

Artificial Intelligence Project

Eng.Amr Eledkawy

TEAM MEMBERS:(1:Salma Tarek Abd-elrazik , 2:Rana Hussien Algendi , 3:Rehab Sakr Sakr)

Section (1)

Project Code:

```
1  from glob import glob
2  from tkinter import *
3  import random
4
5
6  def next_turn(row, col):
7      global player
8      if game_btns[row][col]['text'] == "" and check_winner() == False:
9          if player == players[0]:
10             # Put player 1 symbol
11             game_btns[row][col]['text'] = player
12
13             if check_winner() == False:
14                 # switch player
15                 player = players[1]
16                 label.config(text=(players[1] + " turn"))
17
18             elif check_winner() == True:
19                 label.config(text=(players[0] + " wins!"))
20
21             elif check_winner() == 'tie':
22                 label.config(text=("Tie, No Winner!"))
23
24         elif player == players[1]:
25             # Put player 2 symbol
26             game_btns[row][col]['text'] = player
27
28             if check_winner() == False:
29                 # switch player
30                 player = players[0]
31                 label.config(text=(players[0] + " turn"))
32
33             elif check_winner() == True:
34                 label.config(text=(players[1] + " wins!"))
35
36             elif check_winner() == 'tie':
37                 label.config(text=("Tie, No Winner!"))
38
39
40  def check_winner():
41      # check all 3 horizontal conditions
42      for row in range(3):
43          if game_btns[row][0]['text'] == game_btns[row][1]['text'] == game_btns[row][2]['text'] != "":
44             game_btns[row][0].config(bg="cyan")
45             game_btns[row][1].config(bg="cyan")
46             game_btns[row][2].config(bg="cyan")
47             return True
48
49      # check all 3 vertical conditions
50      for col in range(3):
51          if game_btns[0][col]['text'] == game_btns[1][col]['text'] == game_btns[2][col]['text'] != "":
52             game_btns[0][col].config(bg="cyan")
53             game_btns[1][col].config(bg="cyan")
54             game_btns[2][col].config(bg="cyan")
55             return True
56
57      # check diagonals conditions
58      if game_btns[0][0]['text'] == game_btns[1][1]['text'] == game_btns[2][2]['text'] != "":
59         game_btns[0][0].config(bg="cyan")
60         game_btns[1][1].config(bg="cyan")
61         game_btns[2][2].config(bg="cyan")
62         return True
```

```

62         return True
63     elif game_btns[0][2]['text'] == game_btns[1][1]['text'] == game_btns[2][0]['text'] != "":
64         game_btns[0][2].config(bg="cyan")
65         game_btns[1][1].config(bg="cyan")
66         game_btns[2][0].config(bg="cyan")
67         return True
68
69     # if there are no empty spaces left
70     if check_empty_spaces() == False:
71         for row in range(3):
72             for col in range(3):
73                 game_btns[row][col].config(bg='red')
74
75         return 'tie'
76
77     else:
78         return False
79
80
81 def check_empty_spaces():
82     spaces = 9
83
84     for row in range(3):
85         for col in range(3):
86             if game_btns[row][col]['text'] != "":
87                 spaces -= 1
88
89     if spaces == 0:
90         return False
91     else:
92         return True
93
94
95 def start_new_game():
96     global player
97     player = random.choice(players)
98
99     label.config(text=(player + " turn"))
100
101     for row in range(3):
102         for col in range(3):
103             game_btns[row][col].config(text="", bg="#F0F0F0")
104
105
106 window = Tk()
107 window.title("Tic-Tac-Toe Korsat-X-Parmaga")
108
109 players = ["x", "o"]
110 player = random.choice(players)
111
112 game_btns = [
113     [0, 0, 0],
114     [0, 0, 0],
115     [0, 0, 0]
116 ]
117
118 label = Label(text=(player + " turn"), font=('consolas', 40))
119 label.pack(side="top")
120
121 restart_btn = Button(text="restart", font=(
122     'consolas', 20), command=start_new_game)
123 restart_btn = Button(text="restart", font=(
124     'consolas', 20), command=start_new_game)
125 restart_btn.pack(side="top")
126
127 btns_frame = Frame(window)
128 btns_frame.pack()
129
130 for row in range(3):
131     for col in range(3):
132         game_btns[row][col] = Button(btns_frame, text="", font=('consolas', 50), width=4, height=1,
133                                     command=lambda row=row, col=col: next_turn(row, col))
134         game_btns[row][col].grid(row=row, column=col)
135
136 window.mainloop()

```

PEAS:

- Taxi driver agent:
 - P → speed, safe, legal, profit
 - E → street, traffic, customer
 - A → broke, accelerator, horn
 - S → camera, GPS

ODESA:

- Chess with a clock:
 - O (fully, Partial)
 - Fully Observable
 - D (Deterministic, Stochastic, Strategic)
 - Strategic
 - E (Episodic, Sequential)
 - Sequential
 - S (Static, Dynamic , Semi-Dynamic)
 - Semi-dynamic
 - A (Single agent, Multi-agent)
 - Multi-agent

Problem Formulation:

- Airline travel problem (example of Route finding):

**Initial State:*

specified by the problem

**Successor function:*

the states resulting from taking
any scheduled flight, leaving later
than the current time.

**Goal test:*

are we at the destination on time.

**Path cost:*

depends on monetary cost, waiting
time, etc.