Puissance 4

Maillot Valérie
Projet d'ISN
TS°3

Introduction

Nous avons décidé de créer comme programme le jeu puissance 4. Le principe de ce jeu est simple : il suffit d'aligner 4 jetons identiques que ce soit horizontal, vertical ou en diagonal. Les taches les plus compliquées ont été d'une part trouver un algorithme léger et dans une autre part de le transformer en langage de Python. Nous ne sommes pas réparti ces taches car nous avons jugé plus rentable de travailler tous ensemble sur le programme en proposant nos idées et de voir lesquels seraient mieux à tels ou tels endroits.

Le programme se compose en plusieurs parties :

- Les importations de bibliothèques
- La définition du tableau
- Les coordonnées initiales du jeton
  - Le fonction des détections des alignements possibles
  - Le programme principal

Importation

```
4   from scipy import*
5   from Tkinter import *
6   import time
7   import winsound
8   import Tkinter
9   import threading
10  import os
```

- Les lignes 4 à 10 sont consacrées à l'importation de différentes bibliothèques qui permettent l'utilisation de différentes fonctions qui n'existent pas de base dans Python.

- Note : Toutes les bibliothèques ne sont pas utilisées dans le programme, telles que Tkinter, winsound, threading.

Définition du tableau

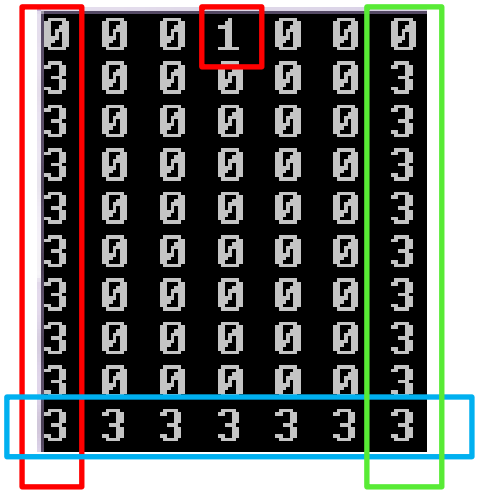
```
13  # Définir le tableau de jeu
14  def affiche_tableau (x1,y1):
15      os.system("cls")    # Toujours sur le même tableau
16      for y in range (y1):
17          for x in range (x1):
18              print t[y,x],
19          print
20
21  t=zeros([10,7], int)
```

Les lignes 13 à 21 sont consacrées à la définition du tableau.

Nous avons décidé de créer un tableau de 10 lignes (y) et 7 colonnes (x) grâce à une fonction « def affiche_tableau ». A la ligne 15 apparaît la fonction « os.system(« cls ») qui permet de réinitialiser le tableau à chaque nouveau mouvement.

Le tableau

```
70  # Bords du tableau
71  t[9,]= 3
72  for x in range(10):
73      t[x,6]= 3
74      t[x,0]= 3
75
```



Cette partie du programme permet de créer des bordures au tableau avec des 3.
t[9,] = 3 : la dixième ligne est remplacé par des trois
t[x,6] = 3 : la septième colonne est remplacé par des trois
t[x, 0] = 3 : la première colonne est remplacé par des trois

Coordonnées

```
77  # Coordonnées initiales de la partie
78  Py1 = 0
79  Px1 = 3
80  t[Py1, Px1] = 1
81  #Py2 = 0
82  #Px2 = 3
83  t[0,0] = 0
84  t[0,6] = 0
85  affiche_tableau(7,10)
86  print
```

De la ligne 78 à la ligne 86, nous définissons les coordonnées du jeton en sa position initiale ([Py1 = 0, Px1=3]) c'est-à-dire à la première ligne quatrième colonne (voir ci-dessus). Nous avons également mis des zéros à la première et dernière colonnes de la première ligne (t[0,0] = 0 ; t[0,6] = 0) pour laisser un libre déplacement au jeton bien qu'il ne pourra pas descendre.

La détection du gagnant

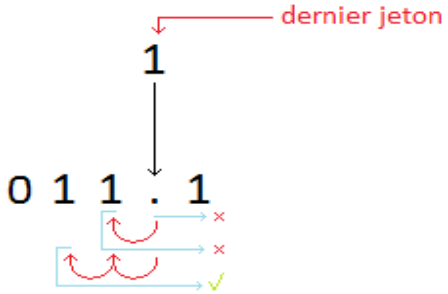
- Les lignes 26 et 66 sont consacrées à la définition de la fonction détection du jeton gagnant. À la ligne 26 on commence par annoncer la définition et commencer la boucle.

```
25    # Définir la fonction détection
26    def detection (x,y, jeton):
27
```

I) Détection horizontale

```
28        # Détection horizontale
29        if (t[y, x+1] == jeton) and (t[y, x+2] == jeton) and (t[y, x+3] == jeton):
30            print "Bravo, le joueur",jeton,"gagne !"
31            exit()
32
33        while t[y, x-1] == jeton :
34            x = x-1
35            if (t[y, x+1] == jeton) and (t[y, x+2] == jeton) and (t[y, x+3] == jeton):
36                print "Bravo, le joueur",jeton,"gagne !"
37                exit()
```

De la ligne 29 à la ligne 37, on détecte l'alignement horizontal du jeton gagnant. La première boucle (l 29-31) est la détection dite « parfaite », c'est-à-dire que le dernier jeton posé est à l'extrême droite de la détection ainsi le programme vérifie les trois derniers jeton pour voir si il y a un alignement. La boucle suivante (l 33-37) est un peu plus compliquée, elle consiste à remonter vers la gauche tant qu'il s'agit du bon jeton (1) pour vérifier par la suite les trois jetons qui suivent pour voir si il y a alignement (2) (voir ci-dessous.) .



II) Détection verticale

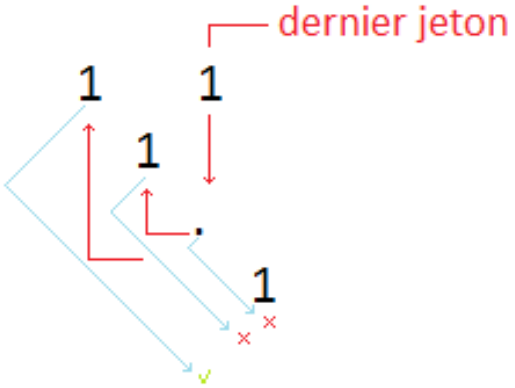
```
39      # Détection verticale
40      if (t[y+1, x] == jeton) and (t[y+2, x] == jeton) and (t[y+3, x] == jeton):
41          print "Bravo, le joueur",jeton,"gagne !"
42          exit()
```

- Il s'agit de la détection la plus facile à programmer, car il suffisait de vérifier si, à partir du dernier jeton tombé, les trois jetons du dessous sont les mêmes.

III) Détection en diagonale (en haut à gauche vers en bas à droite)

```
44      # Détection diagonale de la gauche haut vers la droite bas
45      if (t[y+1, x+1] == jeton) and (t[y+2, x+2] == jeton) and (t[y+3, x+3] == jeton):
46          print "Bravo, le joueur",jeton,"gagne !"
47          exit()
48
49      while t[y-1, x-1] == jeton :
50          x = x-1
51          y = y-1
52          if (t[y+1, x+1] == jeton) and (t[y+2, x+2] == jeton) and (t[y+3, x+3] == jeton):
53              print "Bravo, le joueur",jeton,"gagne !"
54              exit()
```

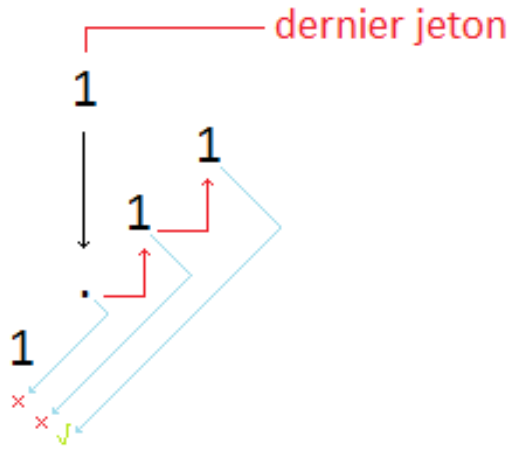
De la ligne 44 à la ligne 54, on détecte les alignements en diagonale qui vont d'en haut à gauche d'en bas à droite. La première boucle (l 45 à 47) est la détection dite « parfaite », c'est-à-dire que le dernier jeton tombé est tout en haut de la diagonale ainsi le programme vérifie les autres jetons pour voir si il y a alignement. La boucle suivante (l 49-54) permet de remonter en haut de la diagonale tant qu'il s'agit du même jeton afin de vérifier les autres jetons pour voir si il y a un alignement.



IV) Détection en diagonale (en haut à droite vers en bas à gauche)

```
56      # Détection diagonale de la droite haut vers la gauche bas
57      if (t[y+1, x-1] == jeton) and (t[y+2, x-2] == jeton) and (t[y+3, x-3] == jeton):
58          print "Bravo, le joueur",jeton,"gagne !"
59          exit()
60
61      while t[y-1, x+1] == jeton :
62          x = x+1
63          y = y-1
64          if (t[y+1, x-1] == jeton) and (t[y+2, x-2] == jeton) and (t[y+3, x-3] == jeton):
65              print "Bravo, le joueur",jeton,"gagne !"
66              exit()
```

De la ligne 56 à la ligne 66, on détecte les alignements en diagonale qui vont d'en haut à droite d'en bas à gauche. La première boucle (l 56 à 59) est la détection dite « parfaite », c'est-à-dire que le dernier jeton tombé est tout en haut de la diagonale ainsi le programme vérifie les autres jetons pour voir si il y a alignement. La boucle suivante (l 61-66) permet de remonter en haut de la diagonale tant qu'il s'agit du même jeton afin de vérifier les autres jetons pour voir si il y a un alignement.



On peut lire à la fin de chaque boucle de détection « print "Bravo, le joueur", jeton, "gagne! " » ce qui permet d'écrire après une victoire en bas du tableau le jeton gagnant. On peut également lire « exit() » ce qui permet de quitter la fenêtre de jeu après victoire en appuyant simplement sur la touche espace.

Le Programme principal

- Les lignes 94 à 157 sont consacrées au programme principal. Il est divisé en deux parties : une pour le jeton 1 et une pour le jeton 2, elles-mêmes divisées en autres parties. A ligne 94, on a créé une boucle infinie dans laquelle tout va se passer.

```
94  while 1:
95      # Déplacement jeton 1
96      while touche <> "2":
97          touche = raw_input("Jeton 1 ~ droite : 6 ; gauche : 4 ; lacher : 2 ")
98          if touche == "6":
99              if t[Py1, Px1+1] == 0:
100                 t[Py1,Px1] = 0
101                 Px1 = Px1 + 1
102                 t[Py1,Px1] = 1
103                 affiche_tableau(7,10)
104
105         if touche == "4":
106             if t[Py1, Px1-1] == 0:
107                 t[Py1,Px1] = 0
108                 Px1 = Px1 - 1
109                 t[Py1,Px1] = 1
110                 affiche_tableau(7,10)
111
112     if touche == "2":
113         while t[Py1 + 1, Px1] ==0:  # Pour que le jeton tombe tout seul
114             t[Py1,Px1] = 0
115             Py1 = Py1 + 1
116             t[Py1,Px1] = 1
117             affiche_tableau(7,10)
118             time.sleep(0.3)   # Pour qu'il se déplace à 0.3s/pas
119         Px1s = Px1
120         Py1s = Py1
121         detection (Px1s,Py1s, 1)
122         Py2 = 0
123         Px2 = 3
124         t[Py2, Px2] = 2
125         affiche_tableau(7,10)
```

Jeton 1

De la ligne 96 à 125, il s'agit des boucles de déplacement du jeton 1. La première boucle (l 96-103) permet de déplacer le jeton droite grâce à la touche 6 du clavier numérique. Le deuxième boucle (l 105-110) permet de déplacer le jeton vers la gauche grâce au bouton 4 du clavier. La dernier boucle permet de lâcher le jeton à une vitesse de 0.3s/pas grâce à la fonction « time.sleep(0.3)», en appuyant sur la touche 2 du clavier, une fois le jeton 1 atterri, le jeton 2 prend les coordonnées initiales du jeton 1.

```
127     # Déplacement jeton 2
128     while touche <> "s":
129         touche = raw_input("Jeton 2 ~ droite : d ; gauche : q ; lacher : s ")
130         if touche == "d":
131             if t[Py2, Px2+1] == 0:
132                 t[Py2,Px2] = 0
133                 Px2 = Px2 + 1
134                 t[Py2,Px2] = 2
135                 affiche_tableau(7,10)
136
137         if touche == "q":
138             if t[Py2, Px2-1] == 0:
139                 t[Py2,Px2] = 0
140                 Px2 = Px2 - 1
141                 t[Py2,Px2] = 2
142                 affiche_tableau(7,10)
143
144     if touche == "s":
145         while t[Py2 + 1, Px2] ==0:  # Pour que le jeton tombe tout seul
146             t[Py2,Px2] = 0
147             Py2 = Py2 + 1
148             t[Py2,Px2] = 2
149             affiche_tableau(7,10)
150             time.sleep(0.3)   # Pour qu'il se déplace à 0.3s/pas
151         Px2s = Px2
152         Py2s = Py2
153         detection (Px2s,Py2s, 2)
154         Py1 = 0
155         Px1 = 3
156         t[Py1, Px1] = 1
157         affiche_tableau(7,10)
```

Jeton 2

- De la ligne 128 à 157, il s'agit des boucles de déplacement du jeton 2. La première boucle (l 128-135) permet de déplacer le jeton droite grâce à la touche « d » du clavier. Le deuxième boucle (l 137-142) permet de déplacer le jeton vers la gauche grâce au bouton « q » du clavier. La dernier boucle (l 144-157) permet de lâcher le jeton à une vitesse de 0.3s/pas grâce à la fonction « time.sleep(0.3)», en appuyant sur la touche « s » du clavier, une fois le jeton 2 atterri, le jeton 1 reprend ses coordonnées initiales.

Note: Pour pouvoir user des touches du clavier nous devions utiliser une fonction à part que nous avons appelé « touche » et que nous avons préalablement définie (l 90). Aux lignes 97 et 129 nous faisions appel à la fonction en définissant quels touches nous voulons utiliser.

Problèmes rencontrés

Nous n'avons pas rencontrés de problèmes vraiment contraignants mis à part le graphisme, que nous avons décidé de ne pas mettre, car nous raisonnions petit à petit avec un algorithme qu'on traduisait ensuite en langage python. Cependant :

- Nous avons eu des difficultés à mettre en jeu un deuxième jeton à cause d'un problème de coordonnées (car nous voulions mettre un jeton à la première et dernière colonne de la première ligne), nous avons finalement décidé de faire apparaître les jetons un par un, ce qui d'ailleurs permettra aux joueurs de rester fair-play !

- Nous avons eu des complications au niveau des différentes détections d'un jeton, nous ne savions pas comment faire pour que le programme dissocie les deux jetons, pour cela nous avons rajouté à la fonction détection (l 26) la possibilité de choisir jeton 1 ou jeton 2. Qu'on a ensuite utilisé tout le long de la fonction sans spécifier de jeton particulier.