```
 1
 2  from tkinter import *
 3  import random
 4  import marshal,copy
 5
 6
 7  master = Tk()
 8  taille = 10
 9  (x, y) = (20, 20)
10  actions = ["up", "down", "left", "right"]
11
12  board = Canvas(master, width=x*taille, height=y*taille)
13  coordonnees = (0, y-1)
14  score = 0
15  restart = False
16  walk_reward = -0.05
17  obstaclage=.15
18  nombrebonus=0
19
20  poubelle=[]
21  coffre={}
22  objets={}
23  listescores=[]
24
25  #position des murs
26
27
28  def ecriture_obstacles(chemin,x,y,obstaclage):
29      obstacles= [(random.randint(0,x-1),random.randint(0,y-1)) for mur in
        range(int(x*y*obstaclage))]
30      fichier=open(chemin,"wb")
31      marshal.dump(obstacles,fichier)
32      fichier.close()
33
34  def lecture (chemin):
35
36      fichier=open(chemin,"rb")
37
38      variable= marshal.load(fichier)
39      return(variable)
40
41      fichier.close()
42
43  #ecriture_obstacles("stockage.txt",x,y,obstaclage)
44  obstacles= copy.copy(lecture("stockage.txt"))
45
46
47  obstacles.append((6,14)) #ici on teste une petite modification au chemin optimal
    pour voir si le programme est capab
48
49  #cases bonus/malus
50  special = [(4, 1, "red", -1),(x-1,0 , "yellow", 1),(14,2,"blue",0.5)]
51
52  bonus_aleatoire =[(random.randint(0,x-1),random.randint(0,y-1),"blue",1) for k in
    range(nombrebonus)]
53  for bonus in bonus_aleatoire:
54      special.append(bonus)
55
56
57  def rendu(special, obstacles, taille, x, y, coordonnees): #méthode graphique pour
    afficher le rendu de la grille
58      for i in range(x):
59          for j in range(y):
60              objets[i,j]= board.create_rectangle(i*taille, j*taille, (i+1)*taille,
                (j+1)*taille, fill="#ffffffff", width=1)
61
62      for (i, j, c, w) in special:
63          board.create_rectangle(i*taille, j*taille, (i+1)*taille, (j+1)*taille,
            fill=c, width=1)
```

```
64          for (i, j) in obstacles:
65              board.create_rectangle(i*taille, j*taille, (i+1)*taille, (j+1)*taille,
                fill="black", width=1)
66
67      #def creation(special, obstacles, taille, x, y, coordonnees):
68
69      rendu(special, obstacles, taille, x, y, coordonnees)
70      joueur = board.create_rectangle(coordonnees[0]*taille+taille*2/10,
        coordonnees[1]*taille+taille*2/10,coordonnees[0]*taille+taille*8/10,
        coordonnees[1]*taille+taille*8/10,fill="orange", width=1, tag="joueur")
71      board.grid(row=0, column=0)
72      #   return joueur
73
74      #joueur= creation(special, obstacles, taille, x, y, coordonnees)
75
76      def recup_coul(couleur):#transforme une "couleur" au format tkinter en triplet
        rouge vert bleu d'entiers
77          coul2=""
78          flag=False
79          for elem in couleur:
80              if elem=="#":
81                  flag=True
82              elif flag:
83                  coul2+= elem.upper()
84          r,g,b=int("0x"+coul2[:3],16),int("0x"+coul2[3:6],16),int("0x"+coul2[6:],16)
85          return r,g,b
86
87      def recomp_coul(r,g,b): #retransforme un triplet d'entiers rouge vert bleu au
        format couleur tkinter
88          triplet= [ hex(r)[2:],hex(g)[2:],hex(b)[2:] ] #codes hexadécimaux de la forme
            0x... donc on slice juste après l'indicateur hexadécimal 0x
89          sortie="#"
90          for i in range(3):
91
92              while len(triplet[i])<3:
93                  triplet[i] ='0'+triplet[i]            # correction d'erreur, sinon la
                    sortie n'est pas sous le bon format, il manquera un caractere
94              sortie+=triplet[i]
95
96          return sortie
97
98      def edit_coul(x,y,mod): #modifie la couleur d'une case aux positions x,y
99          r,g,b= recup_coul(board.itemcget(objets[x, y], "fill"))
100         r,g,b= r,max(0,g-mod),b #on diminue la teneur en rouge , pour avoir une case
            de plus en violette
101
102         board.itemconfigure( objets[x, y] , fill=recomp_coul(r,g,b))
103
104
105     def mouvement(dx, dy):
106         global coordonnees, x, y, score, walk_reward, joueur, restart , coffre
107         if restart :
108             reinitialisation()
109         new_x = coordonnees[0] + dx
110         new_y = coordonnees[1] + dy
111         score += walk_reward
112
113         if (new_x >= 0) and (new_x < x) and (new_y >= 0) and (new_y < y) and not
            ((new_x, new_y) in obstacles):
114             board.coords(joueur, new_x*taille+taille*2/10, new_y*taille+taille*2/10,
                new_x*taille+taille*8/10, new_y*taille+taille*8/10)
115             coordonnees = (new_x , new_y)
116             edit_coul(new_x , new_y , 10)
117         else:
118             pass#score += walk_reward #permet de pénaliser les actions inutiles comme
                un mouvement contre un mur
119
120         for (i, j, c, w) in special: # c = couleur w = récompense
121             if new_x == i and new_y == j:
```

```
122
123             if c== "yellow" or c=="red":
124                 score -= walk_reward
125                 score += w
126                 for bonus in coffre:
127                     score+=w*int(coffre[bonus])
128                 restart = True
129                 print("score=",score)
130
131             elif c== "blue":
132                 score+=w
133                 coffre[i,j]=True
134                 #obstacles.append((i,j))
135             else:
136                 poubelle.append((i,j,c,w)) # pour qu'on ne puisse récolter le bonus qu'une seule fois
137                 #obstacles.append(i,j) #pour transformer l'ancien bonus en mur
138                 special.remove((i,j,c,w))
139             return
140
141 def reinitialisation():
142     global coordonnees, score, joueur, restart,listescores,coffre
143     coordonnees = (0, y-1)
144     listescores.append(score)
145     score = 1
146     ###############
147     coffre={}
148     ###############
149     restart = False
150     for tupls in poubelle:
151         special.append(tupls)
152         poubelle.remove(tupls)
153     board.coords(joueur, coordonnees[0]*taille+taille*2/10, coordonnees[1]*taille+taille*2/10, coordonnees[0]*taille+taille*8/10, coordonnees[1]*taille+taille*8/10)
154
155 def etat_reinit():
156     return restart
157
158
159 def gui():
160     master.mainloop()
161
162
```