Jeu du serpent : Tkinter

Règles du jeu : un serpent constitué d'une tête et d'anneaux se déplace sur la scène.

Lorsqu'il arrive sur un bord, il réapparait de l'autre côté.

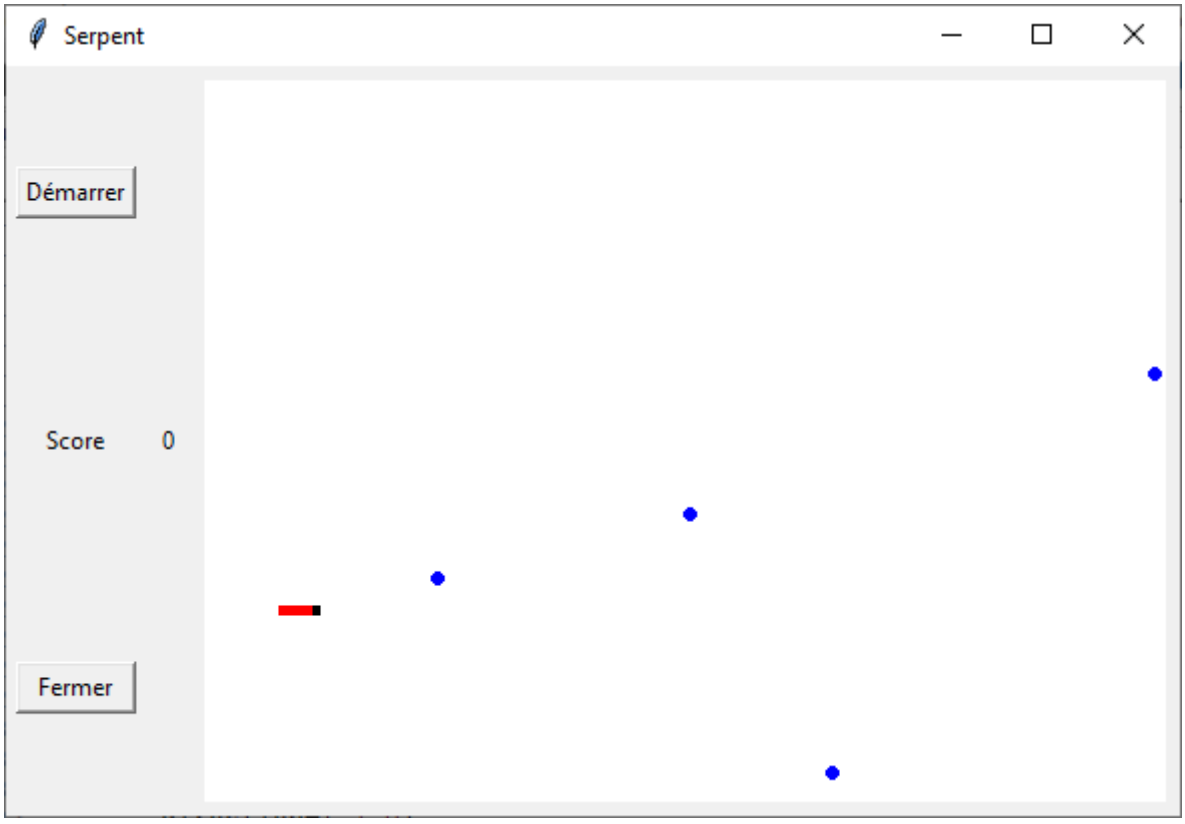
Les flèches haut, bas, à droite ou à gauche permettent de changer la direction du serpent.

Sur la scène sont placées des pommes. Lorsque le serpent mange une pomme, il marque un point et son corps s'allonge.

Si le serpent se mord la queue, le score est diminué de 3 points.

Le jeu s'arrête lorsqu'on clique sur le bouton fin.

La partie démarre lorsqu'on clique sur le bouton démarrer.



Les objets :

La tête est un carré noir de 4 x 4 pixels, et les pommes sont des cercles s'inscrivant dans un carré de 6 x 6 pixels.

L'anneau est identique à la tête, mais il est de couleur rouge.

La mécanique du jeu

Une liste **position_serpent** contiendra **les coordonnées [x, y]** de chaque élément du serpent : en tête de liste, les coordonnées de la tête du serpent, puis les coordonnées de chaque anneau constituant la queue.

Une liste **serpent** contiendra **une référence à chaque élément du serpent**. Les coordonnées de chacun de ces éléments se trouvent dans la liste position_serpent, au même indice.

Une liste **pomme** contiendra **une référence à chacune des pommes** créées pour le jeu.

Une liste **position_pomme** contiendra **les coordonnées [x, y]** de chacune des pommes créées.

Nous devons également disposer de 8 variables :

largeurFond=480 : largeur du fond (scène)

hauteurFond=360 : hauteur du fond

sx=4 : dimension en largeur de la tête.

sy=4 : dimension en hauteur de la tête.

Ces variables permettent de modifier facilement les dimenssions de la tête et des anneaux.

px=6 : dimension en largeur d'une pomme

py=6 : dimension en hauteur d'une pomme

Ces variables permettent de modifier facilement les dimenssions de la tête et des anneaux.

direction=[1,0] #le serpent démarre vers la droite. Indique la direction dans laquelle le serpent se déplace.

sc=0 : score.

La construction de l'interface et l'initialisation du jeu

```
import tkinter as tk
import traceback
from random import randint
try :
#=================================================
    largeurFond=480
    hauteurFond=360
```

```
sx=4
sy=4
px=6
py=6
direction=[1,0] #le serpent démarre vers la droite
sc=0
#---------------------------------------------
ecran=tk.Tk()
ecran.title("Serpent")
#bouton pour lancer le jeu
demarrer=tk.Button(ecran,text="Démarrer", command = debutPartie)
demarrer.grid(row=0,column=0, padx = 5, pady = 5)
#Labels Score
LabelScore= tk.Label(ecran, text="Score" )
LabelScore.grid(row=1,column=0, padx = 5, pady = 5)
textA=tk.StringVar()
textA.set("0")
Score= tk.Label(ecran, textvariable=textA )
Score.grid(row=1,column=1, padx = 5, pady = 5)
#bouton pour fermer la fenêtre
finPartie=tk.Button(ecran,text="Fermer",command = finPartie)
finPartie.grid(row=2,column=0,sticky='ew', padx = 5, pady = 5)
#un canvas
fond = tk.Canvas(ecran, width=largeurFond,height=hauteurFond,bg="white")
fond.grid(row=0,column=2, rowspan=3, padx = 5, pady = 5)
#-----------------------------------------
#position de départ de la tête du serpent : valeurs aléatoires
position_serpent=[[randint(40,largeurFond-40),randint(40,hauteurFond-40)]]
```

```
 #Création de la tête du serpent : sa référence est placée dans la liste serpent [ ]
    serpent=[fond.create_rectangle(
                                    position_serpent[0][0],
                                    position_serpent[0][1],
                                    position_serpent[0][0]+sx,
                                    position_serpent[0][1]+sy,
                                    fill="black",
                                    outline="black")]
#Création des anneaux
    ajouterAnneaux()
#Création des pommes
    pomme=[]
    position_pomme=[]
    #créer 4 pommes
    for i in range(4):
        #les oordonnées de chacune des pommes sont aléatoires.
        xp = randint(px,largeurFond-px)
        yp = randint(py,hauteurFond-py)
        position_pomme.append([xp,yp])
        #Création d'une pomme et enregistrement de sa référence dans la liste pomme.
        pomme.append(fond.create_oval(
                                    position_pomme[i][0],
                                    position_pomme[i][1],
                                    position_pomme[i][0]+px,
                                    position_pomme[i][1]+py,
                                    fill="blue",
                                    outline="blue"))
```

```
    #-----------------------------------------
    #rendre l'écran sensible aux évènements provenant des flèches du clavier.
    ecran.bind("<Up>",serpent_up)
    ecran.bind("<Down>",serpent_down)
    ecran.bind("<Right>",serpent_right)
    ecran.bind("<Left>",serpent_left)
    ecran.mainloop()
except:
   #ce bloc permet de récupérer des infos en cas d'erreur
   traceback.print_exc()
```

Les fonctions Call back

```
def serpent_up(event):
        global direction
        direction=[0,-1]

    def serpent_down(event):
        global direction
        direction=[0,1]

    def serpent_right(event):
        global direction
        direction=[1,0]

    def serpent_left(event):
        global direction
        direction=[-1,0]

def debutPartie():
      gestionSerpent()
```

```
def finPartie() :
    ecran.destroy()
```

Les fonctions outils

Déplacer le serpent

```
def deplacerSerpent():
    global position_serpent, sx, sy
    x=position_serpent[0][0]
    dirx=direction[0]*sx
    y=position_serpent[0][1]
    diry=direction[1]*sy
    #calcul de la nouvelle position de la tête
    newPos = [x+dirx,y +diry]
    #La position d'un point du serpent prend la precedente position
    #du point qui le precede
    #insérer la nouvelle position en tête de la liste des positions
    position_serpent.insert(0,newPos)
    #Supprimer la dernière position
    position_serpent.pop()
    #Mise à jour du dessin sur l'écran : on modifie les coordonnées de chaque
    #objet de la liste serpent en fonction des valeurs contenues dans la liste
    #position_serpent
    i=0
    while i<len(position_serpent):
        fond.coords(serpent[i],position_serpent[i][0],
                    position_serpent[i][1],
                    position_serpent[i][0]+sx,
                    position_serpent[i][1]+sy)
        i+=1
```

```
    #vérifier si la tête a touché un bord de l'écran
    #le serpent doit ressortir de l'autre côté.
    x=position_serpent[0][0]
    y=position_serpent[0][1]
    if x <= 0:
        position_serpent[0][0]=480
    if x >= 480:
        position_serpent[0][0] = 0
    if y <= 0:
        position_serpent[0][1] = 360
    if y>= 360:
        position_serpent[0][1] = 0
```

Ajouter des anneaux

```
def ajouterAnneaux():
    global position_serpent, serpent,direction,sx, sy
    #Position du dernier élément du serpent
    x=position_serpent[len(position_serpent)-1][0]
    y=position_serpent[len(position_serpent)-1][1]
    #Suivant la direction de la tête les coordonnées des nouveaux
    #anneaux changent
    dirx=direction[0]*sx
    diry=direction[1]*sy
    #on ajoute 4 anneaux
    for i in range(4):
    #les anneaux du serpent sont placés derrière la tête
        x = x-dirx
        y = y-diry
        position_serpent.append([x,y])
```

```
    serpent.append(fond.create_rectangle(
                        position_serpent[i+1][0],
                        position_serpent[i+1][1],
                        position_serpent[i+1][0]+ sx,
                        position_serpent[i+1][1]+sy,
                        fill="red",
                        outline="red"))
```

Tester une collision

Cette fonction doit permettre de tester une collision entre la tête du serpent et une pomme, mais aussi entre la tête du serpent et un anneau de sa queue.

Nous devons donc faire le test entre la tête du serpent et chacune des pommes de la liste pomme, et la tête du serpent et chaque anneau de la liste serpent. Pour les anneaux nous ne commencerons les tests qu'à partir du troisième anneau.

Comme il faut déplacer la pomme touchée s'il y a eu collision, la fonction renverra une valeur indiquant s'il y a eu collision, et dans le cas d'une pomme, elle renverra le numéro de la pomme touchée.

```
def detectCollision(list1, list2, d1,d2,debut):
    xs=list1[0][0]
    ys=list1[0][1]
    i=debut
    #on compare les coordonnées de la tête du serpent avec les coordonnées
    #des éléments de la deuxième liste.
    while i<len(list2):
        xp=list2[i][0]
        yp=list2[i][1]
        if xs > xp+d2 or xs+d1<xp or ys>yp+d2 or ys+d1 < yp:
            i+=1
        else:
            return (True,i)
    return (False,i)
```

Les paramètres de la fonction désignent : les deux listes sur lesquelles on travaille :

list1 sera la liste **position_serpent** et list2 sera la liste **position_pomme** ou la liste **position_serpent**, suivant que l'on teste une collision entre la tête du serpent et une pomme, ou avec un anneau.

d1 et d2 désigne les largeurs des objets des 2 listes.

d1 correspondra à **sx** et d2 à **px** ou **sx**, suivant que l'on teste une collision entre la tête du serpent et une pomme, ou avec un anneau.

début désigne le numéro du premier élément de la liste 2, avec lequel on débute les tests.

La fonction principale du jeu

```
#-----------------------------------------------------
    def gestionSerpent():
        global position_pomme, pomme, sc
       #déplacer le serpent
       deplacerSerpent()
       #tester si le serpent est entré en collision avec une pomme.
        #si collision serpent-pomme, déplacer la pomme et allonger le serpent
        #Le numéro de la pomme touchée est renvoyé par la fonction detectCollision
        test, i= detectCollision(position_serpent, position_pomme,sx,px,0)
        # s'il y a eu collision
        if test:
          #calcul de nouvelles coordonnées pour la pomme i
           xp=randint(6,largeurFond-px)
           yp=randint(6,hauteurFond-py)
           position_pomme[i]=[xp,yp]
           #Mise à jour de l'interface avec ces nouvelles coordonnées
           fond.coords(pomme[i],
                              position_pomme[i][0],
                              position_pomme[i][1],
                              position_pomme[i][0]+px,
```

```
                    position_pomme[i][1]+py,
                    )
    #ajouter des anneaux car il y a eu collision avec une pomme
    ajouterAnneaux()
   # ajouter 1 au score et afficher le nouveau score
    sc+=1
    textA.set(str(sc))

#tester si le serpent se mord la queue
test, i = detectCollision(position_serpent, position_serpent,sx,sx,3)
#s'il y a eu collision entre la tête du serpent et un anneau
if test:
    #on diminue le score de 3 points
    sc-=3
    textA.set(str(sc))

#La fonction se rappelle elle-même toutes les 50 ms
#En jouant sur cette valeur, on fait varier la vitesse du serpent
fond.after(50,gestionSerpent)
```