Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Современные языки программирования

ЛАБОРАТОРНАЯ РАБОТА № 2
ООР

Выполнил:

Атаев И.М. гр. 910101

Проверила:

Василькова А.Н.

Минск 2022

Задание: Игра «Змейка»

- змейка (упорядоченный набор связанных звеньев с явно выделенными концами – головой
- и хвостом) передвигается по полю «N» х «M»;
- в начале игры змейка состоит из одного звена;
- перемещение змейки состоит в добавлении одного звена к ее голове в требуемом
- направлении (в направлении ее движения) и удалении одного звена хвоста;
- если при перемещении змейки ее голова натыкается на препятствие, то игра проиграна;
- в каждый момент времени на игровом поле находится «T» элементов еды, занимающей
- одну клетку поля;
- если при перемещении голова змейки натыкается на еду, то змейка ее «съедает» и вырастает на одно звено, а для выполнения предыдущего правила на поле в
- произвольном свободном месте автоматически появляется новая порция еды;
- выигрыш состоит в достижении змейкой длины в «L» звена.
- Пример возможного графического интерфейса программы: на поле есть несколько таких же змеек управляемых программой, обладающих разной стратегией поведения; в случае, если змейка пользователя пересекает другую змейку, то хвост другой змейки
- «отгрызается». Если это голова, то змейка съедается полностью (использовать Tkinter или PyGame (самостоятельно), предупреди что большинство реализаций из интернета нам известны

Исходный код:

```
import pyglet
import pyglet.gl
from pyglet.window import mouse, key
import random
import time
from include import Snake, Block


class MyWindow(pyglet.window.Window):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        init_pos_x = snake_vel * 20
        init pos y = snake vel * 20

        self.food = Block(init_pos_x, init_pos_y, snake_vel,
                          color=[1., 0., 0., .6] * 4)
        self.snake = Snake(snake_vel, 0, snake_vel)

        # Board settings
        self.n_squares_x = self.width // snake_vel
        self.n_squares_y = self.height // snake_vel
```

```
        # Display a counter
        self.counter = 0
        self.best_score = 0
        self.label = pyglet.text.Label(
            '',
            font_name='Times New Roman',
            font_size=12,
            bold=True,
            x=80,
            y=self.height - 30,
            width=100,
            height=40,
            anchor_x='center',
            anchor_y='center',
            color=(255, 255, 255, 100),
            multiline=True
        )

        self.game_over = False

    def on_draw(self):
        # Window features ######################################
        pyglet.gl.glEnable(pyglet.gl.GL_BLEND)
        pyglet.gl.glBlendFunc(pyglet.gl.GL_SRC_ALPHA,
                              pyglet.gl.GL_ONE_MINUS_SRC_ALPHA)
        pyglet.gl.glClear(pyglet.gl.GL_COLOR_BUFFER_BIT)
        ########################################################

        # --- Draw the food ------------------------------
        if self.snake.eaten is True:
            # if snake has eaten, draw a new food
            new_x = snake_vel * \
                random.randint(2, self.n_squares_x - 2)
            new_y = snake_vel * \
                random.randint(2, self.n_squares_y - 2)
            self.food.pos_x, self.food.pos_y = new_x, new_y

            self.food.set_vertex()
            self.snake.eaten = False

        food_vertices = pyglet.graphics.vertex_list(
            4,
            ('v2f', self.food.vertex),
            ('c4f', self.food.color)
        )
        food_vertices.draw(pyglet.gl.GL_POLYGON)

        # --- Draw the snake -----------------------------
        for block in self.snake.blocks:
            # Check the boundaries
            if block.pos_x < 0:
                block.pos_x += self.width
            elif block.pos_x > self.width - snake_vel:
                block.pos_x = 0
                # block.pos_x -= self.width + self.snake.size

            if block.pos_y < 0:
                block.pos_y += self.height
            elif block.pos_y > self.height - snake_vel:
                block.pos_y = 0
                # block.pos_y -= self.height + self.snake.size

            block.set_vertex()

            snake_vertices = pyglet.graphics.vertex_list(
```

```
                4,
                ('v2f', block.vertex),
                ('c4f', block.color)
            )

            snake_vertices.draw(pyglet.gl.GL_POLYGON)

        # --- Draw the counter ----------------------------
        text = 'Score:\t{}\nBest:\t{}'.format(self.counter, self.best_score)
        # self.label.text = 'Score: {}\nBest of this game: {}'.format(
        #     self.counter, self.best_score)
        self.label.text = text
        self.label.draw()

    def update(self, dt):
        if self.snake.dead is False:
            # Check if the snake has eaten the food
            head = self.snake.blocks[-1]
            dif_pos_x = abs(head.pos_x - self.food.pos_x) - snake_vel * .5
            dif_pos_y = abs(head.pos_y - self.food.pos_y) - snake_vel * .5
            if dif_pos_x <= 1. and dif_pos_y <= 1.:
                self.snake.eat(self.food.pos_x, self.food.pos_y)
                self.counter += 1
            # Move the snake
            self.snake.move_snake()
        else:
            if self.game_over is False:
                time.sleep(1)
                self.game_over = True
            self.snake.blocks = self.snake.blocks[1:]
            if len(self.snake.blocks) == 0:
                time.sleep(1)
                if self.counter > self.best_score:
                    self.best_score = self.counter
                self.counter = 0

                # Restart game
                self.food = Block(snake_vel * 20, snake_vel *
                                  20, snake_vel, color=[1., 0., 0., .9] * 4)
                self.snake = Snake(snake_vel, 0, snake_vel)
                self.game_over = False

    def mouse(self, x, y):
        pass


if __name__ == '__main__':
    global snake_vel

    snake_vel = 20
    width, height = 800, 600

    world = MyWindow(width, height)
    pyglet.gl.glClearColor(.1, .1, .1, .1)
    world.on_draw()

    @world.event
    def on_mouse_press(x, y, button, modifiers):
        if button == mouse.LEFT:
            world.mouse(x, y)

    @world.event
    def on_key_press(symbol, modifiers):

        if symbol == key.UP or symbol == key.W:
            world.snake.change_vel(0, snake_vel)
```

```
        if symbol == key.DOWN or symbol == key.S:
            world.snake.change_vel(0, -snake_vel)
        if symbol == key.RIGHT or symbol == key.D:
            world.snake.change_vel(snake_vel, 0)
        if symbol == key.LEFT or symbol == key.A:
            world.snake.change_vel(-snake_vel, 0)

    pyglet.clock.schedule_interval(world.update, 1 / 20.)
    pyglet.app.run()
```

```
class Block:
    def __init__(self, pos_x, pos_y, size, color=[0, 1., 0, 1.] * 4):
        self.pos_x = pos_x + size
        self.pos_y = pos_y + size

        self.size = size
        self.vertex = []
        self.color = color

        self.red_size = 3  # Reduce the size on draw a little bit

    def move(self, delta_x, delta_y):
        self.pos_x += delta_x
        self.pos_y += delta_y

    def set_vertex(self):
        self.vertex = []
        # Add the four vertex to the vertex list
        self.vertex.append(self.pos_x + self.red_size)
        self.vertex.append(self.pos_y + self.red_size)

        self.vertex.append(self.pos_x + self.red_size)
        self.vertex.append(self.pos_y + self.size - self.red_size)

        self.vertex.append(self.pos_x + self.size - self.red_size)
        self.vertex.append(self.pos_y + self.size - self.red_size)

        self.vertex.append(self.pos_x + self.size - self.red_size)
        self.vertex.append(self.pos_y + self.red_size)


class Snake:
    def __init__(self, vel_x, vel_y, size):
        # The snake starts with just two blocks
        init_pos = 5 * vel_x
        self.head_color = [.5, 8., .6, 1.] * 4
        self.tail_color = [.0, 1., .0, 1.] * 4
        self.blocks = [
            Block(pos_x=-2 * vel_x + init_pos, pos_y=init_pos, size=size,
                  color=self.tail_color),
            Block(pos_x=-vel_x + init_pos, pos_y=init_pos, size=size,
                  color=self.tail_color),
            Block(pos_x=init_pos, pos_y=init_pos,
                  size=size, color=self.head_color)
        ]

        self.vel_x, self.vel_y = vel_x, vel_y
        self.dead = False
        self.size = size

        self.eaten = True

    def move_snake(self):
```

```
        self.check_block_pos()

        # Move the tail
        for i in range(len(self.blocks) - 1):
            next_block = self.blocks[i + 1]
            dif_x = next_block.pos_x - self.blocks[i].pos_x
            dif_y = next_block.pos_y - self.blocks[i].pos_y

            self.blocks[i].move(dif_x, dif_y)

        # Move the head
        self.blocks[-1].move(self.vel_x, self.vel_y)

        return True

    def change_vel(self, vel_x, vel_y):
        next_head_pos_x = self.blocks[-1].pos_x + vel_x
        next_head_pos_y = self.blocks[-1].pos_y + vel_y

        # Check if we are trying to move backwards
        dif_x = abs(next_head_pos_x - self.blocks[-2].pos_x)
        dif_y = abs(next_head_pos_y - self.blocks[-2].pos_y)
        if dif_x > 1. and dif_y > 1.:
            self.vel_x = vel_x
            self.vel_y = vel_y

    def check_block_pos(self):

        next_head_pos_x = self.blocks[-1].pos_x + self.vel_x
        next_head_pos_y = self.blocks[-1].pos_y + self.vel_y

        for block in self.blocks[:-2:]:
            dif_x = abs(next_head_pos_x - block.pos_x)
            dif_y = abs(next_head_pos_y - block.pos_y)
            if dif_x <= 1. and dif_y <= 1.:
                self.dead = True
                return False

        return True

    def eat(self, x_food, y_food):
        self.blocks.append(
            Block(x_food - self.size, y_food - self.size, self.size))
        self.blocks[-2].color = self.tail_color
        self.blocks[-1].color = self.head_color
        self.eaten = True


def main():
    # snk = Snake()
    pass


if __name__ == '__main__':
    main()
```

Результаты работы программы:

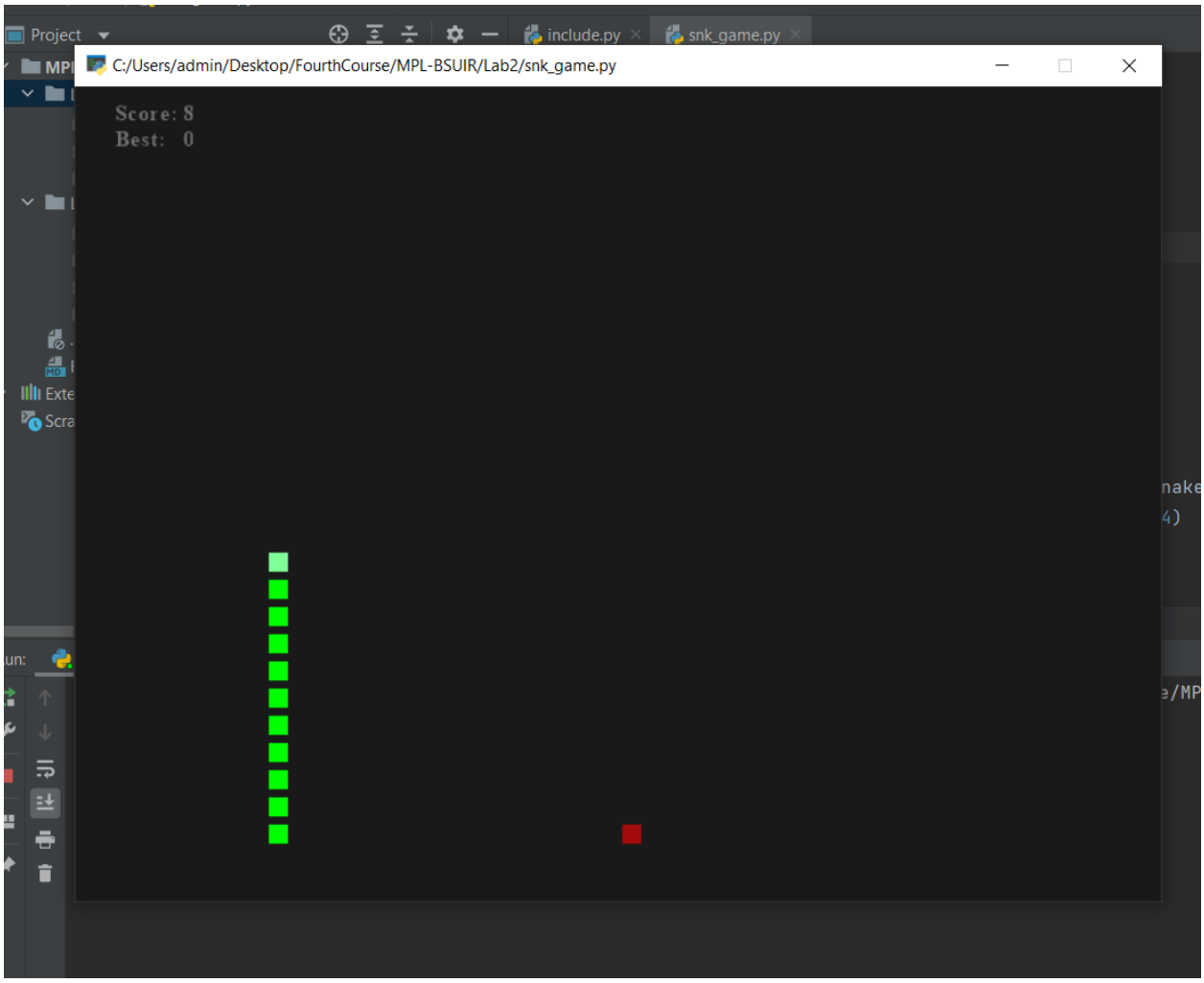

Рис. 1 – Змейка работает.

Вывод: В ходе выполнения лабораторной работы ознакомились с pyglet для визуализации данных, а именно работа с классами. Повторили работу с классами в Python.