Interfaces graphiques

Les interfaces graphiques, appelées GUI (graphical user interface), utilisent une bibliothèque appelée tkinter. Il existe d'autres types d'interfaces graphiques, qui sont en général plus compliquées. Citons tout de même *pygame*, qui, comme son nom l'indique, est particulièrement adaptée pour créer des jeux, avec interfaces graphiques, gestion du temps, gestion de la musique. Il est déconseillé d'utiliser pygame tant que vous n'avez pas des notions de base sur les interfaces graphiques. Par ailleurs, l'installation pour usage avec python 3.4 n'est pas forcément simple, la dernière version de python supportée par pygame étant la 3.2, au jour de la rédaction de ce cours (voir par exemple http://stackoverflow.com/questions/28127730/how-to-install-pygame-on-python-3-4 (http://stackoverflow.com/questions/28127730/how-to-install-pygame-on-python-3-4)).
Il est impératif de ne pas mélanger les bibliothèque GUI : pas de tkinter et pygame dans le même programme. A priori, tkinter fait tout ce que fait pygame. Certains conseillent de tout faire avec tkinter, et de n'utiliser aucune autre bibliothèque GUI, quel que soit le programme que l'on souhaite faire.
Les GUI plantent assez facilement, ne vous inquiétez pas si vous voyez apparaitre un message dont la teneur est "dead kernel, will restart automatically". Après quelques instants, ou bien après une relance du programme, une bulle "kernel restart" bleue devrait apparaitre en haut à droite (à côté de python 3). Si ce n'est pas le cas, commencez par sauver votre travail. Puis allez dans l'onglet "Home", et fermez le TD : cocher la case correspondante, "shutdown" en haut de la page. Fermez ensuite l'onglet du TD, et relancez-le à partir de "Home".
Les éléments d'une GUI sont appelés widgets (windows gadgets).

Afficher une fenêtre

L'exemple suivant crée une fenêtre avec saisie, affichage et modification de texte. La fenêtre est crée en dehors du navigateur utilisé pour ce cours. Elle figure sous le nom tkinter ou tk (dans la barre des tâches sous Windows, idem pour les autres systèmes).

In [ ]:
```
from tkinter import *      # import de la bibliothèque graphique

#########  FONCTIONS
def afficher(event):
    """
    Fonction qui transforme le texte dans champ_label2
    @faux parametre : "event" n'est pas un parametre passe en ar
gument,
       cela désigne juste l'evenement qui se produit
       lorsque l'on appuie sur la touche entree
    @faux return : la chaine de caracteres stockee dans champ_la
bel2 est modifiee avec
       le texte recupere par la methode "texte.get()"
    => On ne specifie pas les fonctions GUI comme on le fait pou
r une fonction normale
       - On donne son effet
       - S'il y a des "vrais" paramètres et/ou un vrai "retour",
on le(s) spécifie
       - Sinon on ne met pas de @param ni @return
    """
    champ_label2.configure(text="bonjour "+texte.get())
    return()

#########  PROGRAMME PRINCIPAL
# On crée une fenêtre, racine de notre interface
fenetre = Tk()

# On crée un label (ligne de texte) de titre
# Note : le premier paramètre passé a Label est notre
#        interface racine, la fenetre creee initialement
champ_label = Label(fenetre, text="C tro bo lol !")

champ_label.pack()   # On affiche le label dans la fenêtre, au m
ilieu par defaut

# On affiche un label, puis un champ de saisie
champ_label2 = Label(fenetre, text="Rentrez votre nom")
champ_label2.pack(side = LEFT)  # On affiche le label dans la fe
nêtre, a gauche

texte = Entry(fenetre, width=30)    # le champ de saisie donne u
ne variable nommee texte
texte.bind("<Return>",afficher)     # quand on appuie sur entree
, on apelle fonction "afficher"
texte.pack()                        # On affiche le texte dans l
a fenêtre

# on rajoute un bouton "quitter", avec une commande fenetre.dest
roy au nom assez explicite !
bouton_quitter = Button(fenetre, text="Quitter", command=fenetre
.destroy)
bouton_quitter.pack()     # On affiche le bouton dans la fenêtre

# On démarre la boucle Tkinter qui s'interrompt quand on ferme l
a fenêtre
fenetre.mainloop()
```

Vous remarquez que les widgets (éléments de la GUI) sont "packés", c'est-à-dire positionnés, les uns au dessus des autres, dans l'ordre de leur apparition dans le programme. Diverses méthodes permettent de mieux préciser l'emplacement des différents composants de l'interface.
L'exemple suivant, basé sur la conversion farenheit/celsius, montre l'usage d'une grille (grid)

In [1]:
```
from tkinter import *

##### FONCTIONS

def convertir(event):
    """
    convertit une temperature de fahrenheit en celsius
    """
    fahr=float(temperature.get())
    celsius = 0.55556*(fahr-32)
    champ_label3.configure(text="En degres Celsius cela donne "+
str(celsius))
    return()

##### PROGRAMME PRINCIPAL
fenetre = Tk()        # On crée une fenêtre, racine de notre int
erface

# On crée tous les labels (les textes) ainsi que tous les elemen
ts de la fenetre
champ_label = Label(fenetre, text="Utilitaire de conversion")
champ_label2 = Label(fenetre, text="Rentrez une temperature en d
egres farenheit")
champ_label3 = Label(fenetre, text="")
temperature = Entry(fenetre)
temperature.bind("<Return>",convertir)
bouton_quitter = Button(fenetre, text="Quitter", command=fenetre
.destroy)
    # la commande fenetre.quit devrait fonctionner mais non
# un bouton qui devrait fonctionner ci-dessous mais non, problem
e de typage
#bouton_convertir = Button(fenetre, text="Convertir en degres Ce
lsius", command=convertir(temperature))



# On affiche tout, avec la methode grid qui range les elements d
ans des lignes et colonnes
champ_label.grid(row=0)
#bouton_convertir.grid(row=0, column = 1)    # le bouton qui dev
rait fonctionner...
champ_label2.grid(row=1, sticky = W)
champ_label3.grid(row = 2,sticky = W)
temperature.grid(row=1, column=1)
bouton_quitter.grid(row=2, column = 1)


# On démarre la boucle Tkinter qui s'interrompt quand on ferme l
a fenêtre
fenetre.mainloop()
```

Dessiner dans une fenêtre

L'exemple suivant trace des formes aléatoires simples (segments, ovales, rectangles), dans une fenêtre.

In [4]:
```
#### BIBLIOTHEQUES
from tkinter import *
from random import *

# FONCTIONS
def change_couleur():
    """
    changement aleatoire de la couleur de trace
    """
    pal=['purple','cyan','maroon','green','red','blue','orange',
'yellow']
    c = randrange(8)
    coul = pal[c]
    return(coul)

def choix_alea():
    """
    choix aléatoire de la forme, de sa couleur, de sa position e
t de ses dimensions
    """
    couleur = change_couleur()
    alea = randint(1,3)
    if alea == 1:
        x1 = randint(10,390)
        y1 = randint(10,390)
        x2 = randint(10,390)
        y2 = randint(10,390)
        dessiner_ligne(x1,y1,x2,y2,couleur)
    elif alea == 2:
        x1 = randint(10,320)
        y1 = randint(10,320)
        x2 = x1 + randint(5,70)
        y2 = y1 + randint(5,70)
        dessiner_ovale(x1,y1,x2,y2,couleur)
    else:
        x1 = randint(10,320)
        y1 = randint(10,320)
        x2 = x1 + randint(5,70)
        y2 = y1 + randint(5,70)
        dessiner_rectangle(x1,y1,x2,y2,couleur)
    return()

def dessiner_ligne(x1,y1,x2,y2,couleur):
    """
    trace d'une ligne dans un canevas
    """
    largeur = randint(1,4)
    can1.create_line(x1,y1,x2,y2,width = largeur,fill=couleur)
    return()

def dessiner_ovale(x1,y1,x2,y2,couleur):
    """
    trace d'un ovale dans un canevas
    """
    can1.create_oval(x1,y1,x2,y2,fill=couleur)
    return()
```

Récupérer les coordonnées d'un clic de souris

L'exemple suivant trace une grille dans une fenêtre, puis trace un cercle centré dans la grille au clic de souris (utile pour les plateaux de jeux !).

In [5]:
```
# BIBLIOTHEQUE
from tkinter import *
from random import *

# FONCTIONS

def dessiner_ligne(xA,yA,xB,yB,couleur):
    """
    trace d'une ligne dans un canevas
    entre les points de coordonnees (xA,yA) et (xB,yB), de coule
ur "couleur"
    """
    can1.create_line(xA,yA,xB,yB,width=2,fill=couleur)
    return

def change_couleur():
    """
    changement aleatoire de la couleur de trace
    """
    pal=['purple','cyan','maroon','green','red','blue','orange',
'yellow']
    c = randrange(8)
    coul = pal[c]
    return coul

def trace_grille():
    """
    trace d'une grille vierge dans le canevas
    """
    global can1
    can1.delete(ALL)
    x1 = 10
    y1 = 370
    x2 = 10
    y2 = 10
    coul = change_couleur()
    for i in range(10):
        dessiner_ligne(x1,y1,x2,y2,coul)
        x1 = x1 + 40
        x2 = x2 + 40

    x1 = 10
    y1 = 10
    x2 = 370
    y2 = 10
    coul = change_couleur()
    for i in range(10):
        dessiner_ligne(x1,y1,x2,y2,coul)
        y1 = y1 + 40
        y2 = y2 + 40
    return()


def pointeur(event):
    """
    dessine un rond dans la grille ou un carre
    """
```

Menus

L'exemple suivant montre la construction des menus.

In [7]:
```
from tkinter import *
def donothing():
   filewin = Toplevel(root)
   button = Button(filewin, text="Bouton qui ne fait rien")
   button.pack()
   return()

root = Tk()     # la fenetre s'appelle ici root (racine), le nom
n'a pas d'importance
menubar = Menu(root)    # on construit une barre
filemenu = Menu(menubar, tearoff=0) # le menu fichier n'est pas
detachable
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=donothing)
filemenu.add_command(label="Save as...", command=donothing)
filemenu.add_command(label="Close", command=donothing)

filemenu.add_separator()

filemenu.add_command(label="Quit", command=root.destroy)
menubar.add_cascade(label="File", menu=filemenu)

editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Undo", command=donothing)

editmenu.add_separator()

editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
editmenu.add_command(label="Delete", command=donothing)
editmenu.add_command(label="Select All", command=donothing)

menubar.add_cascade(label="Edit", menu=editmenu)
helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)

root.config(menu=menubar)
root.mainloop()
```

Exercice

Reprendre un programme simple (le jeu "deviner un nombre" est parfait), et le faire avec une interface graphique, qui comprendra aussi bien un menu qu'un bouton pour lancer la partie.

Compléments : gestion du temps et des fenêtres secondaires

L'exemple suivant montre comment créer une fenêtre secondaire, et comment gérer une boucle temporelle. Cet exemple est complexe : vous pouvez le faire tourner immédiatement pour en voir l'effet. Néanmoins, passez à la suite (lecture et écriture dans un fichier) avant de le comprendre et/ou de le modifier, qui présente des connaissances plus importantes à maîtriser. En effet, comme le titre l'indique, ce paragraphe-ci est un complément.

In [ ]:
```
from tkinter import *
from time import sleep      # cette fonction permet d'arreter mo
mentanement un programme

def clavier(event):
    """
    Fonction qui gere l'evenement (event) "frappe sur une touche
"
    Cree une fenetre fille, affiche un message dedans
    et lance une boucle d'animation
    """
    global fen_fille    # la fenetre fille est une variable glob
ale
                        # pourquoi ? je ne sais pas
    global texte        # le texte est une variable globale, pou
r plus
                        # de simplicite
    touche = event.keysym       # recupere la touche
    canevas.unbind("<Key>")      # empeche l'event clavier de se
produire
                                # pendant l'execution de cette f
onction
    fen_fille = Toplevel()      # creation de la fenetre fille
    # ci-dessous affichage d'un label dans la fenetre fille
    taille =12
    texte = "AAAHHH appui interdit sur la touche "+touche
    champ_label = Label(fen_fille, text=texte,
                        font=("Helvetica", taille),fg = "red",
                        bg = "yellow")
    champ_label.pack()
    # appel de la boucle pour animation
    modif_fen(champ_label,taille)
    return()

def modif_fen(champ_label,taille):
    global texte
    # modification du texte
    champ_label.configure(text=texte,font=("Helvetica", taille),
                          fg = "red",bg = "yellow")
    champ_label.pack()
    if taille < 50:     # boucle d'animation
       taille = taille + 1
       # ci-dessous modification de la fenetre apres 20 millisec
ondes
       # observez le passage des parametres de cette methode:
       # d'abord la duree d'attente, puis on reappelle la foncti
on
       # modif_fen (recursivite), et enfin les parametres de mod
if_fen,
       # qui ne sont pas passes comme d'habitude
       fen_fille.after(20, modif_fen,champ_label,taille)
    else:           # fin de boucle d'animation
       sleep(1.5)   # le programme se met en veille 1,5 seconde
       canevas.bind("<Key>",clavier) # on autorise a nouveau le
clavier
       fen_fille.destroy()
    return()
```

Un site qui présente les bases de tkinter, simple et bien fait à la fois: http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel (http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel)
On y trouve d'autres widgets notamment.

Lire / écrire dans un fichier

Commencez par créer un simple fichier texte, avec le bloc-notes, ou Text Edit, contenant deux lignes séparées par un return. Enregistrez le dans le même dossier que ce TD, en format texte, sous le nom "a_modifier.txt".
Le programme suivant vous montre la lecture et l'écriture dans ce fichier.

In [ ]:
```
# Ouverture du fichier a lire et ecrire (mode "r+" read write)

nomFichier = "a_modifier.txt"
fichierInit = open(nomFichier, "r+")

# ouverture ou creation d'un nouveau fichier s'il n'existe pas l
ors du "open"
fichierCopie = open ("copie.txt", "r+")

# affichage du contenu du fichier par ligne
for line in fichierInit:
    print(line)

# repositionnement au debut du fichier et lecture d'une ligne
fichierInit.seek(0)
print(fichierInit.readline())

# Recopie caractere par caractere dans le fichier "copie.txt"
# On pourrait aussi bien faire une copie par ligne, ou globale (
plus rapide)
fichierInit.seek(0)         # repositionnement au debut du fichi
er
for lettre in fichierInit:
    fichierCopie.write(lettre)

fichierInit.seek(0,2)   # positionnement a la fin du fichier
                        # Oieme bit avant la fin (2)
fichierInit.write(str(42))  # on ne peut ecrire que des chaines
de caracteres

print("lecture du fichier intial")
fichierInit.seek(0)
print(fichierInit.read())

print("lecture du fichier copie")
fichierCopie.seek(0)
print(fichierCopie.read())


fichierInit.close() # Fermeture du fichier source
fichierCopie.close() # Fermeture du fichier destination (essenti
el)
```

Exercice

Modifier le programme avec GUI de "deviner un nombre" , de manière à ce qu'il enregistre à chaque partie le nom du joueur, s'il a perdu ou gagné, et dans ce dernier cas le nombre d'essais.

Frederic Mandon (mailto:frederic.mandon@ac-montpellier.fr), Lycée Jean Jaurès - Saint Clément de Rivière - France (2015)