Chapitre 4

Pierre, papier, ciseaux

(version graphique)

4.1. Thèmes abordés dans ce chapitre

- Le module tkinter : Label, Button
- Fenêtre
- Événements
- Réceptionnaire d'événements

4.2. Règles du jeu

Voir § 3.2.

Exemple de partie

Nous allons dans ce chapitre programmer une version graphique du jeu. Cela ressemblera à ceci :

4.3. Programmes pilotés par des événements

Ce paragraphe est largement inspiré du livre de Swinnen « Apprendre à programmer avec Python », chapitre 8.

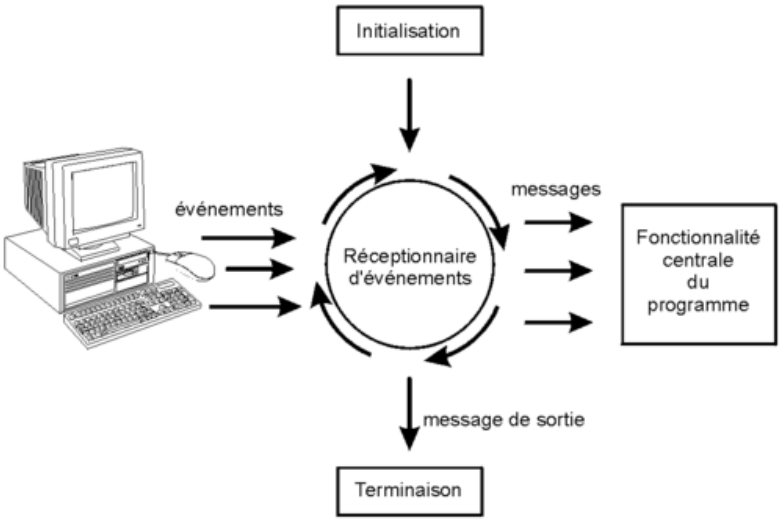
Tous les programmes d'ordinateur comportent *grosso modo* trois phases principales : une phase d'initialisation, laquelle contient les instructions qui préparent le travail à effectuer (appel des modules externes nécessaires, ouverture de fichiers, connexion à un serveur de bases de données ou à Internet, etc.), une phase centrale où l'on trouve la véritable fonctionnalité du programme (c'est-à-dire tout ce qu'il est censé faire : afficher des données à l'écran, effectuer des calculs, écrire dans un fichier, etc.), et enfin une phase de terminaison qui sert à stopper « proprement » les opérations (par exemple fermer les fichiers restés ouverts, fermer les fenêtres, etc.).

Dans un programme « en mode texte », ces trois phases sont simplement organisées suivant un schéma linéaire comme dans l'illustration ci-contre. En conséquence, ces programmes se caractérisent par une interactivité très limitée avec l'utilisateur. Celui-ci ne dispose pratiquement d'aucune liberté : il lui est demandé de temps à autre d'entrer des données au clavier, mais toujours dans un ordre prédéterminé correspondant à la séquence d'instructions du programme.

Dans le cas d'un programme qui utilise une interface graphique, par contre, l'organisation interne est différente. On dit que le programme est piloté par les événements. Après sa phase d'initialisation, un programme de ce type se met en quelque sorte « en attente », et passe la main à un autre logiciel, lequel est plus ou moins intimement intégré au système d'exploitation de l'ordinateur et « tourne » en permanence.

Ce réceptionnaire d'événements, comme on l'appelle, scrute sans cesse tous les périphériques (clavier, souris, horloge, modem, etc.) et réagit immédiatement lorsqu'un événement y est détecté. Lorsqu'il détecte un événement, le réceptionnaire envoie un message spécifique au programme, lequel doit être conçu pour réagir en conséquence.

La phase d'initialisation d'un programme utilisant une interface graphique comporte un ensemble d'instructions qui mettent en place les divers composants interactifs de cette interface (fenêtres, boutons, labels, etc.). D'autres instructions définissent les messages d'événements qui devront être pris en charge : on peut en effet décider que le programme ne réagira qu'à certains événements en ignorant tous les autres.



Alors que dans un programme « en mode texte », la phase centrale est constituée d'une suite d'instructions qui décrivent à l'avance l'ordre dans lequel la machine devra exécuter ses différentes tâches, on ne trouve dans la phase centrale d'un programme avec interface graphique qu'un ensemble de fonctions indépendantes. Chacune de ces fonctions est appelée spécifiquement lorsqu'un événement particulier est détecté par le système d'exploitation : elle effectue alors le travail que l'on

attend du programme en réponse à cet événement, et rien d'autre.

Il est important de bien comprendre ici que pendant tout ce temps, le réceptionnaire continue à « tourner » et à guetter l'apparition d'autres événements éventuels. S'il se produit d'autres événements, il peut donc arriver qu'une deuxième fonction (ou une troisième, une quatrième, ...) soit activée et commence à effectuer son travail « en parallèle » avec la première qui n'a pas encore terminé le sien. Les systèmes d'exploitation et les langages modernes permettent en effet ce parallélisme (que l'on appelle aussi multitâche).

4.4. Code du programme

ppc-graphique.py

```
# jeu pierre, papier, ciseaux
# l'ordinateur joue au hasard

from random import randint
from tkinter import *

def augmenter_scores(mon_coup,ton_coup):
    global mon_score, ton_score
    if mon_coup == 1 and ton_coup == 2:
        ton_score += 1
    elif mon_coup == 2 and ton_coup == 1:
        mon_score += 1
    elif mon_coup == 1 and ton_coup == 3:
        mon_score += 1
    elif mon_coup == 3 and ton_coup == 1:
        ton_score += 1
    elif mon_coup == 3 and ton_coup == 2:
        mon_score += 1
    elif mon_coup == 2 and ton_coup == 3:
        ton_score += 1

def jouer(ton_coup):
    global mon_score, ton_score, score1, score2
    mon_coup = randint(1,3)
    if mon_coup==1:
        lab3.configure(image=pierre)
    elif mon_coup==2:
        lab3.configure(image=papier)
    else:
        lab3.configure(image=ciseaux)
    augmenter_scores(mon_coup,ton_coup)
    score1.configure(text=str(ton_score))
    score2.configure(text=str(mon_score))

def jouer_pierre():
    jouer(1)
    lab1.configure(image=pierre)

def jouer_papier():
    jouer(2)
    lab1.configure(image=papier)

def jouer_ciseaux():
    jouer(3)
    lab1.configure(image=ciseaux)

def reinit():
    global mon_score, ton_score, score1, score2, lab1, lab3
    ton_score = 0
    mon_score = 0
    score1.configure(text=str(ton_score))
    score2.configure(text=str(mon_score))
    lab1.configure(image=rien)
    lab3.configure(image=rien)


# variables globales
ton_score = 0
mon_score = 0
```

```
# fenetre graphique
fenetre = Tk()
fenetre.title("Pierre, papier, ciseaux")

# images
rien = PhotoImage(file ='rien.gif')
versus = PhotoImage(file ='versus.gif')
pierre = PhotoImage(file ='pierre.gif')
papier = PhotoImage(file ='papier.gif')
ciseaux = PhotoImage(file ='ciseaux.gif')

# labels
texte1 = Label(fenetre, text="Humain :", font=("Helvetica", 16))
texte1.grid(row=0, column=0)

texte2 = Label(fenetre, text="Machine :", font=("Helvetica", 16))
texte2.grid(row=0, column=2)

texte3 = Label(fenetre, text="Pour jouer, cliquez sur une des icones ci-dessous.")
texte3.grid(row=3, columnspan=3, pady=5)

score1 = Label(fenetre, text="0", font=("Helvetica", 16))
score1.grid(row=1, column=0)

score2 = Label(fenetre, text="0", font=("Helvetica", 16))
score2.grid(row=1, column=2)

lab1 = Label(fenetre, image=rien)
lab1.grid(row=2, column=0)

lab2 = Label(fenetre, image=versus)
lab2.grid(row=2, column=1)

lab3 = Label(fenetre, image=rien)
lab3.grid(row=2, column=2)

# boutons
bouton1 = Button(fenetre,command=jouer_pierre)
bouton1.configure(image=pierre)
bouton1.grid(row=4, column=0)

bouton2 = Button(fenetre,command=jouer_papier)
bouton2.configure(image=papier)
bouton2.grid(row=4, column=1)

bouton3 = Button(fenetre,command=jouer_ciseaux)
bouton3.configure(image=ciseaux)
bouton3.grid(row=4, column=2)

bouton4 = Button(fenetre,text='Recommencer',command=reinit)
bouton4.grid(row=5, column=0, pady=10, sticky=E)

bouton5 = Button(fenetre,text='Quitter',command=fenetre.destroy)
bouton5.grid(row=5, column=2, pady=10, sticky=W)

# demarrage :
fenetre.mainloop()
```

4.5. Analyse du programme

Reprenons ce programme pour l'expliquer en détails.

```
# jeu pierre, papier, ciseaux
# l'ordinateur joue au hasard

from random import randint
from tkinter import *
```

C'est le module externe `tkinter` qui va nous permettre de faire des graphiques.
Il faudra gérer des événements *(events)* provenant des périphériques (le clavier ou la souris). Par

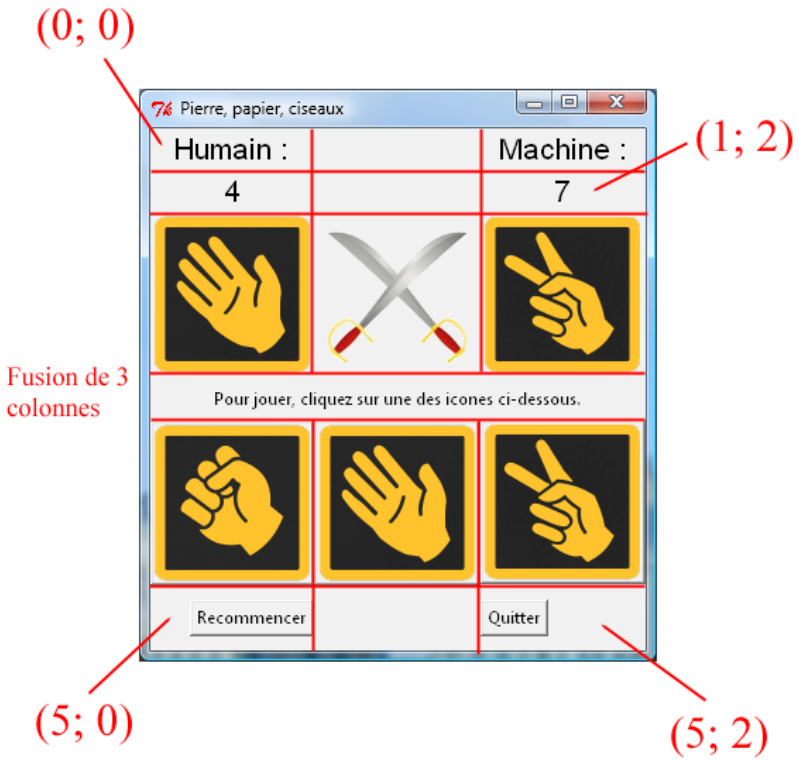
exemple, un événement pourrait être que la touche « espace » a été pressée, ou que la souris a été déplacée. Dans notre jeu, le seul événement qui nous intéressera sera le clic du bouton gauche de la souris. C'est en effet via la souris que le joueur va choisir le coup qu'il va jouer.

4.5.1. Les fenêtres

Il faudra ouvrir une fenêtre graphique dans laquelle on va placer divers éléments (des boutons, des textes, des étiquettes, etc.).

```
# fenetre graphique
fenetre = Tk()
fenetre.title("Pierre, papier, ciseaux")
```

Pour notre jeu, la fenêtre a été divisée virtuellement en plusieurs zones correspondant aux cases d'une grille. Chaque case est repérée par sa ligne et sa colonne. **Attention !** La numérotation commence à 0. Ainsi, la case en haut à gauche a pour coordonnées (0; 0). La case placée sur la deuxième ligne et la troisième colonne a pour coordonnées (1; 2).



On voit que ces zones n'ont pas la même taille. Elles s'adaptent automatiquement aux dimensions des objets qu'elles contiennent.

4.5.2. Importation des images

Nous aurons besoin de cinq images : les symboles pierre, papier, ciseaux, une case noire pour le début du jeu (« rien ») et les deux épées croisées pour faire joli. Toutes ces images, au format gif, doivent se trouver dans le même répertoire que le programme.

C'est la fonction `PhotoImage` du module `tkinter` qui importe les images.

```
# images
rien = PhotoImage(file ='rien.gif')
versus = PhotoImage(file ='versus.gif')
pierre = PhotoImage(file ='pierre.gif')
papier = PhotoImage(file ='papier.gif')
ciseaux = PhotoImage(file ='ciseaux.gif')
```

4.5.3. Les labels

Les labels peuvent être du texte ou des images. Sur la première ligne de la grille, on a écrit les noms des joueurs (Humain et Machine). On a indiqué aussi la police utilisée (`Helvetica`) et la taille (`16`).

```
# labels
texte1 = Label(fenetre, text="Humain :", font=("Helvetica", 16))
texte1.grid(row=0, column=0)

texte2 = Label(fenetre, text="Machine :", font=("Helvetica", 16))
texte2.grid(row=0, column=2)
```

La quatrième ligne (`row=3`) est spéciale : elle est composée d'une seule case, qui s'étend sur la largeur de 3 colonnes (`columnspan=3`). Pour que le texte ne soit pas trop serré, on a mis une marge de 5 pixels en dessus et en dessous (`pady=5`).

```
texte3 = Label(fenetre, text="Pour jouer, cliquez sur une des icones ci-dessous.")
texte3.grid(row=3, columnspan=3, pady=5)
```

Le contenu des labels peut varier durant l'exécution du programme. Par exemple, les scores vont évidemment changer en cours de partie. Au début, ils sont initialisés à 0 (`text="0"`).

Les scores des deux joueurs sont placés sur la deuxième ligne (`row=1`) et sur la première (`column=0`) et troisième colonne (`column=2`). On a indiqué aussi la police utilisée (`Helvetica`) et la taille (`16`).

```
score1 = Label(fenetre, text="0", font=("Helvetica", 16))
score1.grid(row=1, column=0)

score2 = Label(fenetre, text="0", font=("Helvetica", 16))
score2.grid(row=1, column=2)
```

La troisième ligne (`row=2`) est composée de 3 labels : le coup joué par l'humain (rien au départ), les deux épées et le coup joué par la machine (rien au départ).

```
lab1 = Label(fenetre, image=rien)
lab1.grid(row=2, column=0)

lab2 = Label(fenetre, image=versus)
lab2.grid(row=2, column=1)

lab3 = Label(fenetre, image=rien)
lab3.grid(row=2, column=2)
```

4.5.4. Les boutons

Les boutons peuvent être du texte ou des images. Dans notre fenêtre, il y a 5 boutons : les symboles pierre, papier et ciseaux, ainsi que les boutons « Recommencer » et « Quitter ».

Commençons par les boutons images.

```
# boutons
bouton1 = Button(fenetre,command=jouer_pierre)
bouton1.configure(image=pierre)
bouton1.grid(row=4, column=0)

bouton2 = Button(fenetre,command=jouer_papier)
bouton2.configure(image=papier)
bouton2.grid(row=4, column=1)

bouton3 = Button(fenetre,command=jouer_ciseaux)
bouton3.configure(image=ciseaux)
bouton3.grid(row=4, column=2)
```

Pour chacun de ces boutons, la première ligne crée le bouton en indiquant dans quelle fenêtre il se trouve et la commande qui sera appelée quand on cliquera dessus. Cette commande est une procédure sans paramètres. La deuxième ligne indique quelle image sera associée au bouton. Quant à la troisième ligne, elle indique dans quelle case de la grille se trouve le bouton.

Passons maintenant aux boutons contenant du texte.

```
bouton4 = Button(fenetre,text='Recommencer',command=reinit)
bouton4.grid(row=5, column=0, pady=10, sticky=E)

bouton5 = Button(fenetre,text='Quitter',command=fenetre.destroy)
bouton5.grid(row=5, column=2, pady=10, sticky=W)
```

La création d'un bouton est presque identique à celle d'un bouton image, sauf que l'on précise le texte qui figurera dans le bouton. La deuxième ligne donne l'emplacement du bouton. On a précisé ici que le bouton « Recommencer » est calé à droite (`sticky=E`, avec E comme East) et le bouton « Quitter » à gauche (`sticky=W`, avec W comme West). Il existe aussi les lettres N pour North, donc calé en haut et S pour South, donc calé en bas.

4.5.5. Les procédures

On va retrouver des procédures écrites pour la version non graphique, parfois légèrement modifiées, ainsi que les procédures appelées par les boutons.

```
def augmenter_scores(mon_coup,ton_coup):
    global mon_score, ton_score
    if mon_coup == 1 and ton_coup == 2:
        ton_score += 1
    elif mon_coup == 2 and ton_coup == 1:
        mon_score += 1
    elif mon_coup == 1 and ton_coup == 3:
        mon_score += 1
    elif mon_coup == 3 and ton_coup == 1:
        ton_score += 1
    elif mon_coup == 3 and ton_coup == 2:
        mon_score += 1
    elif mon_coup == 2 and ton_coup == 3:
        ton_score += 1
```

Cette procédure est la même que dans la version non graphique.

```
def jouer(ton_coup):
    global mon_score, ton_score, score1, score2
    mon_coup = randint(1,3)
    if mon_coup==1:
        lab3.configure(image=pierre)
    elif mon_coup==2:
        lab3.configure(image=papier)
    else:
        lab3.configure(image=ciseaux)
    augmenter_scores(mon_coup,ton_coup)
    score1.configure(text=str(ton_score))
    score2.configure(text=str(mon_score))
```

Cette procédure `jouer` a été modifiée pour s'adapter au graphisme. Quand le joueur presse sur un des trois boutons, l'ordinateur tire au hasard un coup, puis modifie l'image du label correspondant au coup choisi (`lab3.configure(image=pierre)`). Les nouveaux scores sont ensuite calculés, puis les labels `score1` et `score2` sont mis à jour, grâce à la procédure `configure`.

```
def jouer_pierre():
    jouer(1)
    lab1.configure(image=pierre)

def jouer_papier():
    jouer(2)
    lab1.configure(image=papier)
```

```
def jouer_ciseaux():
    jouer(3)
    lab1.configure(image=ciseaux)
```

Ces trois procédures sont appelées par les boutons. Elles mettent à jour le label indiquant le coup choisi par le joueur.

```
def reinit():
    global mon_score,ton_score,score1,score2,lab1,lab3
    ton_score = 0
    mon_score = 0
    score1.configure(text=str(ton_score))
    score2.configure(text=str(mon_score))
    lab1.configure(image=rien)
    lab3.configure(image=rien)
```

Cette dernière procédure est appelée par le bouton « Recommencer ». Elle réinitialise les variables globales, ainsi que les labels scores et les labels montrant les coups joués.

4.5.6. Gestion des événements

Le réceptionnaire d'événements est lancé par l'instruction :

```
fenetre.mainloop()
```

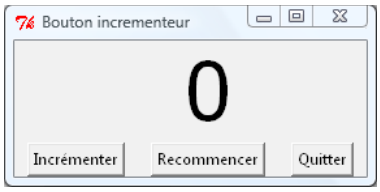
Dans notre programme de jeu, la phase de terminaison consistera simplement à fermer la fenêtre. C'est le bouton « Quitter » qui appellera la fonction `destroy` :

```
bouton5 = Button(fenetre,text='Quitter',command=fenetre.destroy)
```

Il est à noter que ce bouton n'était pas indispensable. En effet, cliquer le bouton rouge habituel en haut à droite de la fenêtre aurait eu le même effet.

Exercice 4.1

Reproduisez la fenêtre ci-dessous :



Le bouton « Incrémenter » augmentera de 1 le compteur. Le bouton « Recommencer » remettra le compteur à 0 et le bouton « Quitter » fermera la fenêtre.

Exercice 4.2

Reproduisez la fenêtre ci-dessous :

Les effets des boutons seront les mêmes que dans l'exercice 4.1.

Exercice 4.3

Reproduisez la frise ci-dessous, en utilisant une boucle (`while` ou `for`).

Exercice 4.4

Reproduisez une frise comme ci-dessous, en tirant les cartes au hasard. Chaque colonne représente une manche de pierre-papier-ciseaux. Comme pour l'exercice 4.3, utilisez une boucle.

Exercice 4.5

Modifiez le code du § 4.4.

Ajoutez une deuxième fenêtre pour que l'on puisse voir l'historique des 20 derniers coups joués dans la partie.

Attention ! Pour ouvrir une nouvelle fenêtre, il faudra utiliser l'instruction :

```
historique = Toplevel()
historique.title("Historique")
```

En effet, on ne peut pas utiliser l'instruction `Tk()` deux fois dans le même programme. Cette fenêtre sera une « fille » de la fenêtre principale, et elle disparaîtra en même temps qu'elle.

Cette deuxième fenêtre ressemblera à celle de l'exercice 4.4, mais en écrivant quelle est la ligne du joueur et celle de l'ordinateur. Quand 20 coups auront été joués, recommencez à dessiner les symboles depuis la première colonne, sans rien effacer.

Exercice 4.6

Modifiez le code du § 4.4.

Ajoutez un quatrième symbole : le puits. La pierre et les ciseaux tombent dans le puits (donc le puits gagne contre eux). Par contre, le papier recouvre le puits (le puits perd donc contre le papier).

Attention ! Les probabilités de gain changent, et la stratégie aussi !

Exercice 4.7

Let's Make A Deal ! est un show télévisé américain qui a commencé le 30 décembre 1963. À la fin du jeu, l'animateur, Monty Hall, vous offrait la possibilité de gagner ce qui se trouvait derrière une porte. Il y avait trois portes : derrière l'une d'entre elles se trouvait un prix magnifique (par exemple un voyage) et derrière les deux autres un prix moins intéressant (par exemple une chèvre ou une barre de chocolat). Vous choisissiez alors une porte.

Pour ménager le suspense, Monty Hall, avant de révéler ce qu'il y avait derrière la porte que vous aviez choisie, ouvrait une des deux autres portes (derrière laquelle ne se trouvait **jamais** le plus beau cadeau).

Il vous posait enfin la dernière question : « Conservez-vous votre premier choix, ou bien choisissez-vous l'autre porte encore fermée ? »

- Programmez ce jeu avec une interface graphique. Vous pourrez vous inspirer de cette page web : www.apprendre-en-ligne.net/random/monty/
- Quelle est la meilleure stratégie : changer de porte ou garder la porte choisie en premier ?

4.6. Ce que vous avez appris dans ce chapitre

- C'est dans le module `tkinter` que l'on trouve les procédure permettant de créer une interface graphique.
- Quand il y a une interface graphique, le programme est piloté par des événements (§ 4.3). C'est très différent des programmes « en mode texte » que l'on a vu dans les trois premiers chapitres.
- Dans le paragraphe 4.5 et les exercices, vous avez vu comment :
  - ouvrir et utiliser des fenêtres graphiques,
  - importer des images au format gif,
  - utiliser les labels,
  - définir des boutons,
  - gérer des événements.