Séance 9 : Animations et algorithmes

L1 – Université Côte d'Azur

Exercice 1 — Apprendre à exprimer ses émotions (⋆)

On considère le code ci-dessous disponible à l'adresse :
https://upinfo.univ-cotedazur.fr/~obaldellon/L1/py/tp9/ex1.py

```
import tkinter as tk
from math import *
(Hauteur,Largeur) = (300,300)
root = tk.Tk()
root.title("Exprimons nos émotions")
Dessin = tk.Canvas(root,height=Hauteur,width=Largeur,bg='white')
Dessin.pack()

def disque(x,y,r,couleur):
    p = (x+r,y+r)
    q = (x-r,y-r)
    Dessin.create_oval(p,q,fill=couleur)

class État():
    def __init__(self):
        self.yeux=20
        self.affichage()

    def affichage(self):
        Dessin.delete('all')
        # Tête
        (xt,yt)=(Largeur/2,Hauteur/2)
        Rt=Hauteur*.9/2
        disque(xt,yt,Rt,'yellow')
        # Œil à droite de l'image
        (xd,yd)=(2*Largeur/3,Hauteur/3)
        disque(xd,yd,self.yeux,'black')
        # Œil à gauche de l'image
        (xg,yg)=(Largeur/3,Hauteur/3)
        disque(xg,yg,self.yeux,'black')
        # Bouche
        (xe,ye)=(Largeur/2,2*Hauteur/3)
        (Rex,Rey)= (40,30)
        Dessin.create_oval(xe-Rex,ye-Rey,xe+Rex,ye+Rey,fill='yellow',width=5)
        (xr,yr)=(Largeur/2,2*Hauteur/3-20)
        (Rrx,Rry)=(70,20)
        Dessin.create_rectangle(xr-Rrx,yr-Rry,xr+Rrx,yr+Rry,fill='yellow',width=0)

état=État()
root.mainloop()
```

1. Recopiez le code précédent (ou téléchargez-le, ce sera plus rapide !). Lancez-le. Que fait-il ? Comment arrive-t-on à obtenir la bouche ?

   Le programme affiche un visage. Pour obtenir la bouche, il affiche une ellipse puis on cache la partie supérieure de l'ellipse avec un rectangle jaune, ce qui donne l'illusion d'un sourire.

2. Rendez le visage triste en changeant les coordonnées `(xr,yr)` du rectangle.

   Il suffit de prendre `(xr,yr)=(Largeur//2, 2*Hauteur//3+20)`

3. Ajoutez un attribut `heureux` à la classe `État` qui prend pour valeur un booléen (`True` ou `False`). Modifiez la méthode `affichage` pour qu'elle affiche un visage triste ou heureux suivant la valeur de cet attribut.

```
class État():

    def __init__(self):
        self.yeux=20
        self.heureux=True
        self.affichage()

    def affichage(self):
        Dessin.delete('all')
        # Tête
        (xt,yt)=(Hauteur//2,Largeur//2)
        Rt=Hauteur*.9/2
        disque(xt,yt,Rt,'yellow')
        # Bouche
        (xe,ye)=(Largeur//2,2*Hauteur//3)
        (Rex,Rey)= (40,30)
        Dessin.create_oval(xe-Rex,ye-Rey,xe+Rex,ye+Rey,fill='yellow',width=5)
        if self.heureux:
            d=-20
        else:
            d=20
        (xr,yr)=(Largeur//2,2*Hauteur//3+d)
        (Rrx,Rry)=(70,20)
        Dessin.create_rectangle(xr-Rrx,yr-Rry,xr+Rrx,yr+Rry,fill='yellow',width=0)
        # Œil à droite de l'image
        (xd,yd)=(2*Largeur//3,Hauteur//3)
        disque(xd,yd,self.yeux,'black')
        # Œil à gauche de l'image
        (xg,yg)=(Largeur//3,Hauteur//3)
        disque(xg,yg,self.yeux,'black')

```

4. Ajoutez deux boutons : « Content », « Pas content », qui changent la valeur de l'attribut `état.heureux` et relance l'affichage. Ainsi, vous pouvez contrôler le sourire de votre visage.

```
def rendre_heureux():
    état.heureux=True
    état.affichage()

def rendre_malheureux():
    état.heureux=False
    état.affichage()

bouton1 = Button(root,text="Bientôt les vacances",command=rendre_heureux, width=20)
bouton2 = Button(root,text="Interro surprise !",command=rendre_malheureux, width=20)

bouton1.pack()
bouton2.pack()
```

5. Ajoutez un curseur pour gérer la taille des yeux. On fera aller la taille des yeux de 0 à 100.

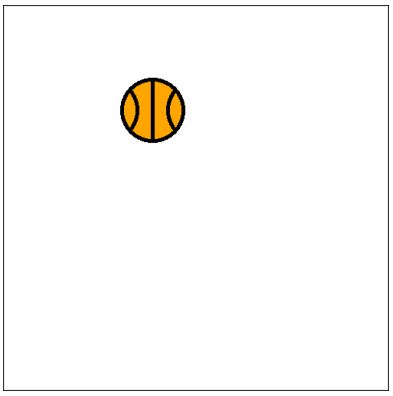
```
def ouvrir_les_yeux(x):
    état.yeux=int(x)
    état.affichage()

curseur = Scale(root, orient = "horizontal", command=ouvrir_les_yeux, from_=1, to=100,
                length=Largeur,label='Épaisseur')
curseur.set(état.yeux)

curseur.pack()
```

6. Que se passe-t-il lorsque les yeux deviennent trop grand ? Corrigez ce bug.
   Lorsque les yeux sont trop grands, il font apparaître le rectangle. Il suffit alors de tracer la bouche avant les yeux. On peut aussi arrêter la taille des yeux à 50 dans le curseur.

Exercice 2 — Un exercice plein de rebondissement (⋆⋆)

L'objectif est de créer une balle rebondissant sur les bords de la fenêtre. On travaillera dans une fenêtre de dimensions 500×500 et on prendra une balle de rayon 50 pixels. On vous laissera choisir le vecteur vitesse initial.

```
def ballon(x,y,r):
    p=(x-r,y-r) ; p1=(x-2.5*r,y-r) ; p2=(x+2.5*r,y-r)
    q=(x+r,y+r) ; q1=(x-.5*r, y+r) ; q2=x+.5*r, y+r
    Dessin.create_oval(p,q,fill='orange',outline='black',width=5)
    Dessin.create_line((x,y-r),(x,y+r),width=5)
    Dessin.create_arc(p1,q1, extent=80, start=-40, width=5,style='arc')
    Dessin.create_arc(p2,q2, extent=80, start=140, width=5,style='arc')
```



1. Créez un objet `État` avec deux attributs (`x` et `y`) pour les coordonnées de la position et deux autres (`xv` et `yv`) pour les coordonnées du vecteur vitesse. On écrira les deux méthodes `__init__` et `affichage`. On pourra utiliser la fonction `ballon` définie ci-dessus.

```python
class État:

    def __init__(self):
        self.x = Largeur//2 # Pour commencer au
        self.y = Hauteur//2 # centre de l'image
        self.vx = 2.25 # Pour la vitesse, on met
        self.vy = 1.75 # ce qu'on veut
        self.rayon=40
        self.pause=False # Pour la question 5
        self.attrapé=False # Pour la question 6
        self.affichage()

    def  affichage(self):
        Dessin.delete('all')
        ballon(self.x,self.y,self.rayon)

état=État()
```

2. Créer une fonction `tictac` qui sera appelée toutes les 10ms et qui modifiera la position de la balle suivant le vecteur vitesse.

```python
def tictac():
    état.x = état.x + état.vx
    état.y = état.y + état.vy
    état.affichage()
    Dessin.after(10,tictac)

tictac() # Il faut l'appeler une fois pour le lancer.
```

3. On veut maintenant que la balle rebondisse sur le bord de la fenêtre plutôt que de disparaître à jamais dans la solitude infinie des pixels hors limites. À chaque fois que le bord de la balle (et non son centre) rencontre le bord de la fenêtre, il faudra modifier le vecteur vitesse. *Indice : par exemple, si la balle touche le bord gauche, on changera le signe de vx.*

```python
def tictac():
    if état.x+état.rayon>=Largeur or  état.x-état.rayon<=0:
            état.vx=-état.vx
    if état.y+état.rayon>=Hauteur or état.y-état.rayon<=0:
            état.vy=-état.vy
    état.x = état.x + état.vx
    état.y = état.y + état.vy
    état.affichage()
    Dessin.after(10,tictac)

tictac()
```

4. Ajouter le code nécessaire pour mettre en pause lorsque l'utilisateur appuie sur la touche espace.

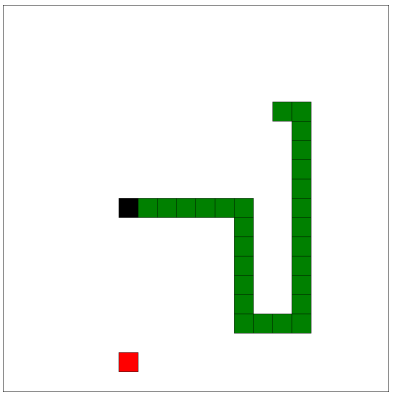
```python
def tictac():
    if not état.pause:
        if état.x+état.rayon>=Largeur or  état.x-état.rayon<=0:
            état.vx = -état.vx
        if état.y+état.rayon>=Hauteur or état.y-état.rayon<=0:
            état.vy = -état.vy
        état.x = état.x + état.vx
        état.y = état.y + état.vy
    état.affichage()
    Dessin.after(10,tictac)

def pause(event): # l'argument est obligatoire
    état.pause = not état.pause

root.bind('<space>',pause)
tictac()
```

5. Question bonus : Modifiez votre programme pour que l'utilisateur puisse attraper le ballon en cliquant dessus et le déplacer en bougeant la souris. Lorsque le ballon est attrapé, sa vitesse devient nulle.

```python
def tictac():
    if not état.pause and not état.attrapé:
        if état.x+état.rayon>=Largeur or  état.x-état.rayon<=0:
            état.vx = -état.vx
        if état.y+état.rayon>=Hauteur or état.y-état.rayon<=0:
            état.vy = -état.vy
        état.x = état.x + état.vx
        état.y = état.y + état.vy
    état.affichage()
    Dessin.after(10,tictac)

def pause(event): # l'argument est obligatoire
    état.pause = not état.pause

def capture(event):
    dx = état.x - event.x # Le clic est sur le ballon si la distance du centre du
    dy = état.y - event.y # ballon à la position de la souris est inférieure au rayon.
    if dx**2 + dy**2 < état.rayon**2:
        état.attrapé = True
        état.x = event.x
        état.y = event.y

def libérée_délivrée(event):
    état.attrapé = False

def migration(event):
    if état.attrapé:
        état.x = event.x
        état.y = event.y

root.bind('<ButtonPress>',capture)
root.bind('<ButtonRelease>',libérée_délivrée)
root.bind('<Motion>',migration) # Pour déplacer le ballon avec la souris
root.bind('<space>',pause)
tictac()
root.mainloop()
```

Exercice 3 — Finir le cours en créant son propre python (⋆ ⋆ ⋆)

Quoi de mieux pour montrer votre maîtrise de Python, que de créer votre propre serpent ! Vous allez, dans cet exercice, implémenter le jeu *snake*. Cet exercice ne sera pas trop guidé. Ce sera à vous de vous inspirer du cours et des exemples précédents pour faire du code qui fonctionne (de préférence correctement).



1. Créez une fonction `mouvement(x,L)` qui ajoute `x` à la fin de liste `L` et qui supprime le premier élément de `L`. *On pourra utiliser la méthode `L.pop(i)`, où `i` indique l'indice de l'élément à supprimer*

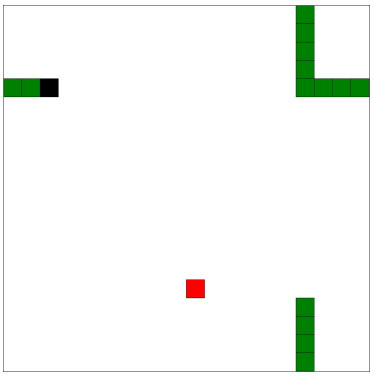
```
>>> L = [ (3, 3) , (3, 4) , (4, 4) ]
>>> mouvement((4,5),L)
>>> L
[(3, 4), (4, 4), (4, 5)]
```

```
def mouvement(a,L):
    L.pop(0)
    L.append(a)
```

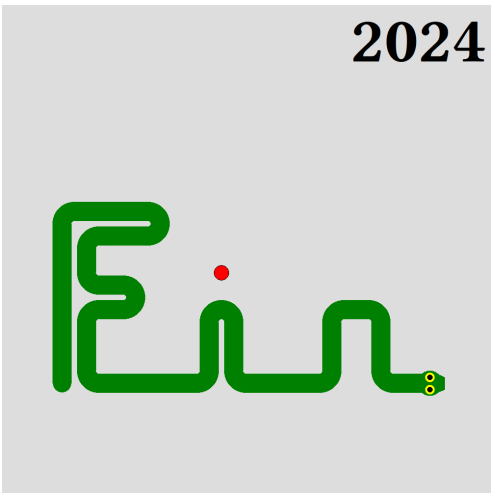
2. Écrivez une fonction `carré(i,j,couleur)` qui affiche un carré à la case `(i,j)`. Évidemment, `(i,j)` ne correspond pas aux coordonnées du pixel mais de la case. Pour simplifier les calculs, on pourra prendre une fenêtre de 1000×1000 pixels que l'on transformera en 20×20 cases.

```
def carré(i,j,c):
    p=(i*D,j*D)
    q=(i*D+D,j*D+D)
    Dessin.create_rectangle(p,q,fill=c)
```

3. Écrivez les méthodes de l'objet `État` (initialisation et affichage). On initialisera le serpent avec un corps de trois cases et la tête sera d'une autre couleur que le corps.
4. Ajoutez le mouvement en écrivant la fonction `tictac`. Puis ajoutez les fonctions pour le contrôler avec les flèches du clavier.
5. Ajoutez une exception si le serpent se mord la queue (c'est-à-dire s'il passe sur une case de son corps). On rattrapera l'exception pour relancer le jeu du début.
6. Quand le serpent quitte la fenêtre, faîtes en sorte qu'il réapparaisse de l'autre côté comme sur l'image ci-dessous. On poura utiliser l'opérateur modulo.



7. Ajouter une pomme aléatoirement sur la fenêtre. Quand le serpent la mange, il grandit d'une case et la pomme re-apparaît aléatoirement.
8. Améliorer votre programme selon votre envie (affichage du score, fichiers de meilleurs scores, accélération du reptile, obstacles, différents types de pomme, bouton pause, etc.)



```python
def incrément(p):
    (i,j)=p
    N=Hauteur//D
    if état.direction=='N':
        return (i,(j-1)%N)
    elif état.direction=='S':
        return (i,(j+1)%N)
    elif état.direction=='O':
        return ((i-1)%N,j)
    elif état.direction=='E':
        return ((i+1)%N,j)

def position_aléatoire():
    return (randint(0,Largeur//D-1),randint(0,Hauteur//D-1))

class État:

    def __init__(self):
        self.direction = 'E'
        self.pomme = position_aléatoire()
        (i,j)= (Hauteur//(2*D), Largeur//(2*D))
        self.L=[ (i-1,j) , (i,j), (i+1,j) ]
        self.affichage()

    def affichage(self):
        Dessin.delete('all')
        # La pomme
        carré(self.pomme[0],self.pomme[1],'red')
        # La queue
        for p in self.L[:-1]:
            (i,j)=p
            carré(i,j,'green')
        # La tête
        (i,j)=self.L[-1]
        carré(i,j,'black')

    def avance(self):
        p=incrément(self.L[-1])
        if p in self.L:
            raise ValueError("Perdu")
        if p==self.pomme:
            self.L.append(p)
            self.pomme = position_aléatoire()
        else:
            mouvement(p,self.L)

    def mange(self):
        p=incrément(self.L[-1])
        self.L.append(p)

print("Création de l'état")
état=État()

```

```python
def haut(event):
    état.direction='N'

def bas(event):
    état.direction='S'

def gauche(event):
    état.direction='O'

def droit(event):
    état.direction='E'

root.bind('<Up>',haut)
root.bind('<Down>',bas)
root.bind('<Left>',gauche)
root.bind('<Right>',droit)


def tictac():
    global état
    try:
        état.avance()
    except:
        # La défaite entraîne une erreur
        # On recommence le jeu
        état=État()
    état.affichage()
    Dessin.after(100,tictac)
```

Pour faire durer le plaisir, vous pouvez chez vous faire un TP bonus sur la résolution du sudoku.

https://upinfo.univ-cotedazur.fr/~obaldellon/L1/py/python-tp-a.pdf