COMPTE RENDU Sujet 4 Algorithmique

J’ai choisi le sujet 4 « Propagation du virus », j’avais plusieurs objectifs.

Tout d’abord, alors qu’il y avait déjà une distinction entre les agents (autonomes et non-
autonomes) je devais en créer une autre mais entre les agents autonomes (les contaminés et
les sains) qui allait se faire par la couleur et diminuer leur état de fonctionnement dans un premier temps. Ce que j’ai réussi à faire sans souci, mais ce sont les étapes d’après qui sont un peu plus ardues, malheureusement pour la propagation du virus, et l’immunité que confère la zone de vaccination sont des choses que je n’ai pas réussi à gérer, j’ai hésité entre trop de solutions : faire des agents une classe dont un des attributs aurait été « isContaminated »,, gérer la contamination dans une fonction spécifique ou la gérer dans la fonction « deplacement ». À cause de cette hésitation, je n’ai pas pu faire un choix entre toutes ces solutions.

Je me suis occupé de l'affichage des différents paramètres (score, durée de la partie, nombre de pièces ramassées, record).
Pour afficher cela, j'ai du créer des différents canvas que celui du jeu, j'en ai précisément créé un par paramètre, de cette manière j'ai pu les manipuler plus facilement (taille par exemple). Ce que j'ai fait avec « canvas.place » qui nécessite des coordonnées x et y, que j'ai eu grâce un petit bout de programme qui m'a beaucoup aidé à faire cette tâche.

```
838
839     # Coordonnées de la position de la souris
840     def callback(e):
841         x= e.x
842         y= e.y
843         print("Coords pointeur : %d, %d" %(x,y))
844     fen_princ.bind('<Motion>',callback)
845     #Rafraichissement de la fenêtre et de tout son contenu
846     fen_princ.mainloop()

Coords pointeur : 276, 232
Coords pointeur : 281, 233
Coords pointeur : 285, 236
Coords pointeur : 296, 241
Coords pointeur : 304, 245
Coords pointeur : 318, 254
Coords pointeur : 331, 262
Coords pointeur : 344, 270
Coords pointeur : 359, 280
Coords pointeur : 373, 289
Coords pointeur : 380, 294
```

Ceci m'a permis d'être très précis dans le placement de mes canvas, mais il y a comme inconvénient que je doive reconfigurer les canvas et leurs items à chaque fois.

Pour afficher convenablement le temps écoulé entre le début de la partie et la fin du jeu, j'ai calculé l'heure de début, grâce à ce module :

```
from timeit import default_timer
```

Et cette fonction :

```
starttime = default_timer()
```

Et pour obtenir le temps écoulé en temps réel sans utiliser de boucle, nous pouvons effectuer ce calcul dans une fonction qui est souvent utilisée, comme la fonction « deplacement »

```
# temps actuel
currentTime =int(default_timer() - starttime)
temps.itemconfigure(vraitemps, text="%d" %currentTime)
```

Pour gérer les records, et l'enregistrement des scores précédents, je l'ai fait avec un fichier texte, ce qui est possible avec le module json,

```
import json
```

```
# ouverture du fichier texte à la fin de la partie
fichier = open("scores.txt", "w")
# écriture du résultat
fichier.write(json.dumps(score_tot))
# fermeture du fichier
fichier.close()
```

Résultats :

Lorsque la partie se finit, le score est bien stocké, mais cette valeur est écrasée par la suivante. Même si je suis loin d'avoir fini, j'ai beaucoup aimé le faire, j'ai posé souvent posé des questions au professeur que ce soit en TP/TD ou par mail qui me répondait avec calme et bienveillance. Je vais continuer chez moi, même si il n'y a plus de notes.