

9


## TP SUR LES INTERFACES GRAPHIQUES : LE MOUVEMENT

### Exercice 1 :

L'objectif de cet exercice est de voir comment on peut déplacer un objet dans un canevas en appuyant sur les flèches du clavier.

1. Créer une nouvelle fenêtre Tkinter nommée `fen` (voir TP 6).
2. Créer un canevas nommé `can` de largeur 500, de hauteur 500 et de couleur blanche.
3. On souhaite créer un carré de couleur bleue de côté 20 px dont les coordonnées du point situé en haut à gauche seront nommées  $(x, y)$ .
  - a) On souhaite qu'au départ le carré soit au centre du canevas. Pour cela, affecter à la variable  $x$  la valeur 240 et à la variable  $y$  la valeur 240.
  - b) Ecrire l'instruction suivante qui permet de créer le carré souhaité et l'affecte à une variable nommée `carre` :  
`carre=can.create_rectangle(x,y,x+20,y+20,fill="blue")`

*Rappel : dans la méthode `create_rectangle`, les deux premiers arguments sont les coordonnées du point en haut à gauche tandis que les troisième et quatrième arguments sont les coordonnées du point en bas à droite. L'option `fill` permet de donner une couleur de remplissage au rectangle.*

4. On souhaite maintenant que lorsqu'on appuie sur la touche , le carré se déplace vers la gauche de 10 px.
  - a) Créer la fonction suivante qui permet de diminuer la variable  $x$  de 10 et de déplacer le carré :

```
def gauche(event):  
    global x  
    x = x-10  
    can.move(carre,-10,0)
```
  - b) Rajouter la ligne suivante en-dessous de la ligne de création du carré à la fin du programme :  
`fen.bind("<Left>", gauche)`
  - c) Tester.

### Explications :

L'instruction `fen.bind("<Left>", gauche)` permet de gérer l'évènement "appuyer sur la touche flèche gauche" : quand l'utilisateur appuie sur cette touche, les instructions de la fonction `gauche` sont exécutées. La méthode `bind` prend comme premier argument le type d'évènement, c'est-à-dire un clic de souris ou l'appui sur une touche du clavier. Elle prend comme deuxième argument une fonction qui décrit l'action à effectuer lorsque l'utilisateur clique ou appuie sur la touche indiquée.

La fonction décrivant l'action à effectuer (ici `gauche`) doit toujours avoir comme premier argument `event`. Cet argument est un objet créé par Tkinter qui contient le type d'évènement (appui sur touche ou clic) et plusieurs propriétés (l'instant où s'est produit l'évènement, ses coordonnées, les caractéristiques du widget concerné...).

La méthode `move` permet de faire subir une translation de vecteur  $\vec{u}(-10,0)$  au carré.

5. Ecrire de même les instructions permettant de déplacer le carré de 10 px vers la droite, vers le haut ou vers le bas.
6. On souhaite maintenant améliorer la fonction `gauche`. En effet, si l'utilisateur appuie trop longtemps sur la touche flèche gauche, alors le carré peut sortir du canevas. Afin de résoudre ce problème, on va imposer une condition au déplacement du carré : on ne le déplacera que s'il est suffisamment éloigné du bord gauche du canevas. Dans cet exemple, on ne le déplacera que si l'abscisse du point en haut à gauche est strictement supérieure à 10.

Modifier la fonction `gauche` de la façon suivante :

```
def gauche(event):  
    global x  
    if (x>10):  
        x = x-10  
        can.move(carre,-10,0)
```

7. Modifier de même les fonctions `droite`, `bas` et `haut`.

### Exercice 2 :

L'objectif de cet exercice est de voir comment on peut faire en sorte qu'un objet se déplace de façon autonome (sans intervention de l'utilisateur).

1. Créer une nouvelle fenêtre Tkinter.
2. Créer un canevas nommé `can` de largeur 500, de hauteur 500 et de couleur blanche.
3. Créer un carré de couleur bleue de côté 20 px dont les coordonnées du point situé en haut à gauche seront nommées  $(x, y)$ . Au départ,  $x$  et  $y$  vaudront 10.

4. Créer deux variables  $dx$  et  $dy$  qui représenteront les coordonnées du vecteur de la translation permettant de déplacer le carré. Comme au départ on souhaite que le carré se déplace vers la droite, alors on affectera 10 à  $dx$  et 0 à  $dy$ .
5. On veut maintenant créer la fonction de déplacement automatique du carré.
  - a) Taper la fonction suivante qui permettra au carré de se déplacer de façon automatique :
 

```
def deplacement():
    global x,y,dx,dy
    x=x+dx
    y=y+dy
    can.move(carre,dx,dy)
    can.after(100,deplacement)
```
  - b) Appeler cette fonction en-dessous de l'instruction de création du carré puis tester le programme.

Explications :

Les instructions  $x=x+dx$  et  $y=y+dy$  permettent de modifier les coordonnées du carré selon la translation souhaitée. L'instruction `can.move(carre,dx,dy)` permet ensuite de déplacer le carré selon cette translation. L'instruction `can.after(100,deplacement)` permet de relancer la fonction `deplacement` après 100 millisecondes. La fonction `deplacement` est donc une fonction récursive puisqu'elle s'appelle elle-même et elle se relancera indéfiniment jusqu'à ce qu'on ferme la fenêtre Tkinter.

- c) On souhaiterait maintenant que lorsque le carré arrive trop proche du bord droit du canevas, alors le carré change de direction et aille vers le bas. Pour cela, modifier la fonction `deplacement` de la façon suivante :
 

```
def deplacement():
    global x,y,dx,dy
    x=x+dx
    y=y+dy
    if(x>470):
        x=470
        .....y=y+10
        dx=0
        dy=10
    can.move(carre,dx,dy)
    can.after(100,deplacement)
```

Explications :

Si  $x$  devient strictement supérieur à 470, cela signifie qu'on est arrivé trop proche du bord droit et que le carré doit donc aller vers le bas, donc on remet  $x$  à 470, on augmente  $y$  de 10 et on change les coordonnées du vecteur de déplacement afin que le carré puisse se déplacer vers le bas.

- d) Créer une deuxième condition afin que si le carré arrive trop bas ( $y > 470$ ), alors le carré change de direction et aille vers la gauche.
  - e) Créer une troisième condition afin que si le carré arrive trop à gauche, alors le carré change de direction et aille vers le haut.
  - f) Créer une quatrième condition afin que si le carré arrive trop haut, alors le carré change de direction et aille vers la droite.
6. On souhaite maintenant pouvoir lancer et arrêter le déplacement au moyen de boutons.
    - a) Supprimer l'appel à la fonction `deplacement()` situé à la fin du programme.
    - b) Créer d'abord une variable `Lancé` initialisée à 0 qui vaudra 1 lorsque le déplacement du carré est lancé et qui vaudra 0 lorsqu'on l'arrête.
    - c) Créer une fonction `demarrer()` ne prenant aucun paramètre, qui, si la variable `Lancé` est à 0, modifie sa valeur en 1 et appelle la fonction `deplacement` (si `Lancé` vaut 1 au début de la fonction, cela signifie que la fonction `deplacement` est déjà lancée et donc il ne faut pas la relancer dans ce cas).
    - d) Créer un bouton qui permet de démarrer le déplacement.
    - e) Créer une fonction `arreter()` ne prenant aucun paramètre et qui modifie la variable `Lancé` en 0.
    - f) Créer un bouton qui permet d'arrêter le déplacement.
    - g) Modifier la fonction `deplacement` afin qu'elle ne se relance que si `Lancé` est à 1 (rajouter la condition juste avant l'instruction `can.after(100,deplacement)`.)
    - h) Tester.