PROGRAMMATION D'INTERFACES

Licence informatique & vidéoludisme

Cours préparé par:
Oumaima EL JOUBARI
Hanane ZERDOUM

UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

Programmation d'interfaces

01

Introduction

La programmation graphique
Le module Tkinter

02

Fenêtres

Création de fenêtres
Gestionnaire de géométrie
Techniques générales

03

Les widgets et évènements

Quelques widgets
Gestion des évènements

04

Pour aller plus loin

Animation
Audios sous Tkinter

01

Les widgets et évènements

1. Quelques widgets
2. Gestion des évènements

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Frame:**

- Le widget Frame est un widget qui sert juste de conteneur, un peu comme une fenêtre Tk.
- Ce type de widget est très utile pour regrouper et organiser des interfaces contenant beaucoup de widgets.

frame.py

```
from tkinter import *

taille = 400
app = Tk()

frame = Frame(app, background="lavender")
frame.pack()

cnv = Canvas(frame, width=taille, \
    height=taille, bg="ivory")
cnv.pack(padx=20, pady=20)

btn = Button(frame, text="Coucou!")
btn.pack(pady=20)

app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Button:**

Ce widget permet de définir un bouton:

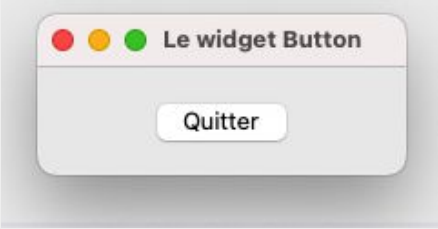
- Un bouton est construit avec le constructeur Button.
- Il faut l'inclure dans son environnement avec une méthode particulière (pack, grid ou).
- Le texte passé dans l'option text est affiché sur le bouton.
- Pour lier une action à un bouton, il faudrait lui passer une option *command*.

button.py

```
#Importer le module tkinter
from tkinter import *

#Creer l'application en tant que classe
class Application(Tk):
    def __init__(self):
        Tk.__init__(self)
    def creerBouton(self):
        texteBouton = "Quitter"
        commandeBouton = app.quit
        bouton = Button(self, text = texteBouton,\
        command = commandeBouton)
        bouton.pack(padx = 15, pady = 15)

app = Application()
app.title("Le widget Button")
app.creerBouton()
app.mainloop()
```



III. Widgets et évènements

1. Quelques widgets

➔ Le widget Button:

Pour changer la couleur du bouton:

- On utilise l'option bg.
- Sur MacOS, la couleur du bouton ne peut pas être changée.

Pour changer la couleur du texte:

- On utilise l'option fg.

```
texteBouton = "Quitter"
commandeBouton = app.quit
bouton = Button(self, text = texteBouton,\
command = commandeBouton, bg='black', fg='white')
bouton.pack(padx = 15, pady = 15)
```

III. Widgets et évènements

1. Quelques widgets

➔ Le widget Button avec une icône:

Sur un bouton, il est usuel de placer du texte. Mais on peut placer une image aussi:

On importe d'abord une image;

Après on utilise l'option *image* au lieu de *text*.

```
logo = PhotoImage(file="red.gif")
bouton = Button(self, image=logo)
bouton.pack(padx = 15, pady = 15)
```

button-icon.py

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Canvas:**

Ce widget permet de créer une zone dans laquelle nous allons dessiner des formes diverses, ou encore insérer d'autres widgets:

- arc, bitmap, image, line, oval, polygone, rectangle, text, window,...

La syntaxe générale: *cnv = Canvas(master, options)*

bd	définir l'épaisseur de la bordure en pixels
bg	définir la couleur du background
cursor	définir le type du curseur
highlight color	la couleur du widget quand il prend le focus
width	la largeur
height	la hauteur

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Canvas:**

Quelques méthodes de la classe Canvas:

- *create_arc*(bbox, options)
- *create_image*(position, options)
- *create_line*(coords, options)
- *create_oval*(bbox, options)
- *create_text*(position, options)

canvas.py

```
from tkinter import *

app = Tk()
cnv = Canvas(app, bg="ivory", \
    height=200, width= 200)
cnv.pack()
cnv.create_oval(0, 0, 200, 200, \
    outline="red", width=10)
cnv.create_line(0, 0, 200, 200,\
    fill="black", width=10)
cnv.create_line(0, 200, 200, 0, \
    fill="black", width=10)
app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Canvas:**

Prise de focus dans un canevas:

- Si un canevas est une surface devant capturer des événements du clavier, il faut lui donner le focus pour qu'il puisse réagir aux touches de clavier.
- La méthode *focus_set()* permet à une widget de prendre le focus.

canvas-focus.py

```
from tkinter import *
from random import randrange

taille=200
app = Tk()
cnv = Canvas(app, width=taille,\
    height=taille, bg="ivory")
cnv.pack(padx=10, pady=10)
cnv.focus_set()

def dessiner(event):
    a=randrange(taille)
    b= randrange(taille)
    cnv.create_rectangle(a, b, a+20, b+2)
cnv.bind("<Key>", dessiner)

app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Label:**

- Label est un widget Tkinter standard utilisé pour afficher un texte ou une image à l'écran.
- Label ne peut afficher du texte que dans une seule police.
- Le texte affiché par ce widget peut être mis à jour à tout moment.

Syntaxe: lab = Label(master, options)

fg	définir l'épaisseur de la bordure en pixels
bg	définir la couleur du background
command	définir la fonction à appeler
font	la police et taille du texte
image	l'image à mettre sur le bouton
width	la largeur
height	la hauteur

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Label:**

Centrer un texte dans un label:

- Un label admet une option d'alignement *justify* permettant le centrage.

Image dans un label:

- Un label peut aussi porter une image au lieu de texte. Il faut remplacer l'option *text* par l'option *image*.

label.py

```
from tkinter import *

app=Tk()
mon_texte="""Je suis un texte long
qui souhaiterait être centré
sur plusieurs lignes."""
annonce=Label(app, height=5, width=50, \
    text=mon_texte, justify=CENTER, \
        bg="ivory")
annonce.pack()
app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Label:**

Utilisation d'une variable de contrôle dans un label:

- L'utilisation d'une variable de contrôle permet de mettre à jour un label suite à un évènement.
- *Exemple:* Quand l'utilisateur clique sur le bouton, un label augmente le compteur d'une unité.

string-var.py

```
from tkinter import *

app=Tk()
msg=StringVar()
entree=Entry(app, textvariable=msg)
entree.pack( padx=20, pady=10)

lbl=Label(app, textvariable=msg)
lbl.pack(padx=20, pady=10)

app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Checkbutton:**

Le widget *Checkbutton* montre du texte et une case à cocher.

Il faut créer autant de widgets *Checkbutton* que de cases à cocher.

Syntaxe: cb = CheckButton(master, options)

title	le titre du widget
active background	la couleur du background sous le curseur
active foreground	la couleur du foreground sous le curseur
bg	la couleur du background

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Checkbutton:**

```
from tkinter import *

app = Tk()
app.geometry("100x100")
var1 = IntVar()
Checkbutton(app,text="Oui",variable=var1)\
    .grid(row=0,sticky=W)
var2 = IntVar()
Checkbutton(app,text="Non",variable=var2)\
    .grid(row=1,sticky=W)

app.mainloop()
```

checkbutton.py

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Entry:**

Le widget Entry est une zone de texte qui permet à l'utilisateur de communiquer avec le programme en lui transmettant des données écrites au clavier (ou par copier-coller).

- Syntaxe: Entry(master, options)

fg	couleur du texte
font	police de texte à utiliser
command	fonction ou méthode à appeler à chaque fois que l'utilisateur modifie l'état du widget
justify	justifier le texte: LEFT, RIGHT, CENTER
show	pour masquer le message écrit par l'utilisateur (ex: saisie de mot de passe). show = « * ».
textvariable	afin de pouvoir récupérer le texte actuel, il faut définir cette option sur une instance de la classe StringVar
xscrollcommand	lier votre widget Entry à une barre de défilement

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Entry:**

```
from tkinter import *

app = Tk()
Label(app,text="Nom").grid(row=0)
Label(app,text="Prénom").grid(row=1)
e1 = Entry(app)
e2 = Entry(app)
e1.grid(row=0, column=1)
e2. grid(row=1, column=1)

app.mainloop()
```

entry1.py

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Entry:**

```
from tkinter import *

app = Tk()
entree = Entry(app)
entree.pack()
def afficher():
    print(entree.get())

bouton=Button(app, text="Afficher", \
    command=afficher)
bouton.pack(padx=50, pady=10)

app.mainloop()
```

entry2.py

III. Widgets et évènements

1. Quelques widgets

➔ Le widget Entry:

Effacer la zone d'écriture:

- Pour effacer la zone d'édition d'une entrée, on utilise la méthode Entry.delete(first,last).
- La méthode supprime les caractères en commençant par celui de l'index *first*, jusqu'au caractère *last*. Si le deuxième argument est omis, seul le caractère à la première position est supprimé.

```
entree.delete(0, END)
```

III. Widgets et évènements

1. Quelques widgets

➔ Le widget Listbox:

Ce widget propose une liste dont les éléments sont sélectionnables à la souris ou avec les flèches Haut et Bas du clavier.

Syntaxe: Listbox(master, options)

width	le nombre de caractères (pas de pixels)
height	le nombre d'entrées devant apparaître dans la liste
selectbackground	la couleur du fond d'une cellule sélectionnée
font	la police utilisée

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Listbox:**

listbox.py

```
from tkinter import *

app = Tk()

liste = Listbox(app)
liste.insert(1, "Bleu")
liste.insert(2, "Rouge")
liste.insert(3, "Vert")
liste.insert(4, "Jaune")
liste.insert(5, "Orange")
liste.insert(6, "Noir")

liste.pack()
app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Listbox:**

Associer une action à un widget:

listbox-action.py

```
from tkinter import *

fruits = ["Litchie", "Kiwi", "Orange", "Raisin", \
    "Banane", "Cerise"]
couleurs = ["pink", "lightgreen", "orange", \
    "purple", "yellow", "red"]
n = len(fruits)

app = Tk()

lbox = Listbox(app, width=8,height=8,\
    font="Verdana 30 bold",selectbackground="blue")
lbox.pack(padx=20,pady=20)
```

```
for item in fruits:
    lbox.insert(END, item)
    lbox.focus_set()
    pos = 1
    lbox.activate(pos)
    lbox.selection_set(pos)
for i in range(0, len(fruits), 2):
    lbox.itemconfigure(i, background="#f0f0ff")
for i in range(1, len(fruits), 2):
    lbox.itemconfigure(i, background="#fff")
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Listbox:**

Associer une action à un widget:

listbox-action.py

```
def show(event):
    index= lbox.curselection()[0]
    cnv["bg"] = couleurs[index]

lbox.bind("<<ListboxSelect>>", show)

cnv = Canvas(app, width =200, height= 200, \
    bg="ivory")
cnv.pack(padx=5,pady=5,side=RIGHT)
cnv["bg"]=couleurs[pos]

app.mainloop()
```

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Radiobutton:**

Typiquement, on utilise le widget Radiobutton dans des situations de choix conduisant à une unique réponse.

Syntaxe: btn = Radiobutton(master, options)

state	DISABLED: pour griser le radiobutton et le désactiver ACTIVE: lorsque la souris est dessus NORMAL: valeur par défaut
selectcolor	la couleur du bouton radio lorsqu'il est défini (rouge par défaut)
image	Image à afficher sur le radiobutton
selectimage	afficher une image différente lorsque le bouton radio est défini
value	donnez à chaque bouton radio du groupe une valeur de type String si la variable de contrôle est StringVar, et une valeur entière si un IntVar
textvariable	définissez cette option sur cette la variable de contrôle StringVar.

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Radiobutton:**

```
from tkinter import *

def sel():
  selected = "Vous avez sélectionné : " \
    + v.get()
  label.config(text = selected)

app = Tk()
v = StringVar()
v.set("Python")
```

```
r1 = Radiobutton(app, text="Python", \
  variable=v, value="Python", command=sel)
r1.pack(anchor = W)
r2 = Radiobutton(app, text="Java", \
  variable=v, value="Java", command=sel)
r2.pack(anchor = W)
r3 = Radiobutton(app, text="PHP", \
  variable=v, value="PHP", command=sel)
r3.pack(anchor = W)

label = Label(app)
label.pack()
app.mainloop()
```

radiobutton.py

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Menubutton:**

Le widget Menubutton est la partie d'un menu déroulant qui reste à l'écran tout le temps.

Chaque MenuButton est associé à un widget Menu qui peut afficher les choix pour ce MenuButton lorsque l'utilisateur clique dessus.

direction	LEFT: pour afficher le menu à gauche du bouton RIGHT: pour afficher le menu à droite du bouton above: pour placer le menu au-dessus du bouton
bitmap	le nom de bitmap à afficher sur le menubutton
image	Image à afficher sur le menubutton
state	DISABLED: pour griser le radiobutton et le désactiver ACTIVE: lorsque la souris est dessus NORMAL: valeur par défaut
text	texte à afficher sur le menubutton.

III. Widgets et évènements

1. Quelques widgets

➔ Le widget Menu:

Le widget Menu permet de créer toutes sortes de menus. La fonctionnalité de base permet de créer trois types de menus: pop-up, toplevel et pull-down.

activeborderwidth	la largeur d’une bordure dessinée autour d’un choix lorsqu’elle se trouve sous la souris.
postcommand	la procédure qui sera appelée chaque fois que quelqu’un affichera ce menu.

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Menu:**

menu.py

```
app = Tk()
def bonjour():
    print("Bonjour tout le monde!")
menubar = Menu(app)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="Nouveau", \
    command=bonjour)
filemenu.add_command(label="Ouvrir", \
    command=bonjour)
filemenu.add_command(label="Enregistrer", \
    command=bonjour)

menubar.add_cascade(label="Fichier", \
    menu=filemenu)

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

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Spinbox:**

Ce widget dispose d'une zone de texte et deux boutons de défilement.

Ce widget permet de faire défiler des éléments dans la zone de texte en progressant dans un sens ou dans l'autre.

from_	La valeur minimale
to	La valeur maximale
values	Un tuple contenant des valeurs valides pour ce widget
textvariable	Cette option est définit sur une instance de la classe StringVar
validate	Mode de validation (NONE par défaut)
width	La largeur du widget déterminée par la taille du caractère affiché
xscrollcommand	Connecter un champ Spinbox à une barre de défilement horizontale

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Spinbox:**

```
from tkinter import *

app = Tk()

sb = Spinbox(app, from_=0, to=15)
sb.pack()

app.mainloop()
```

spinbox.py

III. Widgets et évènements

1. Quelques widgets

➔ Le widget Scrollbar:

Le widget Scrollbar fournit un contrôleur de diapositives qui est utilisé pour ajouter une barre de défilement à des widgets tels que Listbox, Text, et Canvas.

cursor	Le curseur qui apparaît lorsque la souris survole la barre de défilement.
orient	orient = HORIZONTAL pour une barre de défilement horizontale orient = VERTICAL pour une barre verticale
width	Largeur de la barre de défilement (sa dimension y si horizontale et sa dimension x si verticale)

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget Scrollbar:**

```
from tkinter import *

app = Tk()
(variable) scrollbar: Scrollbar
scrollbar = Scrollbar(app)
scrollbar.pack(side = RIGHT, fill = Y)

liste = Listbox(app,yscrollcommand=scrollba
for i in range(200):
    liste.insert(END, str(i) + " Bonjour!")

liste.pack(side = LEFT, fill = BOTH )
scrollbar.config(command = liste.yview )

app.mainloop()
```

scrollbox.py

III. Widgets et évènements

1. Quelques widgets

➔ **Le widget tkMessagebox:**

- Le widget tkMessageBox est utilisé pour afficher des boîtes de message.
- Ce module a un certain nombre de fonctions qui affichent des messages appropriés comme les fonctions: showinfo(), showwarning(), showerror(), askquestion(), askokcancel(), askyesno(), et askretryignore().

Syntaxe: tkMessageBox.FunctionName(title, message [, options])

III. Widgets et évènements

1. Quelques widgets

```
from tkinter import *
from tkinter import messagebox

app = Tk()
def msg():
    messagebox.showinfo("Info", \
        "Bonjour!")

btn = Button(app, text = "Cliquez ici!", \
    command = msg)
btn.pack()

app.mainloop()
```

tkmessagebox.py

III. Widgets et évènements

1. Quelques widgets

➔ **Modifier les options d'un widget:**

Il existe deux syntaxes pour changer une option:

- *widget_name["option"] = new_value*
- *widget_name.configure(option = new_value)*

Il est aussi possible de modifier l'état de manière dynamique en utilisant les variables de contrôle comme StringVar et IntVar.

III. Widgets et évènements

2. Gestion des évènements

➔ Capturer des évènements:

Une interface Tkinter est à l'écoute de certains événements liés à la souris ou au clavier.

Exemples: appuyer ou relâcher une touche du clavier, déplacer le curseur de la souris, cliquer ou relâcher le bouton de la souris.

Il est possible de lier (bind en anglais) un événement à un widget ou fenêtre et aussi définir la fonction à exécuter en utilisant la méthode *bind()*.

III. Widgets et évènements

2. Gestion des évènements

➔ Capturer des évènements:

Exemple de l'utilisation de la méthode bind():

```
from tkinter import *

def f(event):
    t=event.keysym
    print("Touche pressée :", t)
def g(event):
    x=event.x
    y=event.y
    print("Position :", x, y)

app = Tk()
app.bind("<Key>", f)
app.bind("<Motion>",g)
app.mainloop()
```

bind.py

III. Widgets et évènements

2. Gestion des évènements

➔ Évènements du clavier:

Un code d'événement est une chaîne de la forme "<*event*>" où *event* est une chaîne de caractères.

Quelques évènements du clavier:

- Flèches : Left, Right, Up, Down
- ESPACE : space
- Touche ENTRÉE : Return ou, sur le pavé numérique, KP_Enter
- Touche ECHAP : Escape

L'appui sur une touche de caractère, par exemple **a/A**, se nomme **KeyPress-a/KeyPress-A**. On peut abréger le code à seulement **a/A**.

Pour le relâchement d'une touche de caractère: **KeyRelease-a.**

Pour combiner plusieurs touches (ex: ALT+CTRL-a), l'évènement sera nommé: <Control-Shift-KeyPress-a>

III. Widgets et évènements

2. Gestion des évènements

➔ Événements de la souris:

Un widget peut capturer des actions de la souris. Voici un résumé des principaux actions liés à la souris :

Button-1	Clic gauche
Button-3	Clic bouton droit
Button-2	Clic bouton central
Button-4	Molette vers le haut
Button-5	Molette vers le bas
ButtonRelease	Relâchement d’un bouton
Motion	Déplacement du curseur de la souris

III. Widgets et évènements

2. Gestion des évènements

➔ Exemple:

mouse-event.py

```
from tkinter import *

LARGEUR = 480
HAUTEUR = 320

def clic(event):
    X = event.x
    Y = event.y
    r = 20
    cnv.create_rectangle(X-r, Y-r, X+r, Y+r,\
        outline = 'black',fill = 'green')

def effacer():
    cnv.delete(ALL)

app= Tk()
```

III. Widgets et évènements

2. Gestion des évènements

➔ Exemple:

mouse-event.py

```
cnv = Canvas(app, width = LARGEUR, \
    height = HAUTEUR, bg = 'white')
cnv.pack(padx = 5, pady = 5)

cnv.bind('<Button-1>', clic)
cnv.pack(padx =5, pady =5)

Button(app, text = 'Effacer', \
    command = effacer).\
        pack(side = LEFT,padx = 5,pady = 5)
Button(app, text = 'Quitter', \
    command = app.destroy).pack()

app.mainloop()
```

04

Pour aller plus loin

1. Animation
2. Audios sous Tkinter