Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашней работе

Выполнил:

студент группы

ИУ5-32Б

Табахов Евгений

Подпись и дата:

_____

_____

Проверил:

преподаватель каф.

ИУ5

Гапанюк Ю.Е.

Подпись и дата:

_____

_____

Москва, 2023

Реализация игры «Настольный теннис» на Python

```
from tkinter import *
# импортируем библиотеку random
import random

# Добавляем глобальные переменные

# глобальные переменные
# настройки окна
WIDTH = 900
HEIGHT = 300

# настройки ракеток

# ширина ракетки
PAD_W = 10
# высота ракетки
PAD_H = 100

# настройки мяча
# Насколько будет увеличиваться скорость мяча с каждым ударом
BALL_SPEED_UP = 1.05
# Максимальная скорость мяча
BALL_MAX_SPEED = 40
# радиус мяча
BALL_RADIUS = 30

INITIAL_SPEED = 20
BALL_X_SPEED = INITIAL_SPEED
BALL_Y_SPEED = INITIAL_SPEED

# Счет игроков
PLAYER_1_SCORE = 0
PLAYER_2_SCORE = 0

# Добавим глобальную переменную отвечающую за расстояние
# до правого края игрового поля
right_line_distance = WIDTH - PAD_W

def update_score(player):
    global PLAYER_1_SCORE, PLAYER_2_SCORE
    if player == "right":
        PLAYER_1_SCORE += 1
        c.itemconfig(p_1_text, text=PLAYER_1_SCORE)
    else:
        PLAYER_2_SCORE += 1
        c.itemconfig(p_2_text, text=PLAYER_2_SCORE)

def spawn_ball():
    global BALL_X_SPEED
    # Выставляем мяч по центру
    c.coords(BALL, WIDTH/2-BALL_RADIUS/2,
             HEIGHT/2-BALL_RADIUS/2,
             WIDTH/2+BALL_RADIUS/2,
             HEIGHT/2+BALL_RADIUS/2)
    # Задаем мячу направление в сторону проигравшего игрока,
    # но снижаем скорость до изначальной
    BALL_X_SPEED = -(BALL_X_SPEED * -INITIAL_SPEED) / abs(BALL_X_SPEED)

# функция отскока мяча
def bounce(action):
    global BALL_X_SPEED, BALL_Y_SPEED
    # ударили ракеткой
    if action == "strike":
        BALL_Y_SPEED = random.randrange(-10, 10)
```

```
        if abs(BALL_X_SPEED) < BALL_MAX_SPEED:
            BALL_X_SPEED *= -BALL_SPEED_UP
        else:
            BALL_X_SPEED = -BALL_X_SPEED
    else:
        BALL_Y_SPEED = -BALL_Y_SPEED

# устанавливаем окно
root = Tk()
root.title("PythonicWay Pong")

# область анимации
c = Canvas(root, width=WIDTH, height=HEIGHT, background="#003300")
c.pack()

# элементы игрового поля

# левая линия
c.create_line(PAD_W, 0, PAD_W, HEIGHT, fill="white")
# правая линия
c.create_line(WIDTH-PAD_W, 0, WIDTH-PAD_W, HEIGHT, fill="white")
# вертикальная центральная линия
c.create_line(WIDTH/2, 0, WIDTH/2, HEIGHT, fill="white")

# установка игровых объектов

# создаем мяч
BALL = c.create_oval(WIDTH/2-BALL_RADIUS/2,
                     HEIGHT/2-BALL_RADIUS/2,
                     WIDTH/2+BALL_RADIUS/2,
                     HEIGHT/2+BALL_RADIUS/2, fill="white")

# левая ракетка
LEFT_PAD = c.create_line(PAD_W/2, 0, PAD_W/2, PAD_H, width=PAD_W, fill="yellow")

# правая ракетка
RIGHT_PAD = c.create_line(WIDTH-PAD_W/2, 0, WIDTH-PAD_W/2,
                          PAD_H, width=PAD_W, fill="yellow")


p_1_text = c.create_text(WIDTH-WIDTH/6, PAD_H/4,
                         text=PLAYER_1_SCORE,
                         font="Arial 20",
                         fill="white")

p_2_text = c.create_text(WIDTH/6, PAD_H/4,
                         text=PLAYER_2_SCORE,
                         font="Arial 20",
                         fill="white")

# добавим глобальные переменные для скорости движения мяча
# по горизонтали
BALL_X_CHANGE = 20
# по вертикали
BALL_Y_CHANGE = 0

def move_ball():
    # определяем координаты сторон мяча и его центра
    ball_left, ball_top, ball_right, ball_bot = c.coords(BALL)
    ball_center = (ball_top + ball_bot) / 2

    # вертикальный отскок
    # Если мы далеко от вертикальных линий - просто двигаем мяч
    if ball_right + BALL_X_SPEED < right_line_distance and \
            ball_left + BALL_X_SPEED > PAD_W:
        c.move(BALL, BALL_X_SPEED, BALL_Y_SPEED)
```

```
    # Если мяч касается своей правой или левой стороной границы поля
    elif ball_right == right_line_distance or ball_left == PAD_W:
        # Проверяем правой или левой стороны мы касаемся
        if ball_right > WIDTH / 2:
            # Если правой, то сравниваем позицию центра мяча
            # с позицией правой ракетки.
            # И если мяч в пределах ракетки делаем отскок
            if c.coords(RIGHT_PAD)[1] < ball_center < c.coords(RIGHT_PAD)[3]:
                bounce("strike")
            else:
                # Иначе игрок пропустил - тут оставим пока pass, его мы заменим на подсчет очков и респаун мячика
                update_score("left")
                spawn_ball()
        else:
            # То же самое для левого игрока
            if c.coords(LEFT_PAD)[1] < ball_center < c.coords(LEFT_PAD)[3]:
                bounce("strike")
            else:
                update_score("right")
                spawn_ball()
    # Проверка ситуации, в которой мячик может вылететь за границы игрового поля.
    # В таком случае просто двигаем его к границе поля.
    else:
        if ball_right > WIDTH / 2:
            c.move(BALL, right_line_distance-ball_right, BALL_Y_SPEED)
        else:
            c.move(BALL, -ball_left+PAD_W, BALL_Y_SPEED)
    # горизонтальный отскок
    if ball_top + BALL_Y_SPEED < 0 or ball_bot + BALL_Y_SPEED > HEIGHT:
        bounce("ricochet")

# зададим глобальные переменные скорости движения ракеток
# скорось с которой будут ездить ракетки
PAD_SPEED = 20
# скорость левой платформы
LEFT_PAD_SPEED = 0
# скорость правой ракетки
RIGHT_PAD_SPEED = 0

# функция движения обеих ракеток
def move_pads():
    # для удобства создадим словарь, где ракетке соответствует ее скорость
    PADS = {LEFT_PAD: LEFT_PAD_SPEED,
            RIGHT_PAD: RIGHT_PAD_SPEED}
    # перебираем ракетки
    for pad in PADS:
        # двигаем ракетку с заданной скоростью
        c.move(pad, 0, PADS[pad])
        # если ракетка вылезает за игровое поле возвращаем ее на место
        if c.coords(pad)[1] < 0:
            c.move(pad, 0, -c.coords(pad)[1])
        elif c.coords(pad)[3] > HEIGHT:
            c.move(pad, 0, HEIGHT - c.coords(pad)[3])


def main():
    move_ball()
    move_pads()
    # вызываем саму себя каждые 30 миллисекунд
    root.after(30, main)

# Установим фокус на Canvas чтобы он реагировал на нажатия клавиш
c.focus_set()

# Напишем функцию обработки нажатия клавиш
def movement_handler(event):
```

```
    global LEFT_PAD_SPEED, RIGHT_PAD_SPEED
    if event.keysym == "w":
        LEFT_PAD_SPEED = -PAD_SPEED
    elif event.keysym == "s":
        LEFT_PAD_SPEED = PAD_SPEED
    elif event.keysym == "Up":
        RIGHT_PAD_SPEED = -PAD_SPEED
    elif event.keysym == "Down":
        RIGHT_PAD_SPEED = PAD_SPEED

# Привяжем к Canvas эту функцию
c.bind("<KeyPress>", movement_handler)

# Создадим функцию реагирования на отпускание клавиши
def stop_pad(event):
    global LEFT_PAD_SPEED, RIGHT_PAD_SPEED
    if event.keysym in "ws":
        LEFT_PAD_SPEED = 0
    elif event.keysym in ("Up", "Down"):
        RIGHT_PAD_SPEED = 0

# Привяжем к Canvas эту функцию
c.bind("<KeyRelease>", stop_pad)

# запускаем движение
main()

# запускаем работу окна
root.mainloop()
```

Результат работы программы