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*:*

```
from glob import glob
from tkinter import *
import random


def next_turn(row, col):
    global player
    if game_btns[row][col]['text'] == "" and check_winner() == False:
        if player == players[0]:
            # Put player 1 sympol
            game_btns[row][col]['text'] = player

            if check_winner() == False:
                # switch player
                player = players[1]
                label.config(text=(players[1] + " turn"))

            elif check_winner() == True:
                label.config(text=(players[0] + " wins!"))

            elif check_winner() == 'tie':
                label.config(text=("Tie, No Winner!"))

        elif player == players[1]:
            # Put player 2 sympol
            game_btns[row][col]['text'] = player

            if check_winner() == False:
                # switch player
                player = players[0]
                label.config(text=(players[0] + " turn"))

            elif check_winner() == True:
                label.config(text=(players[1] + " wins!"))

            elif check_winner() == 'tie':
                label.config(text=("Tie, No Winner!"))


def check_winner():
    # check all 3 horizontal conditions
    for row in range(3):
        if game_btns[row][0]['text'] == game_btns[row][1]['text'] == game_btns[row][2]['text'] != "":
            game_btns[row][0].config(bg="cyan")
            game_btns[row][1].config(bg="cyan")
            game_btns[row][2].config(bg="cyan")
            return True

    # check all 3 vertical conditions
    for col in range(3):
        if game_btns[0][col]['text'] == game_btns[1][col]['text'] == game_btns[2][col]['text'] != "":
            game_btns[0][col].config(bg="cyan")
            game_btns[1][col].config(bg="cyan")
            game_btns[2][col].config(bg="cyan")
            return True

    # check diagonals conditions
    if game_btns[0][0]['text'] == game_btns[1][1]['text'] == game_btns[2][2]['text'] != "":
        game_btns[0][0].config(bg="cyan")
        game_btns[1][1].config(bg="cyan")
        game_btns[2][2].config(bg="cyan")
        return True
```

```
        return True
    elif game_btns[0][2]['text'] == game_btns[1][1]['text'] == game_btns[2][0]['text'] != "":
        game_btns[0][2].config(bg="cyan")
        game_btns[1][1].config(bg="cyan")
        game_btns[2][0].config(bg="cyan")
        return True

    # if there are no empty spaces left
    if check_empty_spaces() == False:
        for row in range(3):
            for col in range(3):
                game_btns[row][col].config(bg='red')

        return 'tie'

    else:
        return False


def check_empty_spaces():
    spaces = 9

    for row in range(3):
        for col in range(3):
            if game_btns[row][col]['text'] != "":
                spaces -= 1

    if spaces == 0:
        return False
    else:
        return True


def start_new_game():
    global player
    player = random.choice(players)

    label.config(text=(player + " turn"))

    for row in range(3):
        for col in range(3):
            game_btns[row][col].config(text="", bg="#F0F0F0")


window = Tk()
window.title("Tic-Tac-Toe Korsat-X-Parmaga")

players = ["x", "o"]
player = random.choice(players)

game_btns = [
    [0, 0, 0],
    [0, 0, 0],
    [0, 0, 0]
]

label = Label(text=(player + " turn"), font=('consolas', 40))
label.pack(side="top")

restart_btn = Button(text="restart", font=(
    'consolas', 20), command=start_new_game)
restart_btn = Button(text="restart", font=(
    'consolas', 20), command=start_new_game)
restart_btn.pack(side="top")

btns_frame = Frame(window)
btns_frame.pack()

for row in range(3):
    for col in range(3):
        game_btns[row][col] = Button(btns_frame, text="", font=('consolas', 50), width=4, height=1,
                                     command=lambda row=row, col=col: next_turn(row, col))
        game_btns[row][col].grid(row=row, column=col)

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.