Tic Tac Toe Game

Özet

Bu proje, Python programlama dilinde geliştirilen bir Tic Tac Toe (XOX) oyununu içermektedir. Proje, Tkinter kütüphanesi kullanılarak oluşturulan bir masaüstü uygulaması şeklinde sunulur.

Oyun, kullanıcı dostu bir arayüz ve temel oyun mantığıyla tasarlanmıştır.

Projenin ana özellikleri şunlardır:

- Oyuncular X ve O olarak belirlenir ve sırayla tahta üzerindeki boş kutucuklara işaret koyarlar.
- Kazananı belirlemek için her hamle sonrasında oyun durumu kontrol edilir.
- Oyunculardan biri üç işareti yan yana, çapraz veya dikey olarak birleştirirse oyunu kazanır.
- Eğer tüm kutucuklar dolu hale gelirse oyun berabere biter.
- Oyunun skoru takip edilir ve her kazanma durumunda ilgili oyuncunun skoru bir artırılır.

Projede ayrıca "Yapay Zeka (AI) Modu" da bulunmaktadır:

- Bu modda oyuncu, bilgisayara karşı oynar.
- Bilgisayar, minimax algoritması kullanarak en iyi hamleyi belirler ve oyuncunun karşısına çıkar.
- Bu şekilde oyuncular, hem birbirleriyle hem de yapay zekayla rekabet edebilirler.

Proje, geleneksel Tic Tac Toe oyununu modern bir masaüstü uygulaması şeklinde sunması ve yapay zeka moduyla kullanıcılara farklı oyun deneyimleri sunmasıyla özgündür. Kullanıcı dostu arayüzü, basit oyun mantığı ve akıllı bir yapay zeka stratejisiyle birleştiğinde, oyunculara keyifli bir oyun deneyimi sağlamaktadır.

Projenin Amacı:

Bu proje, Tic Tac Toe oyununu Python programlama dili kullanarak geliştirmek ve kullanıcılar için interaktif bir oyun deneyimi sunmaktadır. Amacı, eğlenceli bir masaüstü oyunu oluşturarak kullanıcıların strateji becerilerini geliştirmelerine, rekabetçi bir ortamda keyifli vakit geçirmelerine olanak tanımaktadır.

Projenin Hedefi:

Projenin hedefi, kullanıcıların Tic Tac Toe oyununu basit ve kullanıcı dostu bir arayüzle oynayabilmelerini sağlamaktadır. Ayrıca, yapay zeka moduyla birlikte oyunculara farklı seçenekler sunarak, hem diğer oyuncularla rekabet edebilme hem de bilgisayara karşı oynayabilme imkanı sağlamaktadır. Hedef, oyunun keyifli, adil ve bağımlılık yapıcı bir deneyim sunması ve kullanıcıların zamanlarını eğlenceli bir şekilde geçirebilmelerini sağlamaktadır.

Konu, Kapsam ve Literatür Özeti:

Konu: Bu proje, Tic Tac Toe oyununu kullanıcı dostu bir arayüzle sunarak, kullanıcılara oyunu oynama ve keyifli vakit geçirme imkanı sağlamayı amaçlamaktadır. Ayrıca, yapay zeka moduyla birlikte kullanıcılara bilgisayara karşı da oynayabilme seçeneği sunmaktadır.

Kapsam: Proje, Python programlama dili ve Tkinter kütüphanesi kullanılarak geliştirilmiştir. Kullanıcılar, basit ve anlaşılır bir arayüz üzerinden oyunu oynayabilir, skor takibini yapabilir, oyunu yeniden başlatabilir ve çıkış yapabilirler. Proje, kullanıcıların tek başlarına oynamaları için 1 oyunculu modu ve başka bir kullanıcıyla karşılıklı oynamaları için 2 oyunculu modu desteklemektedir. Ayrıca, yapay zeka moduyla bilgisayara karşı oynanabilme özelliği de bulunmaktadır.

Literatür Özeti: Tic Tac Toe oyunu, klasik bir masa oyunudur ve literatürde çeşitli kaynaklarda ele alınmıştır.

Projenin geliştirilmesinde, Python programlama dili ve Tkinter kütüphanesi kullanılmıştır. Python ve Tkinter hakkında kaynaklara başvurulmuş, kullanım örnekleri incelenmiş ve bu bilgiler proje geliştirme sürecinde uygulanmıştır.

Ayrıca, yapay zeka algoritması olarak Minimax algoritması literatürde yaygın olarak kullanılan bir yaklaşımdır ve proje bu algoritmayı kullanarak yapay zeka modunu gerçekleştirmektedir. Minimax algoritması hakkında literatürdeki kaynaklar üzerinden bilgi edinilmiş ve projeye uyarlanmıştır.

Yöntem: Projenin geliştirilmesi aşamasında aşağıdaki yöntemler kullanılmıştır:

Programlama Dili ve Kütüphane Seçimi: Proje, Python programlama dili kullanılarak geliştirilmiştir. Python, kullanımı kolay ve hızlı bir dil olduğu için tercih edilmiştir. Arayüz tasarımı için Tkinter kütüphanesi kullanılmıştır. Tkinter, Python'un standart kütüphanelerinden biridir ve GUI (Grafik Kullanıcı Arayüzü) uygulamaları oluşturmak için yaygın olarak kullanılan bir araçtır.

Oyun Tahtası Temsili: 3x3'lük bir matris, oyun tahtasını temsil etmek için kullanılmıştır. Bu matris, boşluklar için boş karakterlerle (") başlatılmış ve kullanıcıların hamlelerini kaydetmek için güncellenmiştir.

Oyun Kontrolü: Oyunun akışını kontrol etmek için çeşitli fonksiyonlar kullanılmıştır. Hamlelerin kontrolü, kazananın belirlenmesi, beraberlik durumunun kontrolü gibi işlemler fonksiyonlar aracılığıyla gerçekleştirilmiştir.

Oyuncu ve Skor Takibi: Proje, iki oyuncu (X ve O) arasında skor takibini sağlamaktadır. Her oyuncunun kazandığı oyunlar sayısı skor olarak tutulmuş ve arayüzde güncellenmiştir.

Yapay Zeka Modu: Yapay zeka modu, bilgisayara karşı oynama seçeneği sunmaktadır. Bu modda, Minimax algoritması kullanılarak bilgisayarın hamleleri belirlenmektedir. Minimax algoritması, tüm olası hamleleri değerlendirerek en iyi hamleyi seçen bir rekürsif algoritmadır.

Arayüz Tasarımı: Tkinter kütüphanesi kullanılarak basit ve kullanıcı dostu bir arayüz tasarlanmıştır. Oyun tahtası, düğmeler ve skor takibi gibi bileşenler arayüzde yer almaktadır. Ayrıca, oyunun başlatılması, yeniden başlatılması ve çıkış yapılması gibi işlevler de arayüz üzerindeki düğmeler aracılığıyla gerçekleştirilmektedir.

Bu yöntemler kullanılarak, Tic Tac Toe oyunu için kullanıcı dostu bir arayüz oluşturulmuş, oyunun akışı kontrol edilmiş, skor takibi sağlanmış ve yapay zeka modu eklenmiştir.

Projede Yapılan Çalışmalar:

```
import tkinter as tk
from tkinter import messagebox
import sys
import copy

# TicTacToe oyununun ana sınıfı.
class TicTacToe:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("Tic Tac Toe")
        self.window.geometry("600x500")

# Oyun tahtasını temsil eden 3x3 matris.
        self.game = [['' for _ in range(3)] for _ in range(3)]
        self.players = ['X', 'O']
        self.current_player = self.players[0]
        self.score_x = 0
        self.score_o = 0
        self.ai_mode = False

        main_frame = tk.Frame(self.window, width=600, height=400)
        main_frame.pack(pady=50)

        self.buttons = [[None for _ in range(3)] for _ in range(3)]
        for i in range(3):
            for j in range(3):
                self.buttons[i][j] = tk.Button(main_frame, text='',
command=lambda i=i, j=j: self.click(i, j), height=3, width=6)
                self.buttons[i][j].grid(row=i, column=j, padx=10, pady=10)

        restart_button = tk.Button(self.window, text='Restart',
command=self.restart)
        restart_button.pack(pady=10)

        ai_button = tk.Button(self.window, text='Play with AI',
command=self.toggle_ai_mode)
        ai_button.pack(pady=10)

        exit_button = tk.Button(self.window, text='Exit',
command=self.window.quit)
        exit_button.pack(pady=10)

        self.score_label = tk.Label(self.window, text='Score: X - {}   O -
{}'.format(self.score_x, self.score_o))
        self.score_label.pack(pady=10)
```

```
        self.click_count = 0

    def click(self, i, j):
        if self.game[i][j] == '' and not self.check_winner():
            self.buttons[i][j]['text'] = self.current_player
            self.game[i][j] = self.current_player
            if self.check_winner():
                messagebox.showinfo("Game Over", "{} Oyuncu
Kazandı!".format(self.current_player))
                if self.current_player == 'X':
                    self.score_x += 1
                else:
                    self.score_o += 1
                self.update_score()
            elif self.check_draw():
                messagebox.showinfo("Game Over", "Berabere!")
            else:
                self.current_player =
self.players[(self.players.index(self.current_player) + 1) % 2]

            self.click_count += 1
            if self.click_count % 2 == 1:
                self.buttons[i][j]['bg'] = '#A0C49D'
            else:
                self.buttons[i][j]['bg'] = '#E1ECC8'

            if self.ai_mode and self.current_player == 'O' and not
self.check_winner() and not self.check_draw():
                self.ai_move()

# Bir oyuncunun kazanıp kazanmadığını kontrol eder.
    def check_winner(self):
        for i in range(3):
            if self.game[i][0] == self.game[i][1] == self.game[i][2] != '':
                return True
            if self.game[0][i] == self.game[1][i] == self.game[2][i] != '':
                return True
        if self.game[0][0] == self.game[1][1] == self.game[2][2] != '':
            return True
        if self.game[0][2] == self.game[1][1] == self.game[2][0] != '':
            return True
        return False

# Oyunun berabere olup olmadığını kontrol eder.
    def check_draw(self):
        for row in self.game:
            if '' in row:
                return False
        return True
```

```
# Oyunu yeniden başlatır.
    def restart(self):
        self.game = [['' for _ in range(3)] for _ in range(3)]
        for i in range(3):
            for j in range(3):
                self.buttons[i][j]['text'] = ''
                self.buttons[i][j]['bg'] = 'SystemButtonFace'
        self.current_player = self.players[0]
        self.click_count = 0

# Skoru günceller.
    def update_score(self):
        self.score_label['text'] = 'Score: X - {}   O -
{}'.format(self.score_x, self.score_o)

# AI modunu açar.
    def toggle_ai_mode(self):
        self.ai_mode = not self.ai_mode
        self.restart()

    def ai_move(self):
        best_score = -sys.maxsize
        move = None

        for i in range(3):
            for j in range(3):
                if self.game[i][j] == '':
                    self.game[i][j] = 'O'
                    score = self.minimax(self.game, 0, False)
                    self.game[i][j] = ''
                    if score > best_score:
                        best_score = score
                        move = (i, j)

        self.click(*move)


# Minimax kodları.
    def minimax(self, board, depth, is_maximizing):
        if self.check_winner():
            if is_maximizing:
                return -1
            else:
                return 1
        elif self.check_draw():
            return 0

        if is_maximizing:
```

```
            best_score = -sys.maxsize
            for i in range(3):
                for j in range(3):
                    if board[i][j] == '':
                        board[i][j] = 'O'
                        score = self.minimax(board, depth + 1, False)
                        board[i][j] = ''
                        best_score = max(score, best_score)
            return best_score
        else:
            best_score = sys.maxsize
            for i in range(3):
                for j in range(3):
                    if board[i][j] == '':
                        board[i][j] = 'X'
                        score = self.minimax(board, depth + 1, True)
                        board[i][j] = ''
                        best_score = min(score, best_score)
            return best_score

# Oyunu başlatır.
    def run(self):
        self.window.mainloop()

if __name__ == "__main__":
    game = TicTacToe()
    game.run()
```