Fiche mémo — Tkinter : Menus & événements souris

Cette fiche complète la fiche « Raccourcis clavier ». Idéale pour le projet (menus, context menus, interactions souris).

---

1) Menus d'application (barre de menus)

```
menu_principal = tk.Menu(fenetre)

# Menu Fichier
menu_fichier = tk.Menu(menu_principal, tearoff=0, postcommand=None)
menu_fichier.add_command(label="Nouveau", command=nouveau_fichier)
menu_fichier.add_separator()
menu_fichier.add_command(label="Quitter", accelerator="Ctrl+Q",
command=fenetre.destroy)
menu_principal.add_cascade(label="Fichier", menu=menu_fichier)

# Menu Outils (ex: modules)
menu_outils = tk.Menu(menu_principal, tearoff=0)
menu_outils.add_command(label="Calculs bin./hex.", command=ouvrir_module_calculs)
menu_outils.add_command(label="Opérations logiques",
command=ouvrir_module_logique)
menu_principal.add_cascade(label="Outils", menu=menu_outils)

# Menu Aide
menu_aide = tk.Menu(menu_principal, tearoff=0)
menu_aide.add_command(label="Aide", accelerator="F1", command=afficher_aide)
menu_aide.add_command(label="Contexte", command=afficher_contexte)
menu_principal.add_cascade(label="Aide", menu=menu_aide)

fenetre.config(menu=menu_principal)
```

Notes utiles :

- `tearoff=0` évite le trait pointillé au tout début du menu.
- `postcommand=` permet d'exécuter une fonction **juste avant l'ouverture** du menu (utile pour activer/désactiver dynamiquement des items selon l'état).

---

2) Éléments spéciaux de menu

```
etat_son = tk.BooleanVar(value=True)
mode_vue = tk.StringVar(value="hexa")

menu_options = tk.Menu(menu_principal, tearoff=0)
menu_options.add_checkbutton(label="Activer le son", onvalue=True, offvalue=False,
```

```
variable=etat_son)
menu_options.add_radiobutton(label="Affichage binaire", value="bin",
variable=mode_vue)
menu_options.add_radiobutton(label="Affichage hexadécimal", value="hexa",
variable=mode_vue)

# Désactiver / activer un item par son index
menu_options.entryconfig(0, state="disabled")   # désactive le 1er item
menu_options.entryconfig(0, state="normal")     # réactive
```

À retenir :

- `add_checkbutton` / `add_radiobutton` lient directement une variable Tk.
- `entryconfig(index, state=...)` pour (dé)bloquer un item à la volée.
- Indice `index` commencer à 0 (ou utiliser le label exact).

---

3) Menu contextuel (clic droit)

```
class MenuContextuel:
    def __init__(self, parent):
        self.menu = tk.Menu(parent, tearoff=0)
        self.menu.add_command(label="Copier", command=self.copier)
        self.menu.add_command(label="Coller", command=self.coller)

    def afficher(self, event):
        try:
            self.menu.tk_popup(event.x_root, event.y_root)
        finally:
            self.menu.grab_release()

    def copier(self): ...
    def coller(self): ...

# Usage
ctx = MenuContextuel(fenetre)
fenetre.bind("<Button-3>", ctx.afficher)  # clic droit Windows/Linux
```

Plateformes :

- Windows/Linux : clic droit = `<Button-3>`
- macOS (certaines configs) : clic droit peut être `<Button-2>`.
    - Astuce : binder les deux pour la compatibilité :

```
widget.bind("<Button-2>", ctx.afficher)
widget.bind("<Button-3>", ctx.afficher)
```

---

4) Événements souris courants

Pattern	Signification
`<Button-1>`	Clic gauche (pression)
`<ButtonRelease-1>`	Relâchement du clic gauche
`<Double-Button-1>`	Double-clic gauche
`<B1-Motion>`	Glisser avec bouton gauche enfoncé
`<Enter>` / `<Leave>`	La souris entre / sort du widget
`<Motion>`	Mouvement de la souris dans le widget
`<MouseWheel>`	Molette (Windows/Linux)
`<Button-4>` / `<Button-5>`	Molette (certains systèmes X11)

Exemples rapides :

```
def on_click(e): print("Clic en:", e.x, e.y)
def on_drag(e): print("Drag:", e.x, e.y)

zone.bind("<Button-1>", on_click)
zone.bind("<B1-Motion>", on_drag)
```

---

5) Molette : différences plateformes

Windows / Linux (souvent)

```
def on_wheel(e):
    # e.delta est ±120 * n
    if e.delta > 0:
        print("Scroll up")
    else:
        print("Scroll down")

fenetre.bind_all("<MouseWheel>", on_wheel)
```

X11 / compatibilité large

```
def on_wheel_up(e): print("Scroll up")
def on_wheel_down(e): print("Scroll down")
fenetre.bind_all("<Button-4>", on_wheel_up)
fenetre.bind_all("<Button-5>", on_wheel_down)
```

macOS (selon configuration Tk)

- Parfois `<MouseWheel>` fonctionne également, mais `e.delta` peut être différent (±1, ±120, etc.). Tester et adapter.

---

6) Coordonnées & infos utiles de l'event

Dans les callbacks (ex: `def on_click(e):`), l'objet `event` exposer :

- `e.x`, `e.y` : coords relatives au widget
- `e.x_root`, `e.y_root` : coords écran (utiles pour `tk_popup`)
- `e.widget` : le widget source
- `e.state` : état des modificateurs (Ctrl, Shift…)

Exemple :

```
def on_double_click(e):
    print("Widget:", e.widget, "/ xy:", e.x, e.y, "/ root:", e.x_root, e.y_root)

liste.bind("<Double-Button-1>", on_double_click)
```

---

7) Bind sur un seul widget vs global

- `widget.bind("<Button-1>", cb)` → le callback ne s'applique **qu'à ce widget**.
- `fenetre.bind_all("<Button-1>", cb)` → écoute **partout** (toute l'app).

Bonnes pratiques :

- Préférer `bind` pour des comportements locaux et précis.
- Garder `bind_all` pour des raccourcis globaux (ex: molette globale, F1, Ctrl+Q).

🩹 8) Exemple compact : zone éditable + menu contextuel

```
def copier_selection(entry):
    try:
        selection = entry.selection_get()
        entry.clipboard_clear()
        entry.clipboard_append(selection)
    except tk.TclError:
        pass

def coller_clipboard(entry):
    try:
        entry.insert(tk.INSERT, entry.clipboard_get())
    except tk.TclError:
        pass
```

```
def build_context_menu(entry):
    m = tk.Menu(entry, tearoff=0)
    m.add_command(label="Copier", command=lambda: copier_selection(entry))
    m.add_command(label="Coller", command=lambda: coller_clipboard(entry))

    def popup(e):
        try:
            m.tk_popup(e.x_root, e.y_root)
        finally:
            m.grab_release()

    entry.bind("<Button-3>", popup)      # Windows/Linux
    entry.bind("<Button-2>", popup)      # macOS (compat)
    return m

entree = tk.Entry(fenetre)
entree.pack(padx=10, pady=10, fill="x")
build_context_menu(entree)
```

---

9) Checklist rapide (menus & souris)

- ☐ `tearoff=0` sur tous les menus
- ☐ `postcommand=` pour (dés)activer des items juste avant ouverture
- ☐ Menus spécialisés : `add_checkbutton`, `add_radiobutton` (+ variables Tk)
- ☐ Clic droit multiplateforme (`<Button-3>` + `<Button-2>`)
- ☐ Molette compatible (`<MouseWheel>` + `<Button-4/5>`)
- ☐ Éviter l'abus de `bind_all` (réserver aux vrais raccourcis globaux)
- ☐ Utiliser `x_root/y_root` pour positionner un menu contextuel

Une Chimay rouge pour tout le monde ! 🍺 😄