Faire du dessin avec Tkinter

Avec Tkinter, on ne peut pas dessiner ni insérer d'images en dehors d'un canevas :

Le canevas : Canvas

Le canevas représente la surface pour dessiner (= le container de notre dessin).
La ligne de commande peut être la suivante :

```
can = Canvas(fen,bg='black', height=200, width=200)
can.pack()
```

→ can est le nom de notre canvas, *vous pouvez en changer à condition de garder ce même nom à chaque fois que vous y ferez référence.*
→ can.pack() va ajuster notre fenêtre à la taille du canevas.
→ l'option bg='black' donne la couleur du fond de votre canevas.
De nombreuses couleurs sont disponible : red, blue, yellow, white, black, green, gray, pink,orange mais aussi dark gray, dark red, magenta, turquoise, gold, ...

Le canvas est alors muni d'un repère. Pour repérer vos points sur l'écran :

→Le point en haut à gauche a pour coordonnées (0 ; 0),
→le point en bas à droite par exemple dans une fenêtre de 200 * 200 sera le point(200 ; 200),
→le point central a pour coordonnées : (largeur de fenêtre /2 ; Hauteur de fenêtre / 2), *etc.*

Dessiner des lignes

La ligne de commande peut être la suivante : **can.create_line(x1,y1,x2,y2,fill='red', width=5)**

→ Cela permet de tracer une ligne sur notre canvas (can) du point (x1,y1) au point (x2,y2)
→ fill : pour préciser la couleur choisie
→ width : pour préciser la largeur de ligne

Tous les paramètres sont facultatifs, ils doivent être séparés par des virgules. Par défaut, le tracé se fera en noir sans remplissage.

Vous pouvez créer plusieurs lignes en une seule ligne de commande donnant d'autres points de coordonnées (x,y). Par exemple : **can.create_line(x1,y1,x2,y2,x3,y3)**

Dessiner du texte

can.create_text(x=200, y=200, text="mon texte ici", font="Arial 12",fill="cyan")

→x,y : précise le centre du texte à écrire
→font : précise la police choisie
→fill : précise la couleur de l'écriture

Dessiner un rectangle

can.create_rectangle(x1,y1,x2,y2, outline="red ", fill="pink ")

→ (x1,y1) représente le point en haut à gauche, (x2,y2) le point en bas à droite
→ outline =couleur du tour du tracé entre guillemets,
→ fill = la couleur de remplissage entre guillemets

Dessiner une ellipse

can.create_oval(x1,y1,x2,y2,fill='blue', outline='green')
→ (x1,y1,x2,y2) coordonnées du rectangle circonscrit (qui contient l'ellipse)

Donc pour tracer un cercle vous aurez:
can.create_oval(x-rayon,y-rayon,x+rayon,y+rayon,fill='orange', outline='gold')

Dessiner un polygone

C'est comme des tracés de lignes mais le dernier tracé viendra rejoindre le premier par une dernière ligne :
can.create_polygon(x1,y1,x2,y2,x3,y3)

option possible : **smooth=True** #pour arrondir les lignes

*Ouvrir le fichier ***dessin.py*** sur le bureau de l'ordinateur, l'exécuter.Il contient le script ci-dessous :

```
from tkinter import*
from random import randrange

# --- définition des fonctions gestionnaires d'événements : ---
def drawline():
        #Tracé d'une ligne dans le canevas can1
        global x1, y1, x2, y2, coul     # Nouveau ! Permet de récupérer en lecture et en écriture le contenu des variables
                                          citées après l'appel « global » .

        can1.create_line(x1,y1,x2,y2,width=2,fill=coul)
        # modification des coordonnées pour la ligne suivante :
        y2, y1 = y2+10, y1-10

def changecolor():
    #Changement aléatoire de la couleur du tracé
    global coul
    pal=['purple','cyan','maroon','green','red','blue','orange','yellow']
    c = randrange(8)    # => génère un nombre aléatoire de 0 à 7
    coul = pal[c]

#------ Programme principal ------
# Les variables suivantes seront utilisées de manière globale :
x1, y1, x2, y2 = 10, 190, 190, 10     # coordonnées de la ligne
coul = 'dark green'                   # couleur de la ligne
# Création du widget principal ("maître") :
fen1 = Tk()
# Création des widgets "esclaves" :
can1 = Canvas(fen1,bg='dark grey',height=200,width=200)
can1.pack(side=LEFT)
bou1 = Button(fen1,text='Quitter',command=fen1.destroy)
bou1.pack(side=BOTTOM)
bou2 = Button(fen1,text='Tracer une ligne',command=drawline)
bou2.pack()
bou3 = Button(fen1,text='Autre couleur',command=changecolor)
bou3.pack()
fen1.mainloop()
```

A retenir ! La commande **Global** : permet lors de l'appel d'une fonction d'utiliser et surtout de modifier des variables créées dans le programme principal faisant appel à cette fonction.

Exercices *:* Vous allez avoir à modifier le programme précédent.

Après chaque modification, enregistrez le avec un nom différent pour ne pas l'écraser (*save as*).

1. Modifier le programme pour ne plus avoir que des lignes de couleur cyan, maroon et green.

2. Modifier le programme pour que toutes les lignes tracées soient horizontales et parallèles.

Appeler le professeur pour vérification

---

3. Ajouter une fonction drawline2( ) qui tracera deux ligne rouges en croix au centre du canevas : l'une horizontale et l'autre verticale. Ajouter également un bouton portant l'indication « viseur ». Un clic sur ce bouton devra provoquer l'affichage de la croix.

Appeler le professeur pour vérification

---

4. Reprendre le programme initial. Remplacer la méthode create_line par la méthode create_rectangle. Que se passe-t-il ? Qu' indique les coordonnées fournies en paramètres ?
.....
.....

Appeler le professeur pour vérification

On va s'arrêter là pour le dessin !
Pour dessiner un carré à l'écran on utilisera ces outils mais pour créer nos personnages de jeux, nous allons gagner du temps et de la précision en utilisant des outils de dessins (Photoshop, Gimp, ...) pour ensuite simplement importer ces images dans notre canevas.

Pour tout effacer dans un canevas? La commande **can.delete(ALL)**

Insérer une image dans le canevas

Tkinter ne permet pas d'insérer n'importe quelles images, la principale extension acceptée est **.gif**.
Nous allons donc créer ou transformer toutes nos images en GIF afin de pouvoir les utiliser.
La bonne nouvelle c'est que vous pouvez utiliser des images transparentes au format GIF. Apprenez à créer des images transparentes (c'est à dire dont le fond se confondra avec le fond de votre fenêtre quelle que soit sa couleur), pour cela il vous faudra préciser ou cocher "fond transparent" dans votre outil de dessin préféré quand vous créerez vos images.

fichier_img=PhotoImage(file='mon_image.gif') *#pour charger l'image depuis un fichier*
img=can.create_image(x,y,image=fichier_img) *# pour copier l'image sur le canevas*

→x ,y coordonnées du centre de l'image

Exercice :
Ecrire un script ***canevas_image.py*** contenant un canevas de taille 1000 x 1000, de fond jaune, avec au centre de ce canevas l'image ***madame.gif*** *(que vous trouverez sur le bureau de votre ordinateur et que vous copierez dans votre dossier de TP)*.

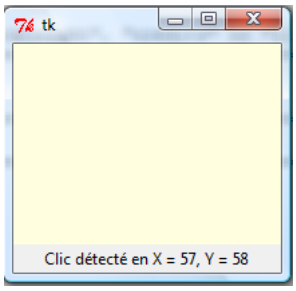
Appeler le professeur pour vérification

Détection et positionnement d'un clic de souris

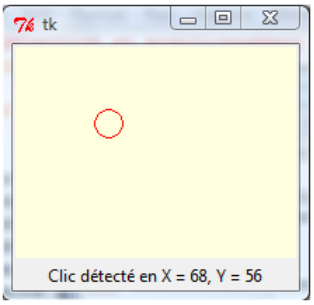
Copier dans votre dossier de travail, ouvrir puis exécuter le programme **test_souris.py** qui se trouve sur le bureau de votre ordinateur dont le script est le suivant :

```
# Détection et positionnement d'un clic de souris dans une fenêtre :
from tkinter import*

def pointeur(event):
        chaine.configure(text="Clic détecté en X = "+str(event.x)+", Y = "+str(event.y))

fen=Tk()
cadre=Frame(fen, width=200, height=150, bg="light yellow")
cadre.bind("<Button-1>", pointeur)
cadre.pack()
chaine=Label(fen)
chaine.pack()
fen.mainloop()
```

Le script fait apparaître une fenêtre contenant un cadre (frame) rectangulaire de couleur jaune pâle.
La méthode ***bind( )*** du widget *cadre* associe l'événement <*clic à l'aide du premier bouton de la souris*> au gestionnaire d'événement « pointeur ».
Ce gestionnaire d'événement peut utiliser les **attributs *x* et *y*** de l'objet ***event*** généré automatiquement par Python, pour construire la chaîne de caractères qui affichera la position de la souris au moment du clic.



Exercice : Modifier le script ci-dessus de manière à faire apparaître un petit cercle rouge à l'endroit où l'utilisateur a effectué son clic
(il faut d'abord remplacer le widget *Frame* par un widget *Canvas*).



Appeler le professeur pour vérification

Le TP en temps limité se termine ici ! Ouf !

Voici une dernière partie pour pouvoir faire des objets animés avec Tkinter. Vous pouvez rendre cette partie en DM facultatif (sur 5) pour la semaine prochaine :

Animation: déplacer un objet

Tkinter vous offre la possibilité de déplacer un objet (une image ou un dessin fait à la main) par la fonction :

can.coords(nom_objet,x1,y1,x2,y2) *# efface et redessine l'objet*

→ nom_objet désigne le nom de l'objet à déplacer,
→ x1,y1,x2,y2 sont les nouvelles coordonnées de l'objet,

cette instruction est magique : elle efface l'objet et le recrée plus loin, sans qu'on ait à écrire ces deux étapes.

Exemple :
Copier dans votre dossier de travail, ouvrir puis exécuter le programme ***balle.py*** qui se trouve sur le bureau de votre ordinateur dont le script est le suivant :

```
from tkinter import*

# Procédure générale de déplacement :
def avance(gd, hb):
        global x1, y1
        x1, y1 = x1+gd, y1+hb
        can1.coords(oval1, x1, y1, x1+30, y1+30)

# Gestionnaire d'événements :
def depl_gauche():
        avance(-10, 0)
def depl_droite():
        avance(10, 0)
def depl_haut():
        avance(0, -10)
def depl_bas():
        avance(0, 10)

##### Programme principal #####
# Les variables suivantes seront utilisées de manière globale :
x1, y1 = 10, 10 # coordonnées initiales
# Création du widget "maître" :
fen1 = Tk()
fen1.title("Exercice d'animation avec Tkinter")
# Création des widgets "esclaves":
can1 = Canvas(fen1, bg='dark gray', height=300, width=300)
oval1=can1.create_oval(x1,y1,x1+30,y1+30,width=2,fill='red')
can1.pack(side=LEFT)
Button(fen1,text='Quitter',command=fen1.destroy).pack(side=BOTTOM)
Button(fen1,text='Gauche',command=depl_gauche).pack()
Button(fen1,text='Droite',command=depl_droite).pack()
Button(fen1,text='Haut',command=depl_haut).pack()
Button(fen1,text='Bas',command=depl_bas).pack()
# Démarrage du réceptionnaire d'événement :
fen1.mainloop()
```

La fonction ***avance( )*** redéfinit les coordonnées de l'objet « cercle coloré » (oval1) à chaque fois que l'on clique sur un des boutons. Ce qui provoque son animation.

Remarque ! Les boutons ont été définis de manière plus compact (pas d'utilisation de variables).

Exercice 1: Modifier le programme précédent de manière à ce que le cercle oval1 se place à l'endroit où l'on clique avec la souris.

Exercice 2 :
Ouvrir et exécuter le fichier **bille.py** que vous trouverez sur le bureau (pensez à le copier dans votre répertoire de travail) puis **compléter les lignes de commentaires** :

```
from tkinter import *

def anime():
  # ……. :
  global x, y
  if x<=250:
    x,y=x+1,y+1
    # ……. :
    canevas.coords(bille,x,y, x+50,y+50)
    #……. :
    fen.after(10, anime)

#### programme principal ####
fen=Tk()
fen.title('animation avec tkinter')
# ……. :
canevas=Canvas(fen,bg='dark gray',height=300, width=300)
canevas.pack()
x,y=0,0
# ……:
bille= canevas.create_oval(x,y,x+50,y+50,fill='orange')

anime()
fen.mainloop()
```

L'instruction **fen.after(10, anime)** redéclenche la fonction anime au bout de 10 millisecondes, comme cette instruction se trouve à la fin de la fonction anime(), elle va se rappeler elle-même, on l'appelle une fonction **récursive** (fonction qui s'appelle elle-même).
On obtient ainsi l'animation attendue avec un affichage toutes les 10 ms.

D'autres instructions sont possibles avec Tkinter comme la détection du clavier, l'ouverture de boîtes de dialogue ou encore la gestion du temps. Vous trouverez des compléments de cours aux liens suivants (qui ont bien servis à faire ce TP) :

http://www.jchr.be/python/tkinter.htm#image

http://fsincere.free.fr/isn/python/cours_python_tkinter.php

http://softdmi.free.fr/tuto_python/tuto-python.php?chapitre=2&page=1

http://www.cgmaths.fr/Atelier/Programmation/InterfacesGraphiques.pdf

http://python.developpez.com/cours/TutoSwinnen/?page=Chapitre8