



Chapitre 13

Damiers

13.1. Nouveau thème abordé dans ce chapitre

- glisser-déposer

Il n'y a qu'une nouvelle notion dans ce chapitre, mais comme beaucoup de jeux se jouent sur un damier (le terme générique est plutôt *tablier*), il est bon de consacrer un chapitre à ce sujet. Cela permettra de renforcer les notions vues aux chapitres précédents.

13.2. Damiers avec des lignes



lignes.py

```
# Dessin d'un damier en traçant des lignes
from tkinter import *

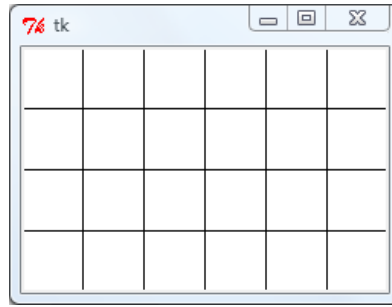
def lignes_verticales(c,nc,nl):
    x=0
    while x < nc*c:
        canvas.create_line(x,0,x,nl*c,width=1,fill='black')
        x += c

def lignes_horizontales(c,nc,nl):
    y=0
    while y < nl*c:
        canvas.create_line(0,y,nc*c,y,width=1,fill='black')
        y += c

def damier(c, nl, nc):
    lignes_verticales(c,nc,nl)
    lignes_horizontales(c,nc,nl)

c = 40          # côté d'un carré
nl = 4          # nombre de lignes du damier
nc = 6          # nombre de colonnes
fen = Tk()
canvas = Canvas(fen, width=c*nc-3, height=c*nl-3, bg='white')
canvas.pack(side=TOP)
damier(c, nl, nc)
fen.mainloop()
```

Voici la grille obtenue :



13.3. Damiers avec des carrés



carres.py

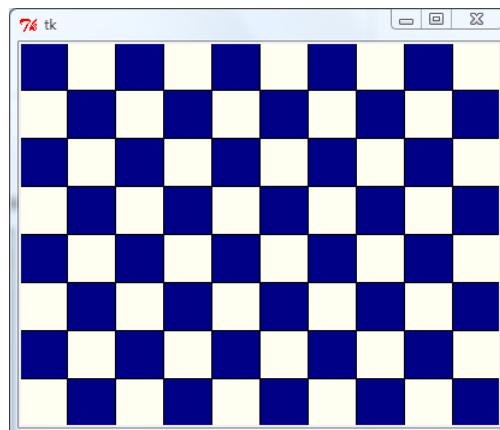
```
# Dessin d'un damier avec des carrés
from tkinter import *

def ligne_de_carres(x, y, c, n):
    # dessiner une ligne de n carrés
    i = 0
    while i < n:
        can.create_rectangle(x, y, x+c, y+c, fill='navy')
        i += 1
        x += c*2                # espacer les carrés

def damier(c, nl, nc):
    # dessiner nl lignes de nc carrés de taille c avec décalage alterné
    y = 0
    while y < nl:
        if y % 2 == 0:          # une ligne sur deux, on
            x = 0               # commencera la ligne de
        else:                  # carrés avec un décalage
            x = 1               # de la taille d'un carré
        ligne_de_carres(x*c, y*c, c, nc)
        y += 1

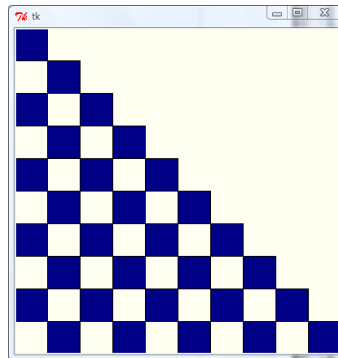
c = 40                        # côté d'un carré
nl = 8                        # nombre de lignes du damier
nc = 10                       # nombre de colonnes
fen = Tk()
can = Canvas(fen, width=c*nc-3, height=c*nl-3, bg='ivory')
can.pack(side=TOP)
damier(c, nl, nc)
fen.mainloop()
```

Voici le damier obtenu :

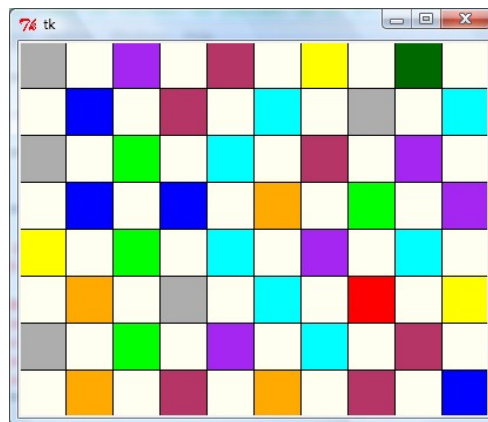


**Exercice 13.1**

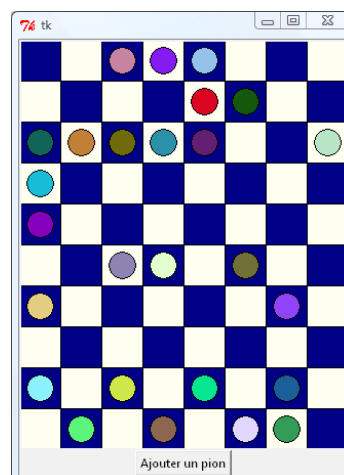
Modifiez le programme du § 13.3 pour obtenir le damier ci-dessous :

**Exercice 13.2**

Modifiez le programme du § 13.3 pour que les cases de couleur « navy » du damier d'origine ait une couleur aléatoire choisie parmi une liste de 10 (à vous de choisir lesquelles).

**Exercice 13.3**

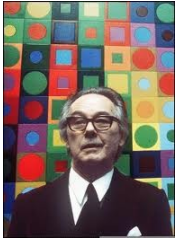
Modifiez le programme du § 13.3. Ajoutez un bouton « Ajouter un pion ». Quand vous cliquerez sur ce bouton, un pion d'une couleur aléatoire apparaîtra sur une case choisie au hasard.



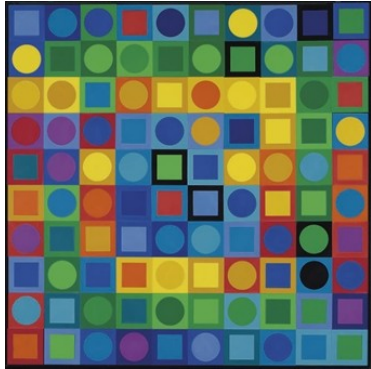


Exercice 13.4

Victor **Vasarely** (1908-1997) est un plasticien hongrois, naturalisé français en 1961, reconnu comme étant le père de l'*art optique* ou *Op art*. Voici une de ces œuvres (à gauche) :



Victor **Vasarely**
(1906-1997)



Vasarely Planetary Folklore Participations No. 1 (1969)



Faux généré par ordinateur

Créez un programme qui générera avec des couleurs aléatoires un tableau ressemblant à « un Vasarely ». Les petits ronds et les petits carrés seront eux aussi placés aléatoirement.



Exercice 13.5

Écrivez un programme qui dessine un quadrillage composé de 16 rectangles sur 30, chaque rectangle étant coloré d'une des 479 couleurs définies dans le code du § 11.2, avec le nom de la couleur écrit en noir à l'intérieur du rectangle correspondant. Un rectangle sera trois fois plus long que large.

13.4. Damiers avec des hexagones



hexagones.py

```
# Dessin d'un damier hexagonal

from tkinter import *
from random import choice

def couleur():
    coul = "#"
    liste = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f']
    for i in range(6):
        coul += choice(liste)
    return coul

def hexagone(origine, c):
    un_seg = c/4
    deux_seg = c/2
    x, y = origine[0], origine[1]
    # hexagone
    can.create_polygon(x, y + deux_seg, x + un_seg, y,
        x + un_seg + deux_seg, y, x + c, y + deux_seg,
        x + un_seg + deux_seg, y + c, x + un_seg, y + c,
        x, y + deux_seg, outline='black', width=1, fill = couleur())

def ligne_d_hexagones(x, y, c, n):
    # dessiner une ligne d'hexagones
    i = 0
    un_seg = c/4
    while i < n:
        hexagone((x+2, y+2), c) # +2 : décalage pour que tout soit bien dessiné
        i += 1
```

```

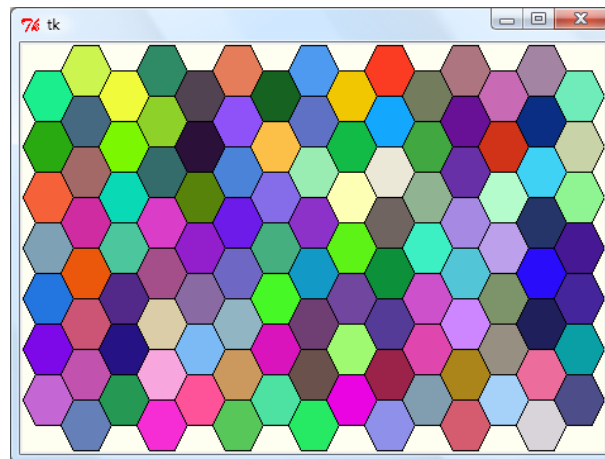
        x += 6*un_seg

def damier(c, nl, nc):
    # dessiner nl lignes d'hexagones de taille c avec décalage alterné
    y = 0
    no_ligne = 0
    while no_ligne < nl:
        if no_ligne%2 == 1:          # une ligne sur deux, on
            x = 0                    # commencera la ligne
        else:                        # avec un décalage
            x = 0.75
        if nc%2 == 1:
            if no_ligne%2 == 0:
                ligne_d_hexagones(x*c, y*c, c, nc//2)
            else:
                ligne_d_hexagones(x*c, y*c, c, nc//2+1)
        else:
            ligne_d_hexagones(x*c, y*c, c, nc//2)
        y += 0.5
        no_ligne += 1

c = 40          # taille des carrés circonscrit à l'hexagone
nl = 15         # nombre de lignes
nc = 15         # nombre de colonnes
fen = Tk()
can = Canvas(fen, width=c+(nc-1)*0.75*c, height=c+(nl-1)*c/2+1, bg='ivory')
can.pack(side=TOP)
damier(c,nl,nc)
fen.mainloop()

```

Voici le résultat obtenu :



13.5. Analyse du programme

```

# Dessin d'un damier hexagonal

from tkinter import *
from random import choice

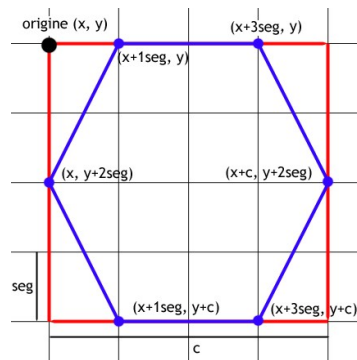
def couleur():
    coul = "#"
    liste = ['0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f']
    for i in range(6):
        coul += choice(liste)
    return coul

```

Vous remarquerez que nous avons utilisé le format RGB pour définir une couleur aléatoire. La couleur aura pour format un # suivi de 6 symboles choisis dans `liste`.

```
def hexagone(origine, c):
    seg = c/4
    x, y = origine[0], origine[1]
    # hexagone
    can.create_polygon(x, y+2*seg, x+seg, y, x+3*seg, y, x+c, y+2*seg,
    x+3*seg, y+c, x+seg, y+c, x, y+2*seg,
    outline = 'black', width = 1, fill = couleur())
```

Le schéma ci-dessous explique comment est construit un hexagone :

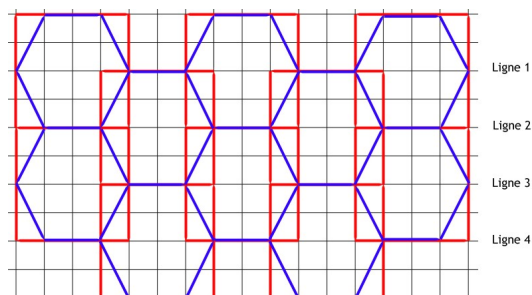


Notez au passage que ce n'est pas un hexagone régulier, puisque ses côtés n'ont pas tous la même longueur. L'avantage de cet hexagone est qu'il est simple à dessiner et à calculer.

```
def ligne_d_hexagones(x, y, c, n):
    # dessiner une ligne d'hexagones
    i = 0
    seg = c/4
    while i < n:
        hexagone((x+2, y+2), c) # +2 : décalage pour que tout soit bien dessiné
        i += 1
        x += 6*seg

def damier(c, nl, nc):
    # dessiner nl lignes d'hexagones de taille c avec décalage alterné
    y = 0
    no_ligne = 0
    while no_ligne < nl:
        if no_ligne%2 == 1:
            # une ligne sur deux, on
            # commencera la ligne
            # avec un décalage
            x = 0.75
        if nc%2 == 1:
            if no_ligne%2 == 0:
                ligne_d_hexagones(x*c, y*c, c, nc//2)
            else:
                ligne_d_hexagones(x*c, y*c, c, nc//2+1)
        else:
            ligne_d_hexagones(x*c, y*c, c, nc//2)
        y += 0.5
        no_ligne += 1
```

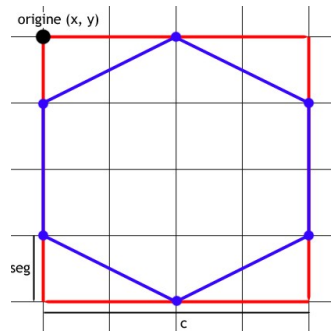
Le schéma ci-dessous explique comment sont alignés les hexagones pour former un damier :



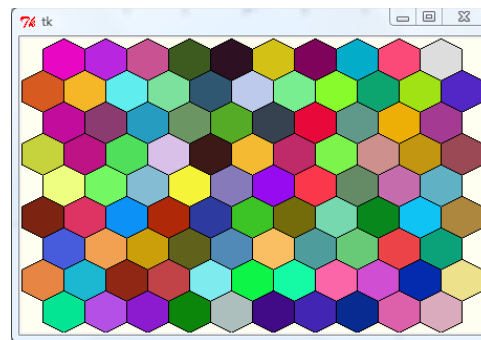


Exercice 13.6

Modifiez le programme du § 13.4. Construisez les hexagones comme expliqué sur le schéma ci-dessous.



Utilisez-les pour dessiner un damier à trame hexagonale ressemblant à celui ci-dessous.



13.6. Glisser-déposer

Le **glisser-déposer** (traduction recommandée de l'anglais *drag and drop*) consiste à utiliser une souris pour déplacer d'un endroit à un autre un élément graphique présent sur l'écran d'un ordinateur. Pour ce faire, il faut cliquer sur cet élément graphique, maintenir le bouton droit de la souris enfoncé, et ne le relâcher que lorsque le pointeur a atteint l'endroit souhaité.

Le programme ci-dessous montre comment glisser-déposer un pion.



dragdrop.py

```
# exemple de glisser-déposer (drag and drop)

from tkinter import *

def glisser(evt):
    global x, y
    if x == -1:
        if x_pion-50 < evt.x < x_pion+50 and y_pion-100 < evt.y < y_pion+100 :
            x, y = evt.x, evt.y
    else :
        delta_x, delta_y = evt.x - x, evt.y - y
        fond.coords(pion, x_pion + delta_x, y_pion + delta_y)

def deposer(evt):
    global x_pion, y_pion, x
    if x != -1 :
        x_pion, y_pion = x_pion + evt.x - x, y_pion + evt.y - y
        x = -1

def dessus(evt) :
    # modifie la forme du curseur
    forme = "arrow"
    if x_pion-50 < evt.x < x_pion+50 and y_pion-100 < evt.y < y_pion+100 :
```

```

        forme = "fleur"
        fond.config(cursor = forme)

fenetre=Tk()
fond=Canvas(fenetre,width=800,height=600,bg="white")

x, y = -1, -1                # coordonnées initiales de la souris
x_pion, y_pion = 400, 400    # coordonnées initiales du pion
f_pion = PhotoImage(file="pion.gif") # dimensions de l'image : 107 x 200
pion = fond.create_image(x_pion,y_pion,image=f_pion)

fond.grid()
fond.bind('<B1-Motion>',glisser)
fond.bind('<ButtonRelease-1>',deposer)
fond.bind('<Motion>',dessus)

fenetre.mainloop()

```

13.7. Analyse du programme Glisser-déposer

Avant toute chose, il faut remarquer que l'image du pion que l'on utilise est inscrite dans un rectangle de 107 pixels de large et de 200 pixels de haut. On va repérer le pion avec son centre (initialement $x_pion = y_pion = 400$). La souris sera sur le pion quand les coordonnées de la souris seront dans le rectangle englobant le pion.

La première fois que l'événement `<B1-Motion>` survient (c'est-à-dire quand la souris se déplace avec le bouton gauche pressé), on appelle la procédure `glisser`. On stocke dans les variables x, y les coordonnées de la souris. Celles-ci ont été initialisées à $(-1; -1)$ pour signifier qu'aucun mouvement n'est en cours.

À chaque fois que l'événement `<B1-Motion>` survient à nouveau, on calcule le vecteur de déplacement entre les coordonnées actuelles de la souris (`evt.x ; evt.y`) et le point $(x; y)$. On fait subir à l'image le même déplacement sans mettre à jour les coordonnées du centre du pion pour le moment.

Enfin, lorsqu'on relâche le bouton (`<ButtonRelease-1>`), on appelle la procédure `deposer`. Les coordonnées x_pion, y_pion sont mises à jour, et on donne à x la valeur -1 pour indiquer que le déplacement est terminé.

Formes du curseur

Pour améliorer le rendu, on transforme le curseur lors du survol du pion pour signifier au joueur que cette pièce peut être déplacée. On exécutera la procédure `dessus` sur l'événement `<Motion>`.

Le module `tkinter` contient différentes formes de curseur, dont le motif peut varier selon le système d'exploitation. Voici une liste des noms les plus intéressants : `arrow`, `circle`, `clock`, `cross`, `dotbox`, `exchange`, `fleur`, `heart`, `man`, `mouse`, `pirate`, `plus`, `shuttle`, `sizing`, `spider`, `spraycan`, `star`, `target`, `tcross`, `trek`, `watch`.



Exercice 13.7

Modifiez le programme du § 13.6 afin de pouvoir glisser-déposer trois pions au lieu d'un.



Exercice 13.8

Modifiez le programme de l'exercice 13.7 afin de pouvoir glisser-déposer un pion noir et un pion blanc sur un damier 8×8 . Ils devront toujours être déposés bien au centre d'une case. Procédez de la manière suivante :

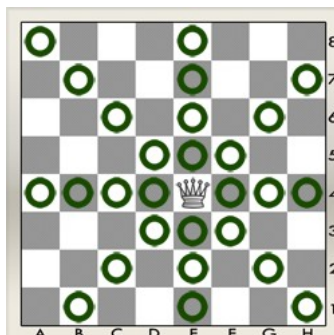
1. Combinez les programmes du § 13.3 et de l'exercice 13.7.
2. Utilisez une liste pour stocker les deux pions.
3. Centrez les pions sur les cases après le glisser-déposer.
4. Pensez au fait que les pions ne doivent pas sortir du damier.

**Exercice 13.9**

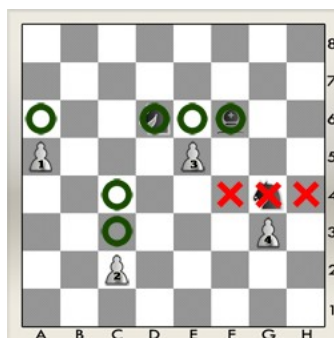
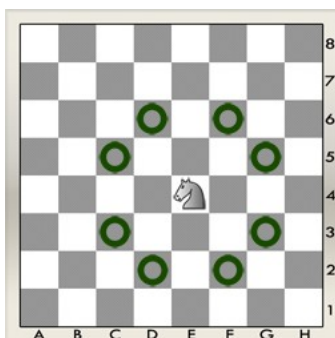
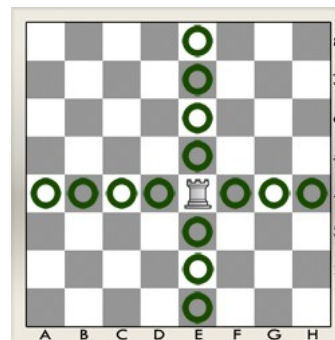
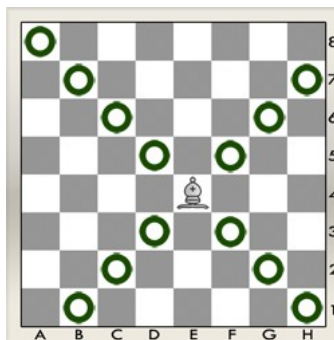
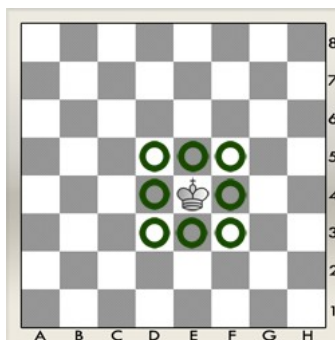
En vous inspirant de l'exercice 13.8, écrivez un programme qui permette de déplacer une dame sur un échiquier en respectant les règles du jeu (voir schéma ci-dessous).

Source des images :

<http://www.chess-mind.com/fr/echecs/regles>



Faites ensuite de même pour les autres pièces du jeu (un programme par pièce).



Comme case de départ, choisissez pour le pion une case de la seconde ligne.

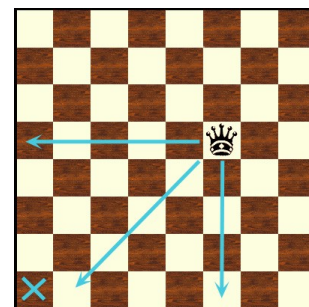
**Exercice 13.10**

On déplace une dame sur un échiquier. Comme la dame des échecs, elle peut se déplacer horizontalement, verticalement et en diagonale, mais seulement vers la gauche et vers le bas.

Deux joueurs s'affrontent. Le premier pose la dame sur l'échiquier, le second la déplace, puis chaque joueur la déplace à tour de rôle. Le vainqueur est le joueur qui réussit à atteindre la case inférieure gauche de l'échiquier.

Programmez ce jeu de sorte que l'on puisse jouer contre l'ordinateur. Il faudra d'abord trouver une stratégie avant de la programmer...

Vous déplacerez la dame avec un glisser-déposer et vérifierez que les mouvements sont valides.

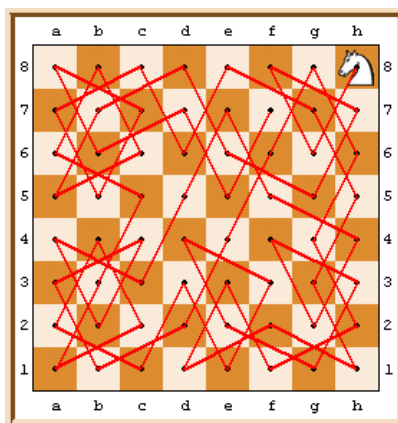




Exercice 13.11 - Le parcours du cavalier

Le *parcours du cavalier* sur un échiquier est un problème ancien : il s'agit de déterminer un trajet parcouru par un cavalier qui part d'une case quelconque de l'échiquier et qui visite toutes les cases une et une seule fois. On exige parfois le retour sur la première case, on parle alors de parcours fermé.

Voici un exemple de parcours (départ en h8, arrivée en f7) :



Modifiez le programme de l'exercice 13.9 pour qu'un joueur puisse tenter de trouver un parcours du cavalier. Vous tracerez le chemin du cavalier comme ci-dessus.



Exercice 13.12 - L'heuristique de Warnsdorff

Une *heuristique* est un algorithme qui fournit rapidement une solution (pas forcément optimale) pour un problème d'optimisation. L'heuristique proposée en 1823 par le mathématicien allemand *Warnsdorff* permet de trouver une solution au problème du cavalier d'Euler en un temps linéaire. Cependant, il est possible dans certains cas que cette heuristique ne marche pas.

Le principe de cette heuristique est le suivant :

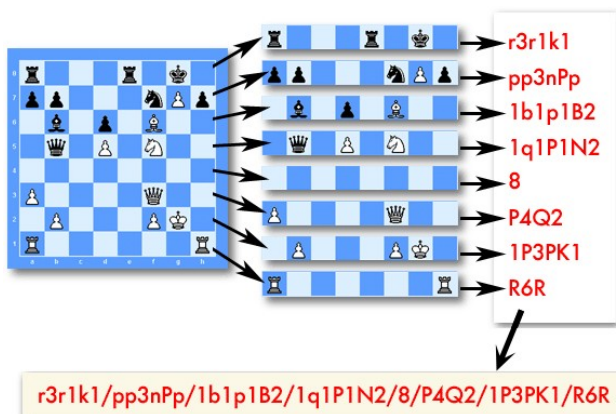
- à chaque case *c* de l'échiquier, on affecte un poids égal au nombre de cases accessibles à partir de *c* ;
- lorsque l'on parcourt l'échiquier, on choisit les cases de poids minimal.

Modifiez le programme de l'exercice 13.11 et programmez cette heuristique.



Exercice 13.13 - La notation FEN

FEN est l'abréviation de Forsyth-Edwards Notation. Ce format permet de décrire la position des pièces lors d'une partie d'échecs.



En partant de la case A8 en haut à gauche, chaque rangée de l'échiquier est décrite de gauche à droite. Elles sont séparées par le symbole /.

Une lettre indique la présence d'une pièce, un chiffre indique le nombre de cases vides jusqu'à la prochaine pièce ou la fin de la rangée (le total d'une rangée doit faire 8, en disant qu'une pièce est égale à 1). Les lettres minuscules désignent les pièces noires, les lettres majuscules les pièces blanches.



Vous trouverez des images sur le site web compagnon.

K = roi blanc	k = roi noir	Q = dame blanche	q = dame noire
R = tour blanche	r = tour noire	B = fou blanc	b = fou noir
N = cavalier blanc	n = cavalier noir	P = pion blanc	p = pion noir

Écrivez un programme qui affiche successivement à l'écran les trois positions suivantes :

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR
rnbqkbnr/pp1ppppp/8/2p5/4P3/5N2/PPPP1PPP/RNBQKB1R
2kr3r/1pp2ppp/1n2b3/p3P3/PnP2B2/1K2P3/1P4PP/R4BNR
```



Exercice 13.14

Exercice inverse de l'exercice 13.13 : écrivez un programme qui permette de sauvegarder dans un fichier une position d'une partie d'échecs avec la notation FEN.

Il faudra programmer une interface graphique agréable qui permettra à l'utilisateur de choisir sa pièce et de la placer sur la case souhaitée. Il faudra aussi penser à pouvoir corriger une erreur lors de la mise en place.



Exercice 13.15 - Échecs aléatoires Fischer

Les *échecs aléatoires Fischer* sont une variante du jeu d'échecs dans laquelle l'emplacement initial des pièces de la première et de la dernière rangée est tiré au sort, et identique pour les deux camps. Ils sont nommés d'après le joueur d'échecs américain Bobby **Fischer**.

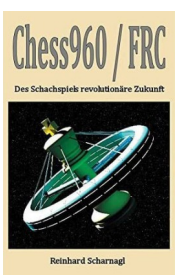
Les règles du jeu sont exactement les mêmes qu'aux échecs orthodoxes, sauf que l'emplacement initial des pièces de la première et de la dernière rangée est tiré au sort.

- Le roi doit se trouver quelque part entre les deux tours afin que le petit et le grand roque soient tous deux possibles dans toutes les positions (la position du roi et de la tour après le roque/grand-roque est la même qu'avec une position normale).
- Les fous doivent obligatoirement être placés sur des cases de couleurs opposées.
- Les pièces noires (huitième rangée) sont disposées symétriquement face aux blanches (première rangée), exactement dans le même ordre. Exemple : si du côté blanc, il y a une dame en g1, et un cavalier en c1, du côté noir, la dame se trouvera en g8 et le cavalier en c8.
- Les pions sont disposés de façon habituelle, huit pions blancs sur la deuxième rangée et huit pions noirs sur la septième rangée.

- a. Cette variante du jeu d'échecs s'appelle aussi « Chess 960 », car il y a 960 configurations initiales possibles. Écrivez un programme qui confirme ce nombre en listant toutes les configurations de départ.



Bobby Fischer
(1943-2008)



Reinhard **Scharnagl** recommande l'identification de chacune des 960 positions initiales par un numéro de 0 à 959. Voici la méthode pour trouver l'emplacement initial des pièces d'après le numéro d'identification :

- Divisez le nombre par 4 et vous obtenez un nombre « X ». Ce nombre se termine soit par .00 (désignant la première case blanche à gauche), .25 (1/4) (désignant la deuxième), .50 (2/4) (la troisième) ou par .75 (3/4) (la quatrième). Placez le fou blanc sur la case appropriée [.00 = b .25 = d .50 = f .75 = h]
- Divisez la partie entière de « X » par 4 et vous obtenez un nombre « Y ». Fonctionnez de la même façon qu'à l'étape précédente pour déterminer la case du fou noir [.00 = a .25 = c .50 = e .75 = g]
- Divisez la partie entière de « Y » par 6 et vous obtenez le nombre « Z ». Il peut se terminer

de 6 façons différentes : **.00** - **.167** - **.333** - **.500** - **.667** - **.833**. Placez la dame sur sa case en suivant toujours la même technique [.00 = première case libre depuis la gauche - .16. = deuxième case libre depuis la gauche, etc.]

- La partie entière de « Z » correspond au code RTC (Roi-Tour-Cavalier). Placez alors les tours, les cavaliers et le roi dans les cases libres selon l'ordre indiqué.

code RTC	ordre des pièces restantes
0	Cavalier-Cavalier-Tour-Roi-Tour
1	Cavalier-Tour-Cavalier-Roi-Tour
2	Cavalier-Tour-Roi-Cavalier-Tour
3	Cavalier-Tour-Roi-Tour-Cavalier
4	Tour-Cavalier-Cavalier-Roi-Tour
5	Tour-Cavalier-Roi-Cavalier-Tour
6	Tour-Cavalier-Roi-Tour-Cavalier
7	Tour-Roi-Cavalier-Cavalier-Tour
8	Tour-Roi-Cavalier-Tour-Cavalier
9	Tour-Roi-Tour-Cavalier-Cavalier

La disposition orthodoxe des pièces correspond au numéro d'identification **518**.

- Écrivez un programme qui donne les 960 configurations initiales, numérotées selon ce système.
- Écrivez un programme qui affiche au hasard une de ces configurations initiales.



13.8. Ce que vous avez appris dans ce chapitre

- Vous avez vu comment dessiner plusieurs types de damiers et les utiliser.
- Vous pouvez déplacer des objets par la méthode du glisser-déposer.
- Vous avez rencontré plusieurs exercices concernant le jeu d'échecs, qui sera le sujet du chapitre suivant.