Chap 19. Tkinter

Livre p 329 – Chap 25 Interaction avec l'utilisateur

1. Programmation événementielle

a. Paradigmes de programmation

Jusqu'à présent on a surtout utilisé de la programmation impérative avec une suite d'instructions à exécuter dans un ordre prédéfini, ainsi qu'un peu de programmation fonctionnelle où les instructions à exécuter sont des fonctions que l'on appelle quand on en a besoin.
Il existe d'autres paradigmes de programmation comme la programmation logique, la programmation orienté objet ou la programmation événementielle.

b. Boucle infinie

Dans programmation événementielle, les instructions sont exécutées lorsque des événements prédéfinis ont lieu, ces événements sont déclarés dans une boucle infinie.

c. Evénements

Les événements peuvent être déclenchés par :

- Un périphérique d'entrée (clavier, souris, …)
- L'horloge interne
- Un élément prédéfini (bouton, curseur, case à cocher, menu, …)

2. Module tkinter

a. Fenêtre principale

```
import tkinter as tk
fenetre = tk.Tk
#
# Evénements à traiter
#
fenetre.mainloop()
```

Comme toute bibliothèque Python, il faut l'importer au début du programme : i*mport* tkinter *as* tk
On doit ensuite créer une fenêtre principale dans laquelle seront placés les « widget » (windows gadget) avec lesquels on pourra interagir :
fenetre = *tk.Tk*
Le principe même de la programmation événementielle est d'attendre que des événement se produisent pour déclencher des actions, il nous faut donc une boucle infinie liée à la fenêtre :
fenetre.*mainloop*()
On peut aussi modifier quelques attributs de la fenêtre principale, par exemple :

fenetre.*wm_attributes*("-topmost", 1) :	La fenêtre *tkinter* restera toujours en premier plan.
fenetre.*title*("Titre") :	Donne un titre à la fenêtre *tkinter*.
fenetre.*destroy*() :	Ferme la fenêtre *tkinter*.

b. Exemples de widget

Gestion d'un événement clavier :	fenetre.*bind*("<*Key*>", fonction)
Gestion d'un événement souris :	fenetre.*bind*("<*Button-1*>", fonction)
Création d'un bouton :	bouton = *tk.Button*(fenetre, *text*="Titre", *command*=fonction)
Création d'une case à cocher :	case = *tk.Checkbutton*(fenetre, *text*="Titre", *variable*=x)
Création d'une case à choisir :	choix = *tk.Radiobutton*(fenetre, *text*="Titre", *variable*=x, *value*=v)
Création d'une liste de choix :	choix = *tk.Spinbox* (fenetre, *text* ="Titre", *value*=liste_choix)
Création d'un curseur :	curseur = *tk.Scale*(fenetre, *from_*=x, *to*=y, *value*=v)
Création d'un menu :	menu = *tk.Menu*(fenetre) suivi de menu.*add_command*(label="Titre")
Création d'une zone de saisie :	saisie = *tk.Entry*(fenetre, *textvariable*=entree, *width*=w)

c. Variables

Attention : Contrairement aux habitudes prises jusqu'à présent en Python, ici les variables doivent être déclarées au préalable avec *tk.StringVar*() pour une chaîne de caractère, *tk.IntVar*() pour un entier ou *tk.DoubleVar*() pour un nombre à virgule flottante… puis on utilise la méthode *get*() pour accéder à leur contenu car ce sont en fait des objets et non réellement des variables que l'on a créés !
Exemple : v = *tk.IntVar*() puis : *print*(v.*get*()) pour voir son contenu.
De plus, on a accès à des variables systèmes liées à l'événement considéré comme par exemple :

event.x et *event.y* : donnent la position de la souris
event.keysym : donne le nom de la touche du clavier appuyée

d. Positionnement des widget

Chaque widget doit être positionné en utilisant la méthode *pack* ou la méthode *grid* :

objet.*pack*(*side* = position) permet de le positionner en haut (*tk.TOP*), en bas(*tk.BOTTOM*), à droite(*tk.RIGHT*), à gauche(*tk.LEFT*).
objet.*grid*(*row*=x, *column*=y, *sticky*=position) permet de le positionner précisément dans une grille en indiquant : la ligne (x), la colonne (y) et la position par un point cardinal (*tk.N, tk.NE, tk.E, tk.SE, tk.S, tk.SW, tk.W tk.NW*) dans la cellule sélectionnée.

e. Gestion du temps

fenetre.*after*(n, fonction) permet d'exécuter une fonction après un délai de n millisecondes.
Remarque : Si l'on souhaite que la fonction soit exécutée toutes les n millisecondes, il faut alors réécrire cette commande dans la fonction elle-même pour relancer le compte-à-rebours à chaque fois qu'il expire !

f. Exemples de fonctions graphiques prédéfinies

Création d'une zone graphique : zone = *tk.Canvas*(fenetre, *width*=w, *height*=h, *bg*=couleur)

Création d'une ligne : zone.*create_line*(x1, y1, x2, y2, *width*=w, *fill*=couleur)
(x1, y1) et (x2, y2) étant les coordonnées des extrémités de la ligne.
Remarques : On peut même donner les coordonnées de plus de deux points pour obtenir une ligne brisée.
Les paramètres d'épaisseur et de couleur sont optionnels, il en existe d'autres permettant d'arrondir les angles, de tracer des pointillé, d'ajouter des flèches, …

Création d'un ovale : zone.*create_oval*(x1, y1, x2, y2, *width*=w, *fill*=couleur)
ou d'un rectangle : zone.*create_rectangle*(x1, y1, x2, y2, *width*=w, *fill*=couleur)
(x1, y1) et (x2, y2) étant les coordonnées des extrémités de la diagonale du rectangle (en haut à gauche, puis en bas à droite) dans lequel l'ovale est inscrit.

Création d'un texte : zone.*create_text*(x, y, *text*="Texte", *fill*=couleur, font=(police, taille))

Import d'une image : nom = *tk.PhotoImage*(*file* = "Fichier.gif")
puis zone.*create_image*(x, y, *image*=nom, *anchor*=position)

Les objets graphiques créés sont de véritables objets que l'on peut ensuite manipuler, par exemple :

zone.*move*(objet, x1, y1) : permet de déplacer un objet dans la zone graphique.
zone.*delete*(objet, x1, y1) : permet de supprimer un objet de la zone graphique.
zone.*itemconfig*(objet, parametre=valeur) : permet de modifier un des paramètres (épaisseur, couleur, …) de l'objet de la zone graphique.

g. Sites de référence

Voici quelques adresses pour aller plus loin :
http://tkinter.fdex.eu/index.html
http://www.jchr.be/python/tkinter.htm
http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel
https://zestedesavoir.com/tutoriels/1729/programmation-avec-tkinter/programmation-evenementielle-avec-tkinter/
http://ftp-developpez.com/michel-aubry/temp/tutoriels/python/tkinter-8-4-reference-interface-utilisateur-graphique-gui-pour-python/tkinter.pdf