**UNIWERSYTET EKONOMICZNY   
W KATOWICACH**

SZTUCZNA INTELIGENCJA W GRACH

Przebieg gry Snake z wykorzystaniem języka Python

Marcin Satława

Informatyka II stopień I rok

I grupa

Studia niestacjonarne

Opis projektu

Gra komputerowa Snake wydana w roku 1976 polegająca na kontrolowaniu węża przez gracza, które porusza się po planszy zbierając jedzenie, starając się uniknąć uderzenia głową o ścianę oraz ugryzienia własnego ciała. Celem gry jest zjedzenie przez węża jak największej ilości jedzenia, dzięki której rozmiar węża zwiększa się. Gracz kontroluje kierunek ruchu węża za pomocą klawiszy strzałek (góra, dół, lewo, prawo). Gracz nie może zatrzymać węża, gdy gra jest w toku.

W grze wykorzystano bibliotekę tkinter oraz random. Wymiary gry są następujące: szerokość 600 px, wysokość 400 px. Wąż występuje w kolorze zielony a jedzenie występuje w kolorze czerwonym.

W przypadku ugryzienia lub uderzenia w ścianę przez węża zostaje pokazany komunikat ‘’Przegrałeś’’, jednakże z poziomu gry można uruchomić ponownie grę.

Kod do gry

import tkinter as tk

import random as rn

def main():

predkosc = 100 # szybkosc węża

okno = tk.Tk() # tworzy okno gry

okno.title("Snake") # tytuł gry

c\_draw = tk.Canvas(okno, width = 600, height = 400) # tworzenia canvasa

c\_draw.pack() # paczka canvas

class Snake: # klasa snake

def \_\_init\_\_(self, width, height):

self.width = int(width) # szerokosc

self.height = int(height) # wysokosc

self.msnake = [[self.width / 2, self.height / 2], [self.width / 2 + 1, self.height / 2], [self.width / 2 + 2, self.height / 2]] # elementy weza

self.move = 0 # indeks opisujacy kierunek weza

self.tmove = [[0,1],[1,0],[0,-1],[-1,0]] # indeks okreslajacy ruch weza

self.size = 10

self.col = False # określa, czy doszło do kolizji

self.food = [rn.randint(1, self.width - 2), rn.randint(1, self.height - 2)]

def drawBox(self,x, y, color = 'green'):

c\_draw.create\_rectangle([x, y, x + self.size, y + self.size], fill=color)

def draw(self):

c\_draw.delete("all")

if self.col:

c\_draw.create\_text([self.width / 2 \* self.size, self.height / 2 \* self.size], text = "Przegrałeś")

else:

for i in range(self.width):

self.drawBox(i \* self.size, 0)

self.drawBox(i \* self.size, (self.height - 1) \* self.size)

for i in range(1, self.height):

self.drawBox(0, i \* self.size)

self.drawBox((self.width - 1) \* self.size, i \* self.size)

for i in self.msnake:

self.drawBox(i[0] \* self.size, i[1] \* self.size)

self.drawBox(self.food[0] \* self.size, self.food[1] \* self.size, color = 'red')

def eat(self): # waz musi zjesc

if self.msnake[0][0] == self.food[0] and self.msnake[0][1] == self.food[1]: # warunek, gdy waz jest na jedzeniu to je

self.msnake.append([0,0]) # element powiekszajacy weza

self.food = [rn.randint(1, self.width - 2), rn.randint(1, self.height - 2)] # nowe losowe jedzenie dla weza

def move\_snake(self):

for i in range(len(self.msnake) - 1,0,-1):

self.msnake[i][0] = self.msnake[i-1][0]

self.msnake[i][1] = self.msnake[i-1][1]

self.msnake[0][0] += self.tmove[self.move][0]

self.msnake[0][1] += self.tmove[self.move][1]

self.colision()

self.eat()

self.draw()

def turnLeft(self):

self.move = (self.move + 1) % len(self.tmove)

def turnRight(self):

self.move = (self.move - 1) if self.move > 0 else len(self.tmove) - 1

def colision(self):

if self.msnake[0][0] == 0 or self.msnake[0][1] == 0 or self.msnake[0][0] == self.width - 1 or self.msnake[0][1] == self.height - 1:

self.col = True

for i in self.msnake[1:]: # colision snake - snake

if self.msnake[0][0] == i[0] and self.msnake[0][1] == i[1]:

self.col = True

def reset(self):

self.col = False

self.msnake = [[self.width / 2, self.height / 2], [self.width / 2 + 1, self.height / 2], [self.width / 2 + 2, self.height / 2]] #elementy weza

self.move = 0 # indeks opisujacy kierunek weza

self.tmove = [[0,1],[1,0],[0,-1],[-1,0]] # indeks okreslajacy ruch weza

self.size = 10

self.col = False # określa, czy doszło do kolizji

self.food = [rn.randint(1, self.width - 2), rn.randint(1, self.height - 2)]

okno.after(predkosc, move)

sn = Snake(600 / 10, 400 / 10)

menubar = tk.Menu(okno)

menu = tk.Menu(menubar, tearoff=0)

menubar.add\_cascade(label="Program", menu=menu)

menu.add\_command(label="Od nowa", command = sn.reset)

menu.add\_command(label="Wyjście", command = okno.quit)

okno.config(menu=menubar)

sn.draw()

def move():

sn.move\_snake()

if not sn.col:

okno.after(predkosc, move)

def turnLeft(event):

sn.turnLeft()

def turnRight(event):

sn.turnRight()

okno.after(predkosc, move)

okno.bind\_all("<KeyPress-Left>", turnLeft)

okno.bind\_all("<KeyPress-Right>", turnRight)

okno.mainloop()

return 0

if \_\_name\_\_ == '\_\_main\_\_':

main()

Zdjęcia gry





