Documentation projet RUSH HOUR

Rigaud Clara et Léré Gwladys - L2S3 MIASHS

Notre projet est un jeu appelé Rush Hour qui consiste à déplacer des voitures sur une grille afin de faire sortir la voiture rouge. C’est une sorte de parking et les voitures ne peuvent bouger que dans un sens (hoizontal ou vertical). Le but est donc de dégager le passage devant la voiture rouge pour la faire sortit du parking. Nous avons 40 grilles de jeu, toutes de la même taille mais avec un nombre de voitures différent et une difficulté croissante. L’utilisateur peut choisir la grille sur laquelle il veut jouer ou il peut jouer directement dans l’ordre. Chaque joueur a sa partie, ses scores et ses grilles déjà effectuée. Les scores sont ensuite comparés afin d’afficher les meilleurs.

Nous avons réalisé ce projet en python, avec une version console mais aussi une version graphique avec tkinter. Nous avons également eu recours au format json pour stocker les grilles récupérées directement sur internet, les joueurs et leurs scores.

1. Fichier grid.py

Le fichier grid.py contient la classe Grille qui correspond à la grille de jeu. L’objet de type grille va contenir des objets de type car, c’est donc la grille qui contient les voitures.

Dans le \_\_init\_\_ nous récupérons un numéro de grille en paramètre et nous allons chercher la grille de jeu correspondant dans le fichier grilles.json. Nous l’utilisons comme des dictionnaires.

Ensuite, pour chaque voiture dans la grille, on utilise le type car dont nous parlerons plus bas pour contenir toutes leurs informations.

toArray nous permet de récupérer un objet de type grille et de renvoyer un tableau afin d’utiliser ezCLI pour l’affichage de la grille.

move nous permet de vérifier si le mouvement demandé par le joueur peut se faire ou pas. Elle prend en paramètres un objet de type car, et une str correspondant à la direction donnée par l’utilisateur. Ainsi nous vérifions si la voiture existe, si la direction demandée est possible, si la voiture reste dans la grille et si elle ne rentre pas en collision avec une autre voiture. La fonction renvoie un str correspondant au type d’erreur. Les coordonnées de la voiture changent et si cela entraine une erreur, les coordonnées sont rechangées pour avoir celles de base.

On appelle ici les fonctions carIsOutOfGrid et IsRoadFree de la classe grid et les fonctions canMove et movement de la classe car.

carIsOutOfGrid vérifie si le mouvement entré par l’utilisateur fait sortir la voiture de la grille.

isRoadFree sert à vérifier s’il y avait déjà une voiture à cet emplacement. Elle va appeler la fonction crashes qui appartient à la classe car. Elle retourne False s’il y a une collision.

win vérifie si la voiture sort du tableau et donc si l’utilisateur a gagné.

hasCar vérifie si le nom de la voiture pris en paramètre correspond à une voiture de la grille.

1. Fichier car.py

Ce fichier contient la classe Car qui correspond aux voitures. Chaque voiture est un objet de type Car et elles sont toutes stockées dans un objet de type Grille.

Dans la partie \_\_init\_\_ on donne les valeurs de la voiture, sa position en x et y, sa direction (verticale ou hoizontale), sa hauteur et sa largeur. Elle prend en paramètres le dictionnaire contenant les informations de la voiture et son nom.

canMove vérifie si la direction entrée par l’utilisateur correspond au sens de mouvement de la voiture. Elle est appelée dans la fonction move de la classe grid.

crashes vérifie s’il y a une collision entre deux voitures. Elle est appelée dans la fonction isRoadFree de la classe grid.

movement effectue le mouvement demandé sur la voiture. Elle a un paramètre avec une valeur par défaut (back), si on ne le spécifie pas elle fait le mouvement, sinon elle fait le mouvement inverse. Cela permet d’effectuer le mouvement demandé pour faire les vérifications et si on se rend compte que le mouvement est impossible, on l’annule.

1. Fichier players.py

Ce fichier contient la classe players avec toutes les fonctions permettant de gérer les joueurs et leurs scores (dont 3 ne faisant pas partie de la classe players).

Dans le \_\_init\_\_ on charge les scores joueur s’il a un compte, on initialise un tableau de scores sinon.

Hasplayed vérifie si le joueur à un score associé au numéro de grille.

Islogged vérifie si le joueur est enregistré.

Saveinfile enregistre le joueur et ses nouveaux scores dans le fichier json.

Setname permet de changer le nom du joueur.

Set\_player\_grid\_score modifie le score du joueur sur la grille jouée si son score est meilleur que l’ancien.

set\_player\_grid\_score renvoie le score du joueur pour une grille donnée.

Get\_player\_points renvoie une liste des points du joueur pour chaque grille

Sync syncronise les scores de deux joueurs. Elle est utilisée dans la fonction qui permet d’enregistrer la partie.

Playernames renvoie les noms des joueurs contenus dans le fichier json.

Get\_players\_points renvoie les points des joueurs.

Get\_grid\_best\_score renvoie une liste contenant le meilleur score sur une grille donnée.

1. Le fichier main\_console.py

Il permet de lancer le jeu en console.

Error\_msg sert à afficher les erreurs. Une erreur est affichée selon le code donné en paramètre.

Start\_level est la boucle de jeu sur la grille. Elle s’occupe de gérer les entrées de l’utilisateur pour le mouvement, l’appel des fonctions pour effectuer le mouvement, ou encore l’appel de la fonction si l’utilisateur veut ouvrir les options. Elle appelle également la fonction win\_menu en cas de victoire.

playerchoice charge un joueur par rapport au nom entré par l’utilisateur.

gridchoice permet à l’utilisateur de choisir une grille.

classement effectue le classement quand l’utilisateur veut voir les scores et l’affichage.

Scoreslist appelle classement et prend en compte le retour au menu.

newplayer permet de créer un nouveau joueur.

save permet d’enregistrer la partie en cours.

quitgame permet de sauvegarder la partie ou non avant de quitter.

win\_menu permet l’affichage de la fenetre après une victoire et l’appel des fonctions selon le choix de l’utilisateur.

Level\_option\_menu permet l’affichage du menu pendant le jeu en cours, si l’utilisateur entre « OP ». Elle appelle les fonctions nécessaires.

Main\_menu permet l’affichage du menu principal et l’appel des fonctions correspondantes.

1. Le fichier graphique.py

Ce fichier correspond à la classe Graphic qui est utilisée dans le fichier main\_graphic.py et comprend toutes les fonctions liées au déroulement du jeu en graphique, l’affichage de la grille, le déplacement des voitures et la gestion des obstacles.

Dans le \_\_init\_\_ on définit toutes les variables qui vont être utiles par la suite, la taille de la grille voulue et la largeur des blocs. On définit un canvas dans une fenetre puis on le remplit avec un dammier. Ensuite on ajoute les voitures avec une fonction dont nous parlerons plus bas et on lie le canvas à l’événement « clic gauche » pour lancer une fonction quand l’utilisateur l’effectue.

colorCars donne une couleur à chaque voiture.

cars permet l’affichage des voitures dans le canvas.

clic permet de récupérer la position du clic de l’utilisateur et d’appeler la fonction pour bouger.

outOfGrid vérifie si la voiture est en dehors de la grille.

crashes vérifie s’il y a une collision.

isRoadFree vérifie si la voiture ne touche pas une autre en appelant la fonction crashes

moveCar effectue le déplacement des rectangles sur le canvas, donc des voiutures. Elle vérifie donc sur quel voiture a cliqué l’utilisateur puis sur quelle partie de la voiture pour connaitre le sens de mouvement. Ensuite elle change les coordonnées du rectangle dans le tableau et s’il n’y a pas d’obstacles, elle bouge le rectangle. Sinon elle remet les coordonnées du rectangle à zéro dans le tableau.

win vérifie si le joueur a gagné et change de fenetre si c’est le cas.