

בניית משחק
איקס עיגול

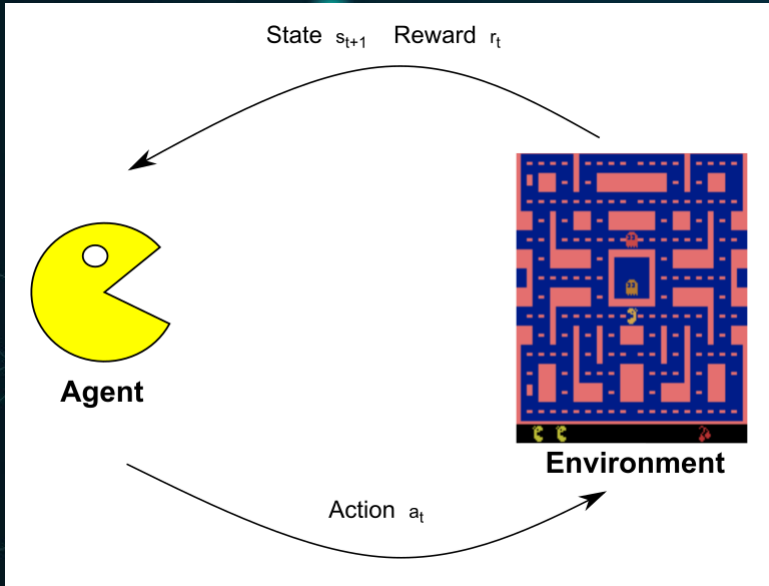
גלעד מרקמן



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

בניית משחק לפי מודל סביבה לקוח

• נדגים בניית משחק בהתאם למודל סביבה לקוח באמצעות בניית משחק איקס עיגול.



• המשחק יכול את המחלקות הבאות:

• סביבה (Environment): Tic_Tac_Toe

• סוכן (Agent): Human_Agent

• מצב: state

• פעולה (action): Tuple (row, col)

• גרפיקה: Graphics

• Game : התוכנית הראשית עם לולאת המשחק.

מחלקות התוכנית

Environment: Tic_Tac_Toe		
מאפיין	מצב – state	המצב הנוכחי של הסביבה
פונקציה	Move(action)	הפונקציה משנה את המצב הנוכחי בהתאם לפעולה שמקבלת.
פונקציה	isLegal(action)	הפונקציה בודקת אם הפעולה חוקית
פונקציה	isEndOfGame(state)	הפונקציה מחזירה אמת אם המצב הינו סוף משחק.
פונקציה	Next_state(state, action)	הפונקציה מקבלת מצב ופעולה ומחזירה את המצב הבא (בדומה ל- move הפועלת על המצב של הסביבה)

state		
מאפיין	Board	מערך המתאר את מצב המשחק
מאפיין	Player	תור של מי לשחק
פעולה	Init_board()	יצירת מצב התחלתי של המחשב

Action = Tuple (row, col)

מחלקות התוכנית - המשך

Human_Agent		
מאפיין	סביבה – env	הסביבה בה פועל הסוכן
פעולה	get-action(state)	הסוכן מקבל מצב מסויים ומחזיר את הפעולה אותה הוא בוחר לבצע.

Graphics		
מאפיין	Screen	מסך ראשי של PyGame
פעולה	Draw(state)	הפעולה מקבלת מצב ומדפיסה אותו על המסך
פעולה	Write(text)	פעולה כותבת טקסט על המסך
פעולה	calc_row_col(pos)	פעולה מקבלת מיקום העכבר ומחזירה את השורה והעמודה המתאימים למיקום
פעולה	Calc_pos(row_col)	הפעולה מקבלת שורה ועמודה בלוח ומחזירה את המיקום במשטח pyGame.

סוכן אדם – Human Agent

```
from TicTacToe import TicTacToe
import pygame
from Graphics import *

class Human_Agent:
    def __init__(self, player, env: TicTacToe, graphics: Graphics):
        self.env = env
        self.player = player
        self.graphics = graphics

    def get_action(self, events):
        for event in events:
            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos()
                action = self.graphics.calc_row_col(pos)
                if self.env.legal(self.env.state, action):
                    return action
        return None

    def __call__(self, events):
        return self.get_action(events)
```

• המחלקה Graphics

```
def calc_row_col (self, pos):
    x, y = pos
    if y < 100:
        return None
    row = (y-H_HEIGHT) // SQUARE_SIZE
    col = x // SQUARE_SIZE
    return row, col
```



הוספת שחקנים ללולאה הראשית

```
player1 = Human_Agent(1, env, graphics)
player2 = Human_Agent(2, env, graphics)
player = player1

def main ():
    run = True

    while (run):
        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                run = False
        action = player(events)
        if action:
            env.move(action)
            player = switch_players(player)
        graphics(env.state)
        pygame.display.update()
        clock.tick(FPS)
```

```
def switch_players(player):
    if player == player1:
        return player2
    else:
        return player1
```

הוספת פעולות בסביבה

```
def move (self, action):  
    self.state.board[action] = self.state.player  
    self.switch_players(self.state)  
    self.end_of_game(self.state)  
  
def legal (self, state:State, action):  
    if state.board[action]==0:  
        return True  
    return False  
  
def end_of_game (self, state: State):  
    return False  
  
def switch_players (self, state: State):  
    if state.player == 1:  
        state.player = -1  
    else:  
        state.player = 1
```

לא end_of_game הפעולה מומשה.

מימוש end_of_game

```
def end_of_game (self, state: State):
    board = state.board
    row_sum = np.sum(board, axis=1)
    col_sum = np.sum(board, axis=0)
    diagonals = [np.trace(board), np.trace(np.fliplr(board))]
    piece_num = np.count_nonzero(board)

    # print (f'row_sum: {row_sum} col_sum: {col_sum} diagonals: {diagonals}')
    if 3 in row_sum or 3 in col_sum or 3 in diagonals:
        state.end_of_game = 1
        return True
    if -3 in row_sum or -3 in col_sum or -3 in diagonals:
        state.end_of_game = -1
        return True
    if piece_num == 9:
        state.end_of_game = 2
        return True
    return False
```




בדיקת סוף משחק בלולאה הראשית

```
while (run):  
    events = pygame.event.get()  
    for event in events:  
        if event.type == pygame.QUIT:  
            run = False  
    action = player(events)  
    if action:  
        env.move(action)  
        switch_players(player)  
        if env.state.end_of_game != 0:  
            run = False  
    graphics(env.state)  
    clock.tick(FPS)  
  
pygame.time.wait(2000)
```



הוספת כיתוב וסוף משחק

```
def draw (self, state : State):  
    self.header_surf.fill(CADETBBLUE1)  
    self.main_surf.fill(LIGHTGRAY)  
  
    self.draw_Lines()  
    if state.end_of_game == 1:  
        self.write('Player X win')  
    elif state.end_of_game == -1:  
        self.write('Player O win')  
    elif state.end_of_game == 2:  
        self.write('Tie')  
    else:  
        if state.player == 1:  
            self.write('Player X')  
        else:  
            self.write('Player O')  
    self.draw_pieces(state)  
    self.screen.blit(self.header_surf, (0,0))  
    self.screen.blit(self.main_surf, (0,100))  
    pygame.display.update()
```

תרגיל

- הוסף סוכן רנדומלי Random_Agent אשר בוחר בכל תור פעולה רנדומלית חוקית.
- יש להוסיף פעולה לסביבה get_legal_actions המקבלת state ומחזירה רשימה של כל הפעולות החוקיות.
- מומלץ להשתמש בפעולות numpy.where, ופעולה zip.
- [how-to-find-the-index-of-a-value-in-2d-array-in-python](#)
- הסוכן יבחר בצורה רנדומלית מרשימה זו פעולה אחת אותה הוא יחזיר.
- יש להוסיף סוכן רנדומלי למשחק כך שסוכן אדם ישחק מול סוכן רנדומלי.