fichier « save.txt » les coordonnées des joueurs et le score des joueurs.
Pour load on utilise la touche « b » qui d'abord supprime la position des joueurs de départ et ensuite lis le fichier « save.txt » qui affecte les données présentes dans le fichier respectivement aux variables enregistrer précédemment.

c) Version graphique

Dans ma version graphique j'utilise la bibliothèque pygame. Etant donné que j'ai déjà utilisé un peu cette bibliothèque l'apprivoisement a été plutôt simple, je me suis rendu sur YouTube pour voir les cours en ligne qui exister dessus mais je me suis aussi rendu sur la page officielle de pygame ou il existe de la documentation sur les fonctions présentes dans pygame. Voici le site : https:// www.pygame.org . J'ai trouvé les images sur github et sur google image.
Pour compiler le main il faut d'abord installer pygame en faisant : pip3 install pygame
J'ai le même nombre de classe que dans la version terminale c'est-à-dire une classe jeu, joueur 1 et joueur 2 mais j'ai aussi rajouté une classe bouton car j'ai fait un menu pause. Pour mettre en pause le jeu il faut aussi appuyer sur « ESPACE ». Les commandes pour jouer sont les mêmes que sur le terminal. Je n'ai juste pas ajouté d'obstacles sur cette version.

3/ Bonuses

a) Le son

Le son a été rajouté grâce à la bibliothèque pygame et la fonction « mixer.music » et « mixer.Sound ».
Dans la version terminal et graphique lors du passage au menu_pause une musique de fond se lance grâce à « mixer.music »
Dans la version terminale lorsque les personnages marchent cela produit un bruitage grâce à « mixer.Sound »

b) Github

Voici le lien du github : https://github.com/razal563/escrime_jeu

4/ Les difficultés rencontrées

Avant de débuter le projet je ne savais pas comment commencer et ce que j'allais faire exactement ensuite j'ai eu l'idée de faire une matrice. L'apprivoisement de la bibliothèque pynput a été plus longue car je ne savais pas comment et ou placé les listeners mais en cherchant dans la documentation j'ai trouvé comment faire et comment cela fonctionner. Je trouve aussi que la partie ou il fallait trouver comment faire lorsque les 2 joueurs appuyer sur une touche du clavier en même temps était plutôt rude, c'est ce qui m'a pris le plus de temps. Finalement j'ai réalisé le projet et je l'ai trouvé intéressant car il m'a appris de nouvelle chose.