Евменов Владимир

ИУ5-32Б

***Домашнее задание***

*Задание: написать программу для реализации игры тетрис на языке* ***Python***

Код программы:

import pygame as pg

import random, time, sys

from pygame.locals import \*

fps = 25

window\_w, window\_h = 600, 500

block, cup\_h, cup\_w = 20, 20, 10

side\_freq, down\_freq = 0.15, 0.1 # передвижение в сторону и вниз

side\_margin = int((window\_w - cup\_w \* block) / 2)

top\_margin = window\_h - (cup\_h \* block) - 5

colors = ((0, 0, 225), (0, 225, 0), (225, 0, 0), (225, 225, 0)) # синий, зеленый, красный, желтый

lightcolors = ((30, 30, 255), (50, 255, 50), (255, 30, 30), (255, 255, 30)) # светло-синий, светло-зеленый, светло-красный, светло-желтый

white, gray, black = (255, 255, 255), (185, 185, 185), (0, 0, 0)

brd\_color, bg\_color, txt\_color, title\_color, info\_color = white, black, white, colors[3], colors[0]

fig\_w, fig\_h = 5, 5

empty = 'o'

figures = {'S': [['ooooo',

'ooooo',

'ooxxo',

'oxxoo',

'ooooo'],

['ooooo',

'ooxoo',

'ooxxo',

'oooxo',

'ooooo']],

'Z': [['ooooo',

'ooooo',

'oxxoo',

'ooxxo',

'ooooo'],

['ooooo',

'ooxoo',

'oxxoo',

'oxooo',

'ooooo']],

'J': [['ooooo',

'oxooo',

'oxxxo',

'ooooo',

'ooooo'],

['ooooo',

'ooxxo',

'ooxoo',

'ooxoo',

'ooooo'],

['ooooo',

'ooooo',

'oxxxo',

'oooxo',

'ooooo'],

['ooooo',

'ooxoo',

'ooxoo',

'oxxoo',

'ooooo']],

'L': [['ooooo',

'oooxo',

'oxxxo',

'ooooo',

'ooooo'],

['ooooo',

'ooxoo',

'ooxoo',

'ooxxo',

'ooooo'],

['ooooo',

'ooooo',

'oxxxo',

'oxooo',

'ooooo'],

['ooooo',

'oxxoo',

'ooxoo',

'ooxoo',

'ooooo']],

'I': [['ooxoo',

'ooxoo',

'ooxoo',

'ooxoo',

'ooooo'],

['ooooo',

'ooooo',

'xxxxo',

'ooooo',

'ooooo']],

'O': [['ooooo',

'ooooo',

'oxxoo',

'oxxoo',

'ooooo']],

'T': [['ooooo',

'ooxoo',

'oxxxo',

'ooooo',

'ooooo'],

['ooooo',

'ooxoo',

'ooxxo',

'ooxoo',

'ooooo'],

['ooooo',

'ooooo',

'oxxxo',

'ooxoo',

'ooooo'],

['ooooo',

'ooxoo',

'oxxoo',

'ooxoo',

'ooooo']]}

def pauseScreen():

pause = pg.Surface((600, 500), pg.SRCALPHA)

pause.fill((0, 0, 255, 127))

display\_surf.blit(pause, (0, 0))

def main():

global fps\_clock, display\_surf, basic\_font, big\_font

pg.init()

fps\_clock = pg.time.Clock()

display\_surf = pg.display.set\_mode((window\_w, window\_h))

basic\_font = pg.font.SysFont('arial', 20)

big\_font = pg.font.SysFont('verdana', 45)

pg.display.set\_caption('Тетрис Lite')

showText('Тетрис Lite')

while True: # начинаем игру

runTetris()

pauseScreen()

showText('Игра закончена')

def runTetris():

cup = emptycup()

last\_move\_down = time.time()

last\_side\_move = time.time()

last\_fall = time.time()

going\_down = False

going\_left = False

going\_right = False

points = 0

level, fall\_speed = calcSpeed(points)

fallingFig = getNewFig()

nextFig = getNewFig()

while True:

if fallingFig == None:

# если нет падающих фигур, генерируем новую

fallingFig = nextFig

nextFig = getNewFig()

last\_fall = time.time()

if not checkPos(cup, fallingFig):

return # если на игровом поле нет свободного места - игра закончена

quitGame()

for event in pg.event.get():

if event.type == KEYUP:

if event.key == K\_SPACE:

pauseScreen()

showText('Пауза')

last\_fall = time.time()

last\_move\_down = time.time()

last\_side\_move = time.time()

elif event.key == K\_LEFT:

going\_left = False

elif event.key == K\_RIGHT:

going\_right = False

elif event.key == K\_DOWN:

going\_down = False

elif event.type == KEYDOWN:

# перемещение фигуры вправо и влево

if event.key == K\_LEFT and checkPos(cup, fallingFig, adjX=-1):

fallingFig['x'] -= 1

going\_left = True

going\_right = False

last\_side\_move = time.time()

elif event.key == K\_RIGHT and checkPos(cup, fallingFig, adjX=1):

fallingFig['x'] += 1

going\_right = True

going\_left = False

last\_side\_move = time.time()

# поворачиваем фигуру, если есть место

elif event.key == K\_UP:

fallingFig['rotation'] = (fallingFig['rotation'] + 1) % len(figures[fallingFig['shape']])

if not checkPos(cup, fallingFig):

fallingFig['rotation'] = (fallingFig['rotation'] - 1) % len(figures[fallingFig['shape']])

# ускоряем падение фигуры

elif event.key == K\_DOWN:

going\_down = True

if checkPos(cup, fallingFig, adjY=1):

fallingFig['y'] += 1

last\_move\_down = time.time()

# мгновенный сброс вниз

elif event.key == K\_RETURN:

going\_down = False

going\_left = False

going\_right = False

for i in range(1, cup\_h):

if not checkPos(cup, fallingFig, adjY=i):

break

fallingFig['y'] += i - 1

# управление падением фигуры при удержании клавиш

if (going\_left or going\_right) and time.time() - last\_side\_move > side\_freq:

if going\_left and checkPos(cup, fallingFig, adjX=-1):

fallingFig['x'] -= 1

elif going\_right and checkPos(cup, fallingFig, adjX=1):

fallingFig['x'] += 1

last\_side\_move = time.time()

if going\_down and time.time() - last\_move\_down > down\_freq and checkPos(cup, fallingFig, adjY=1):

fallingFig['y'] += 1

last\_move\_down = time.time()

if time.time() - last\_fall > fall\_speed: # свободное падение фигуры

if not checkPos(cup, fallingFig, adjY=1): # проверка "приземления" фигуры

addToCup(cup, fallingFig) # фигура приземлилась, добавляем ее в содержимое стакана

points += clearCompleted(cup)

level, fall\_speed = calcSpeed(points)

fallingFig = None

else: # фигура пока не приземлилась, продолжаем движение вниз

fallingFig['y'] += 1

last\_fall = time.time()

# рисуем окно игры со всеми надписями

display\_surf.fill(bg\_color)

drawTitle()

gamecup(cup)

drawInfo(points, level)

drawnextFig(nextFig)

if fallingFig != None:

drawFig(fallingFig)

pg.display.update()

fps\_clock.tick(fps)

def txtObjects(text, font, color):

surf = font.render(text, True, color)

return surf, surf.get\_rect()

def stopGame():

pg.quit()

sys.exit()

def checkKeys():

quitGame()

for event in pg.event.get([KEYDOWN, KEYUP]):

if event.type == KEYDOWN:

continue

return event.key

return None

def showText(text):

titleSurf, titleRect = txtObjects(text, big\_font, title\_color)

titleRect.center = (int(window\_w / 2) - 3, int(window\_h / 2) - 3)

display\_surf.blit(titleSurf, titleRect)

pressKeySurf, pressKeyRect = txtObjects('Нажмите любую клавишу для продолжения', basic\_font, title\_color)

pressKeyRect.center = (int(window\_w / 2), int(window\_h / 2) + 100)

display\_surf.blit(pressKeySurf, pressKeyRect)

while checkKeys() == None:

pg.display.update()

fps\_clock.tick()

def quitGame():

for event in pg.event.get(QUIT): # проверка всех событий, приводящих к выходу из игры

stopGame()

for event in pg.event.get(KEYUP):

if event.key == K\_ESCAPE:

stopGame()

pg.event.post(event)

def calcSpeed(points):

# вычисляет уровень

level = int(points / 10) + 1

fall\_speed = 0.27 - (level \* 0.02)

return level, fall\_speed

def getNewFig():

# возвращает новую фигуру со случайным цветом и углом поворота

shape = random.choice(list(figures.keys()))

newFigure = {'shape': shape,

'rotation': random.randint(0, len(figures[shape]) - 1),

'x': int(cup\_w / 2) - int(fig\_w / 2),

'y': -2,

'color': random.randint(0, len(colors)-1)}

return newFigure

def addToCup(cup, fig):

for x in range(fig\_w):

for y in range(fig\_h):

if figures[fig['shape']][fig['rotation']][y][x] != empty:

cup[x + fig['x']][y + fig['y']] = fig['color']

def emptycup():

# создает пустой стакан

cup = []

for i in range(cup\_w):

cup.append([empty] \* cup\_h)

return cup

def incup(x, y):

return x >= 0 and x < cup\_w and y < cup\_h

def checkPos(cup, fig, adjX=0, adjY=0):

# проверяет, находится ли фигура в границах стакана, не сталкиваясь с другими фигурами

for x in range(fig\_w):

for y in range(fig\_h):

abovecup = y + fig['y'] + adjY < 0

if abovecup or figures[fig['shape']][fig['rotation']][y][x] == empty:

continue

if not incup(x + fig['x'] + adjX, y + fig['y'] + adjY):

return False

if cup[x + fig['x'] + adjX][y + fig['y'] + adjY] != empty:

return False

return True

def isCompleted(cup, y):

# проверяем наличие полностью заполненных рядов

for x in range(cup\_w):

if cup[x][y] == empty:

return False

return True

def clearCompleted(cup):

# Удаление заполенных рядов и сдвиг верхних рядов вниз

removed\_lines = 0

y = cup\_h - 1

while y >= 0:

if isCompleted(cup, y):

for pushDownY in range(y, 0, -1):

for x in range(cup\_w):

cup[x][pushDownY] = cup[x][pushDownY-1]

for x in range(cup\_w):

cup[x][0] = empty

removed\_lines += 1

else:

y -= 1

return removed\_lines

def convertCoords(block\_x, block\_y):

return (side\_margin + (block\_x \* block)), (top\_margin + (block\_y \* block))

def drawBlock(block\_x, block\_y, color, pixelx=None, pixely=None):

#отрисовка квадратных блоков, из которых состоят фигуры

if color == empty:

return

if pixelx == None and pixely == None:

pixelx, pixely = convertCoords(block\_x, block\_y)

pg.draw.rect(display\_surf, colors[color], (pixelx + 1, pixely + 1, block - 1, block - 1), 0, 3)

pg.draw.rect(display\_surf, lightcolors[color], (pixelx + 1, pixely + 1, block - 4, block - 4), 0, 3)

pg.draw.circle(display\_surf, colors[color], (pixelx + block / 2, pixely + block / 2), 5)

def gamecup(cup):

# граница игрового поля-стакана

pg.draw.rect(display\_surf, brd\_color, (side\_margin - 4, top\_margin - 4, (cup\_w \* block) + 8, (cup\_h \* block) + 8), 5)

# фон игрового поля

pg.draw.rect(display\_surf, bg\_color, (side\_margin, top\_margin, block \* cup\_w, block \* cup\_h))

for x in range(cup\_w):

for y in range(cup\_h):

drawBlock(x, y, cup[x][y])

def drawTitle():

titleSurf = big\_font.render('Тетрис Lite', True, title\_color)

titleRect = titleSurf.get\_rect()

titleRect.topleft = (window\_w - 425, 30)

display\_surf.blit(titleSurf, titleRect)

def drawInfo(points, level):

pointsSurf = basic\_font.render(f'Баллы: {points}', True, txt\_color)

pointsRect = pointsSurf.get\_rect()

pointsRect.topleft = (window\_w - 550, 180)

display\_surf.blit(pointsSurf, pointsRect)

levelSurf = basic\_font.render(f'Уровень: {level}', True, txt\_color)

levelRect = levelSurf.get\_rect()

levelRect.topleft = (window\_w - 550, 250)

display\_surf.blit(levelSurf, levelRect)

pausebSurf = basic\_font.render('Пауза: пробел', True, info\_color)

pausebRect = pausebSurf.get\_rect()

pausebRect.topleft = (window\_w - 550, 420)

display\_surf.blit(pausebSurf, pausebRect)

escbSurf = basic\_font.render('Выход: Esc', True, info\_color)

escbRect = escbSurf.get\_rect()

escbRect.topleft = (window\_w - 550, 450)

display\_surf.blit(escbSurf, escbRect)

def drawFig(fig, pixelx=None, pixely=None):

figToDraw = figures[fig['shape']][fig['rotation']]

if pixelx == None and pixely == None:

pixelx, pixely = convertCoords(fig['x'], fig['y'])

#отрисовка элементов фигур

for x in range(fig\_w):

for y in range(fig\_h):

if figToDraw[y][x] != empty:

drawBlock(None, None, fig['color'], pixelx + (x \* block), pixely + (y \* block))

def drawnextFig(fig): # превью следующей фигуры

nextSurf = basic\_font.render('Следующая:', True, txt\_color)

nextRect = nextSurf.get\_rect()

nextRect.topleft = (window\_w - 150, 180)

display\_surf.blit(nextSurf, nextRect)

drawFig(fig, pixelx=window\_w-150, pixely=230)

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

main()

