

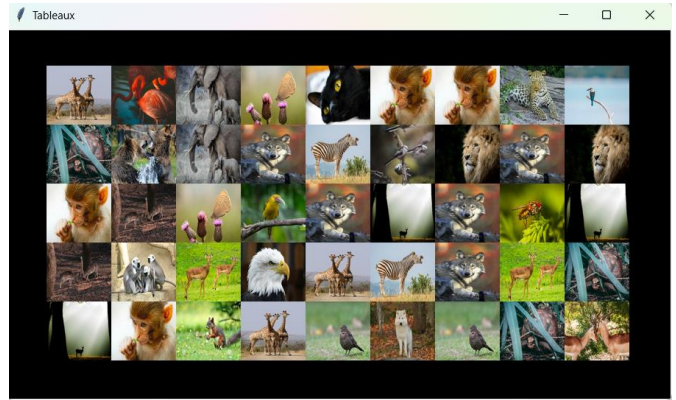
Info 15-B - Tkinter – Tableau dans Canvas

OBJECTIFS : On prolonge ici le tp15-A. On reprend les mêmes objectifs avec les différences suivantes :

- On insère dans le Canvas, non pas des rectangles, mais des images.
- L'évènement souris que l'on déclenche est un évènement click.

1. DEMARRAGE :

⇒ Ouvrir un nouveau fichier, toujours dans le répertoire *tp15* et y copier-coller le script donné ci-dessous. Sauvegardez sous le nom *tp15B.py* . Remettre les indentations pour les fonctions.



```
# Modules -----
from tkinter import Tk , Menu , Canvas , Label , Entry , StringVar ,Button, Text
from PIL import Image, ImageTk # pip install pillow
from random import randint

# Fonctions -----
def creer_fenetre() :
    fenetre = Tk()
    fenetre.title("Tableaux")
    return fenetre

def creer_widgets() :
    zone_graphique = Canvas(fenetre, width=W, height=H , bg = 'black')
    zone_graphique.grid(row = 0 , column = 0)
    return zone_graphique

def dimensions() :
    margeX = 50
    margeY = 50
    largeur = (W-2*margeX) // nColonnes
    hauteur = (H-2*margeY) // nLignes
    margeX = (W-nColonnes*largeur) //2
    margeY = (H-nLignes*hauteur) //2
    return largeur,hauteur, margeX, margeY

# Variables globales -----
W = 900
H = 500
nLignes = 5
nColonnes = 9

# Main -----
fenetre = creer_fenetre()
zone_graphique = creer_widgets()
largeur,hauteur, margeX, margeY = dimensions()

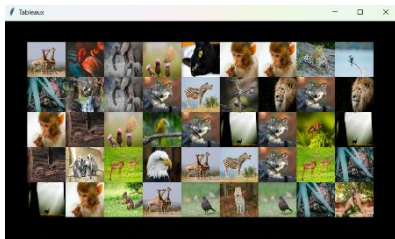
fenetre.mainloop()
```



⇒ Exécuter ce script qui est un extrait de celui réalisé dans le *tp15-A* précédent.

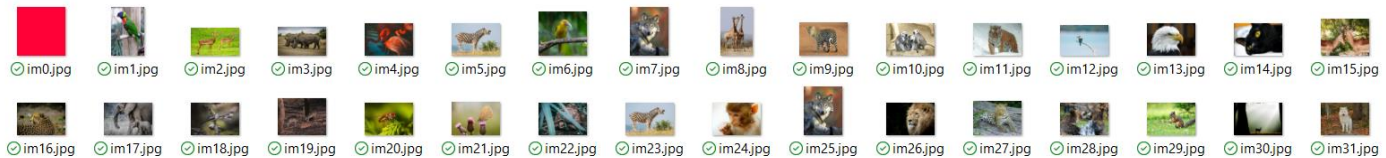
On retrouve la fenêtre tkinter et le Canvas du *tp15-A*.

On se propose dans la suite, de le compléter pour obtenir une mosaïque d'images ...



2. CONVERSION DES IMAGES EN OBJET UTILISABLE PAR TKINTER ET REDIMENSIONNEMENT :

Le répertoire *tp15* contient 33 images :



Leur nom est du type '*im...jpg*'. L'image '*im0.jpg*' est une image rouge de dimension 500x500 pixels. De l'image '*im2.jpg*' à l'image '*im32.jpg*', les tailles sont variables, avec plusieurs centaines de pixels en hauteur et en largeur.

Pour pouvoir réaliser notre mosaïque il est nécessaire de redimensionner toutes ces images et ensuite de les convertir en objet utilisable par tkinter. On réalise ces opérations dans la fonction *creerImagesTk()* donnée ci-dessous.

⇒ On commence par créer 2 variables dans le programme principal : *nbPhotos* pour le nombre de photos et *dicImagesTk* pour le dictionnaire qui contiendra les objets images tk :

```
# Variables globales -----
W = 900
H = 500
nLignes = 5
nColonnes = 9
nbPhotos = 33
dicImagesTk = {}
```

⇒ On écrit le script de la fonction *creerImagesTk()* :

```
def creerImagesTk(largeur, hauteur) :
    for i in range(nbPhotos) :
        fichier = "im"+str(i)+".jpg"
        objPil = Image.open(fichier)
        taille = (largeur, hauteur)
        objPil = objPil.resize(taille)
        objTk = ImageTk.PhotoImage(objPil , master = fenetre)
        dicImagesTk[fichier] = objTk
```



⇒ On appelle *creerImagesTk()* dans le programme principal :

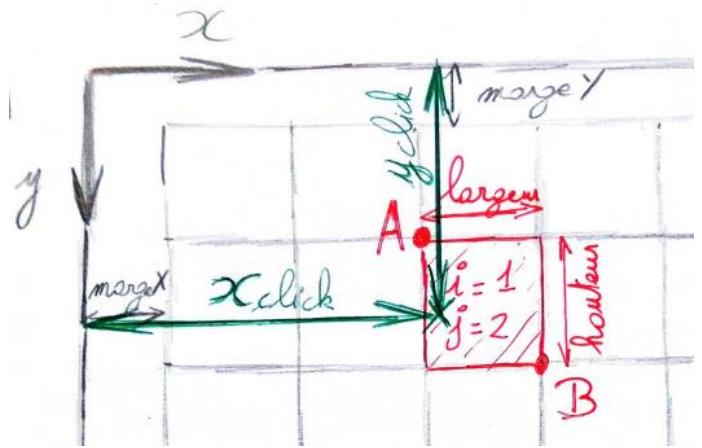
```
# Main -----
fenetre = creer_fenetre()
zone_graphique = creer_widgets()
largeur, hauteur, margeX, margeY = dimensions()
creerImagesTk(largeur, hauteur)
```

3. INSERTION DES IMAGES DANS LE CANVAS :

Dans le tp précédent, on avait inséré des rectangles qui étaient positionnés et dimensionnés en définissant les coordonnées des points $A(x_A, y_A)$ et $B(x_B, y_B)$ définis sur le croquis ci-contre.



Pour une insertion d'images, leurs dimensions étant déjà définies, on gère uniquement le positionnement en définissant les coordonnées d'un point particulier appelé *ancree*.



On choisira dans la suite de prendre comme *ancree* le point milieu de l'image. L'insertion sera réalisée dans la fonction `creer_graphisme()` qui sera pratiquement une copie de celle écrite dans le tpA.

On copie-colle la fonction du tp précédent et on modifie (en rouge ci-dessous) quelques lignes :

```
def creer_graphisme() :
    tab = []
    for i in range(nLignes) :
        ligne = []
        for j in range(nColonnes) :
            xA = margeX + j*largeur
            yA = margeY + i*hauteur
            xB = margeX + (j+1)*largeur
            yB = margeY + (i+1)*hauteur
            num = zone_graphique.create_rectangle(xA,yA,xB,yB,fill = "white")
            ligne.append(num)
        tab.append(ligne)
    return tab

im = dicImagesTk["im1.jpg"]
num = zone_graphique.create_image(xA,yA,anchor="center",image = im )
```

Supprimé, remplacé par

Car *ancree* au milieu

Supprimé, remplacé par

⇒ Après avoir copié-collé la fonction `creer_graphisme()` du tp15A et modifié les lignes identifiées ci-dessus., appeler cette fonction dans le programme principal :

```
# Main -----
fenetre = creer_fenetre()
zone_graphique = creer_widgets()
largeur,hauteur, margeX, margeY = dimensions()
creerImagesTk(largeur,hauteur)
tab = creer_graphisme()

fenetre.mainloop()
```

⇒ Exécuter l'ensemble du script. On obtient bien une mosaïque d'images, mais cette image est toujours la même.

On améliore ce script en affichant une image tirée au sort aléatoirement dans le lot de 32 images (on ne retient pas l'image '`im0.jpg`' que l'on réserve pour l'évènement `click` qui sera vu dans la suite).

Le format utilisé pour définir le nom des fichiers 'im_.jpg' permet de réaliser facilement des boucles.

⇒ On crée une fonction nommée `aleaFichier()` qui tire au sort aléatoirement une image et retourne son nom :

```
def aleaFichier():  
    n = randint(1,nbPhotos-1)  
    fichier = "im"+str(n)+".jpg"  
    return fichier
```

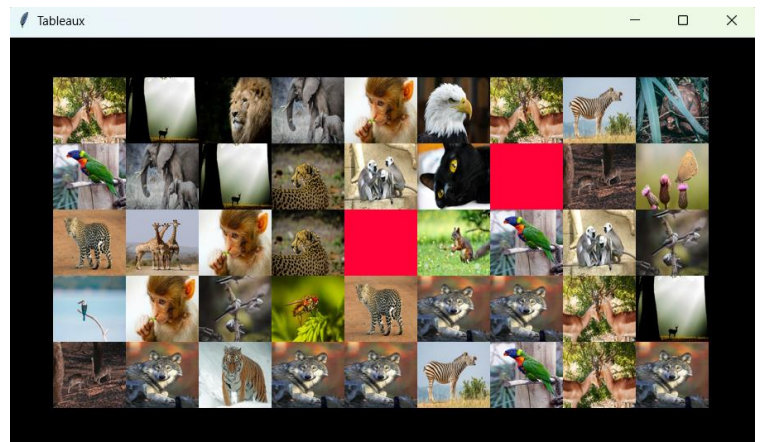
⇒ On met à jour le code de la fonction `creer_graphisme()`. Cela donne finalement :

```
def creer_graphisme() :  
    tab = []  
    for i in range(nLignes) :  
        ligne = []  
        for j in range(nColonnes) :  
            xA = margeX + j*largeur + largeur//2  
            yA = margeY + i*hauteur + hauteur//2  
            fichier = aleaFichier()  
            num = zone_graphique.create_image(xA,yA,anchor="center",image = dicImagesTk[fichier] )  
            ligne.append(num)  
        tab.append(ligne)  
    return tab
```

4. CHANGEMENT D'IMAGE EN CAS DE CLICK :

On se propose à présent, de créer l'évènement suivant :

« En cas de click sur une image de la mosaïque, celle-ci est remplacée par une image rouge. En re cliquant sur une image rouge, l'image initiale revient. »



⇒ On commence par créer cet évènement dans le programme principal. La fonction callback correspondante est appelée `gestionClick()` :

```
# Main -----  
fenetre = creer_fenetre()  
zone_graphique = creer_widgets()  
largeur,hauteur, margeX, margeY = dimensions()  
creerImagesTk(largeur,hauteur)  
  
tab = creer_graphisme()  
zone_graphique.bind("<ButtonPress-1>",gestionClick)  
  
fenetre.mainloop()
```

⇒ On crée la fonction `gestionClick()` et on teste déjà le script :

```
def gestionClick(event):
    x_click = event.x
    y_click = event.y
    print(x_click , y_click)
```

Pour atteindre notre objectif qui est de remplacer une image par une autre, en ayant la possibilité de revenir en arrière ensuite, il est nécessaire pour chacune des cases de la mosaïque, de conserver en mémoire non seulement le numéro d'identification de l'image, mais également le nom du fichier image correspondant.

D'autre part, pour savoir si l'image affichée a été modifiée ou pas, on mémorise une 3^{ème} donnée qui sera un booléen égal à `False` par défaut et égal à `True` si l'image a déjà été modifiée.

On reprend ainsi dans un premier temps le script de la fonction `creer_graphisme()` qui devient à présent :

```
def creer_graphisme() :
    tab = []
    for i in range(nLignes) :
        ligne = []
        for j in range(nColonnes) :
            xA = margeX + j*largeur + largeur//2
            yA = margeY + i*hauteur + hauteur//2
            fichier = aleaFichier()
            num = zone_graphique.create_image(xA,yA,anchor="center",image = dicImagesTk[fichier] )
            ligne.append([num,fichier,False])
        tab.append(ligne)
    return tab
```


Avant on avait : `ligne.append(num)`
A présent, on a : `ligne.append([num,fichier,False])`

⇒ Modifier la ligne identifiée `ligne.append([num,fichier,False])` qui permet d'enrichir les données stockées dans la liste `tab` qui devient ainsi une liste de listes de listes.

Exécuter l'ensemble et fermer ensuite la fenêtre tkinter. Afficher dans la console la liste `tab` : `>>> tab`
... cette liste contient-elle bien les bonnes informations : numéros d'identifications, nom du fichier et booléen ?

⇒ On continue en modifiant la fonction callback de l'évènement :

```
def gestionClick(event):
    x_click = event.x
    y_click = event.y
    i = (y_click - margeY) // hauteur
    j = (x_click - margeX) // largeur
    if 0<=i and i<nLignes and 0<=j and j<nColonnes :
        num = tab[i][j][0]
        if tab[i][j][2] == False :
            zone_graphique.itemconfigure(num , image = dicImagesTk["im0.jpg"])
            tab[i][j][2] = True
        else :
            fichier = tab[i][j][1]
            zone_graphique.itemconfigure(num , image = dicImagesTk[fichier])
            tab[i][j][2] = False
```



⇒ Exécuter et tester l'ensemble.

⇒ Uploader le fichier `tp15B.py` sur nsibranly.fr avec le code **tp13**.