```
1  ######### Ceci est une variante du fichier Apprentissage sans interface graphique
   (GUI = Graphic User Interface)
2  import Milieu_nogui
3  import threading
4  import time
5  import marshal
6  import math
7  import matplotlib.pyplot as plt
8
9
10 def init_var(n):
11     global actions,etats,Q,cpteur,listescores
12     import Milieu_nogui
13     Milieu_nogui.x=n
14     Milieu_nogui.y=n
15     listescores=[]
16     cpteur=0
17     actions = Milieu_nogui.actions
18     etats = []
19     Q = {}
20
21 def init_mat():
22     #on initialise tous les états possibles , soit tous les tuples (a,b) où a,b
       appartiennent à [|0;x|]*[|0;y|]
23     for i in range(Milieu_nogui.x):
24         for j in range(Milieu_nogui.y):
25             etats.append((i, j))
26
27
28     for etat in etats:
29         temp = {}
30         for action in actions:
31             temp[action] = 0.1
32         Q[etat] = temp
33
34     for (i, j, c, w) in Milieu_nogui.special:
35         for action in actions:
36             Q[(i, j)][action] = w
37
38 def signal_arret(liste,stop):
39 #cette fonction nous permet d'envoyer un signal d'arret au programme principal
   quand le score a convergé
40 #on considère qu'il y a convergence lorsqu'un même score se répète 10 fois
41     n=len(liste)
42     if n<stop+1:
43         return False
44     aux= liste[n-1]
45
46     for k in range(n-2,n-stop,-1):
47         if liste[k] != aux:
48             return False
49         aux = liste[k]
50     return True
51
52 def faire(action):
53     s = Milieu_nogui.coordonnees
54     r = -Milieu_nogui.score
55
56     if action == actions[0]: #up
57         Milieu_nogui.mouvement(0, -1)
58     elif action == actions[1]: #down
59         Milieu_nogui.mouvement(0, 1)
60     elif action == actions[2]: #left
61         Milieu_nogui.mouvement(-1, 0)
62     elif action == actions[3]: #right
63         Milieu_nogui.mouvement(1, 0)
64     else:
65         return
66     s2 = Milieu_nogui.coordonnees
```

```
67          r += Milieu_nogui.score
68          return s, action, r, s2
69
70      def max_Q(s): #retourne la plus grande valeur de Q[s] et l'action associée
71          val ,act= 0,0
72          for a, q in Q[s].items():
73              if val == 0 or (q > val):
74                  val , act = q,a
75          return act, val
76
77      def inc_Q(s, a, alpha, inc):
78          Q[s][a] = round((1-alpha)*Q[s][a] + alpha*inc , 5) #incrémente la valeur de
            la matrice Q
79
80      def lancer(gamma,alpha):
81
82          global cpteur,listescores
83          time.sleep(1)
84          t = 1
85          while not signal_arret(listescores,5):
86              # Choix de l'action menant à la meilleure récompense
87              s = Milieu_nogui.coordonnees
88              max_act, max_val = max_Q(s)
89              (s, a, r, s2) = faire(max_act)
90
91              # Modification de la matrice Q
92              max_act, max_val = max_Q(s2)
93              inc_Q(s, a, alpha, r + gamma * max_val)
94              #print(Q[s],Milieu_nogui.score)
95
96              # on veut savoir si le jeu a redémarré
97              t += 1.0
98              if Milieu_nogui.etat_reinit() or abs(Milieu_nogui.score) > 10000: #la
                deuxieme condition est nécessaire en cas de blocage sur une case non
                terminale
99                  Milieu_nogui.reinitialisation(listescores)
100                 cpteur += 1
101                 #time.sleep(0.001)
102                 t = 1.0
103
104             # vitesse de rafraichissement de l'interface graphique
105             #time.sleep(1/vit)
106
107     ######## Methodes de tracé graphique ##########
108
109
110     def moyenne(l):
111         s=0
112         for elem in l:
113             s+=elem
114         return s/len(l)
115
116     nombre_iterations=1
117     #for taillematrice in range(15,21):
118     listegamma,listegen=[],[[],[]]
119     #Milieu_nogui.ecriture_obstacles("stockage.txt",taillematrice,taillematrice,Milieu_n
        ogui.obstaclage)
120     taillematrice=20
121
122     gamma=[99,100]
123
124     for gamma_var in [0.01*i for i in range(gamma[0],gamma[1])]:
125         liste_lissage=[[],[]]
126         for k in range(nombre_iterations):
127             t0=time.time() #référence de temps, suivi du temps d'exécution
128
129             init_var(taillematrice)
130             init_mat()
131             alpha=1
```

```
132
133             lancer(gamma_var,alpha)
134             #thr= threading.Thread(target=lancer(gamma,1))
135             #thr.daemon=True
136             #thr.start
137             liste_lissage[0].append(listescores[len(listescores)-1])
138             liste_lissage[1].append(len(listescores))
139
140             print("taillematrice=",taillematrice,"g=",gamma_var,"
                t=",round(time.time()-t0,3))
141
142         listegamma.append(gamma_var)
143         listegen[0].append(moyenne(liste_lissage[0]))
144         listegen[1].append(moyenne(liste_lissage[1]))
145
146
147     marshal.dump([listegamma,listegen],open("resultats.txt","wb")) #exportation des
    résultats pour analyse ultérieure
148     marshal.dump(Q,open("Q.txt","wb"))  #Exportation de la matrice Q apres calculs :
    chemin optimal
149
150
151     plt.subplot(211)
152     titre= "matrice "+str(Milieu_nogui.x)+"x"+str(Milieu_nogui.y)+" alpha:" +
    str(alpha)# + "(dégréssif en e^-0,1)"
153     plt.title(titre)
154     plt.xlabel("gamma")
155     plt.ylabel("score final")
156     plt.plot(listegamma,listegen[0],label="n="+ str(taillematrice))
157
158     plt.subplot(212)
159     #titre= "matrice "+str(Milieu_nogui.x)+"x"+str(Milieu_nogui.y)+" alpha:" +
    str(alpha) #+ "(dégréssif en e^-0,001)"
160     #plt.title(titre)
161     plt.xlabel("alpha")
162     plt.ylabel("nombre de générations")
163     plt.plot(listegamma,listegen[1],label="n= "+ str(taillematrice))#+ "
    d/da="+str(round((listegen[1][-1]-listegen[1][0])/(listegamma[-1]-listegamma[0]),2))
    )
164
165     plt.legend()
166
167     manager = plt.get_current_fig_manager()
168     manager.resize(*manager.window.maxsize())
169
170     #plt.savefig(titre+'.png')
171     plt.show()
172
173
174
175
176
177
178
179
180
181
```