WARGON Solal
LE MASSON Arthur

Projet NSI Groupe F: Snake

Pour notre projet de NSI, nous avons décidé de coder le jeu Snake.
Nous avons rencontré de nombreuses difficultés, surtout Solal, qui a dû coder les déplacements du serpent.

I. Les difficultés de codage

Il fallait bien commencer par quelque part, donc nous avons commencé par le menu de jeu. Le menu était notre première expérience avec le module d'interface graphique Tkinter et ce fut donc assez long puisque nous devions apprendre comment le module marchait (les mots clés, les fonctions, …). Nous voulions que la fenêtre prenne la taille de l'écran de l'ordinateur et c'est la que nous avons rencontré notre première difficulté : comment faire ? Après des recherches sur internet et ses nombreux forums, nous avons découverts qu'il était possible de les calculer sur Tkinter. Nous avons donc écrit ces lignes :

screen_width = menu.winfo_screenwidth()
screen_height = menu.winfo_screenheight()

Cependant il restait toujours un problème : le paramètre de la fonction *geometry* d'une fenêtre est écrit entre guillemets et le programme ne comprenait pas si on écrivait :

menu.geometry("screen_heightxscreen_width")

Nous avons donc fini par trouver la réponse (toujours sur internet) :

sw = str(screen_width)
sh = str(screen_height-60)
menu.geometry("{}x{}".format(sw,sh))

La seconde difficulté, que nous avons vite surmonté, était d'importer une image et de la centrer. Nous l'avons fait comme cela :

```
sw = str(screen_width)
sh = str(screen_height-60)
scale_w = int(int(sw)/660)
scale_h = int(int(sh)/330)
photo=PhotoImage(file='snakeimage.gif',height=cdrH_1,width=cdrW_1)
photo=photo.zoom(scale_w, scale_h)
img=cdr.create_image(cdrW_1,cdrH_1,image=photo)
```

A l'aide d'un bouton « Jouer », nous arrivions sur le jeu

Après avoir créé un fond pour le jeu, il a fallu entrer dans le vif du sujet, et coder les déplacements du serpent.
Il fallait tout d'abord créer un corps au serpent, en définissant des coordonnées aux blocs composant le serpent. Ce n'était pas la partie la plus compliquée, mais ce n'était pas non plus la plus simple : il fallait faire apparaître les cubes et les faire partir dans la bonne direction. Puis, nous avons dû coder les déplacements du serpent : ce fut la partie la plus compliquée :

Nous avions commencé par coder 4 fonctions qui codait le déplacement quand on appuyait sur une flèche :

```
def up(event):
  a=1
  global x,y
  intface.coords(head,x,y-10,x+10,y)
  y=y-10
```

Il y avait donc 4 fonctions comme celle-ci mais le serpent se déplaçait seulement quand on appuyait sur la flèche et son mouvement n'était pas continu. Nous avons naturellement tenté une boucle *while* telle que :

```
def up(event):
  a=1
  global x,y
  while a=1 :
    intface.coords(head,x,y-10,x+10,y)
     y=y-10
```

Cette boucle ne marchait pas : l'ordinateur faisait l'opération trop vite sans jamais s'arrêter. Nous avons eu la solution en cours et avons codé la fonction *deplacement()* qui figure dans notre programme

Ensuite, il fallait réussir à faire se déplacer le corps du serpent. Nous avons rencontré une grande difficulté ici, car il fallait donner différentes coordonnées et différents noms à chaque bloc de serpent ce qui est impossible quand le nombre de carré est inconnu. Nous avons donc pensé à faire des listes : une liste pour les abscisses de chaque carré(dot_x), une pour les ordonnées(dot_y) et une pour le corps du serpent où nous entrions chaque carré.

Enfin, il fallait animer le serpent pour que son corps et sa tête se déplacent.
Nous avons dû chercher un moyen de décaler les listes de coordonnées de 1 pour que chaque carré puisse prendre les coordonnées de celui d'avant. Solal a vraiment eu du mal, et a essayé de nombreuses choses, comme par exemple dot_x[f]=dot_x[f+1] avant de finir par trouver la solution, en regardant les manipulations possibles sur une liste et en prenant une feuille pour résoudre mathématiquement et algorithmiquement le problème. Nous sommes arrivés à cette conclusion (dans l'exemple ci-dessous, quand le serpent va vers le haut) :

```
dot_x.reverse()
dot_x.append(x)
del dot_x[0]
dot_x.reverse()

dot_y.reverse()
dot_y.append(y-10)
del dot_y[0]
dot_y.reverse()
```

-Ensuite, il fallait définir les coordonnées de collision entre le serpent et les autres éléments du jeu : les pommes, les murs, et éventuellement lui-même :

Il a ensuite fallu définir la localisation des pommes et l'agrandissement du serpent à chaque fois qu'il en mangeait une. Cette partie nous a posé un petit problème, qui était d'agrandir le serpent en plaçant le carré aux bonnes coordonnées en fonction de la direction du dernier carré ce qui nous a forcé à créer une liste avec la direction de chaque carré.
Il fallait définir tous les cas de fin de partie : le serpent cogne un mur ou se mord la queue Cette partie ne fut pas la plus difficile car nous savions déjà comment il fallait faire et nous n'avons pas dû repenser le programme. De plus nous savions mieux coder qu'au début du projet, le seul problème étant de ne pas se tromper lorsqu'on définissait les coordonnées pour lesquelles on perdait si le serpent y était.

-Enfin, il fallait créer une fonction « reset » pour pouvoir rejouer, ce qui n'était pas si compliqué, il suffisait de recopier les premières lignes initialisant le programme

II. Les autres erreurs liées au codage

Nous avons également eu quelques difficultés dans le codage, mais qui n'étaient pas forcément présentes uniquement sur le jeu Snake :

-Nous ignorions un certain nombre de fonctions et mots clé, qui auraient pu nous aider à avancer plus vite dans notre projet (par exemple nous avons eu des difficultés pour donner les mêmes variables à tout le code jusqu'à découvrir l'existence du « global »). C'est aussi pour ça que plus le code avançait, plus nous allions vite, ayant moins besoin de faire des recherches et comprenant de mieux en mieux le module.

-Une des difficultés était également de ne rien oublier (en tout cas, le minimum de choses), donc il fallait ne rien négliger pour pouvoir rajouter des «if » en cas d'oublis (par exemple, il fallait bien définir toutes les conditions de directions du serpent et les conditions de fin de partie) .

-Enfin, il y avait une partie un peu longue de débogage, car même si la majeur partie erreurs était indiquée par l'ordinateur, il fallait tout revérifier. Une des difficultés majeures du débogage était de trouver également les erreurs non signalées par l'ordinateur... Par exemple, nous avons perdu du temps à cause de l'inversion d'un + et d'un =, qui faisaient que le serpent ne pouvait s'agrandir de plus de 3. L'avancée du projet fut bloquée pendant une semaine avant de comprendre d'où venait l'erreur.

```
f=+1 au lieu de f+=1
```

Annexe : première version du menu