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»

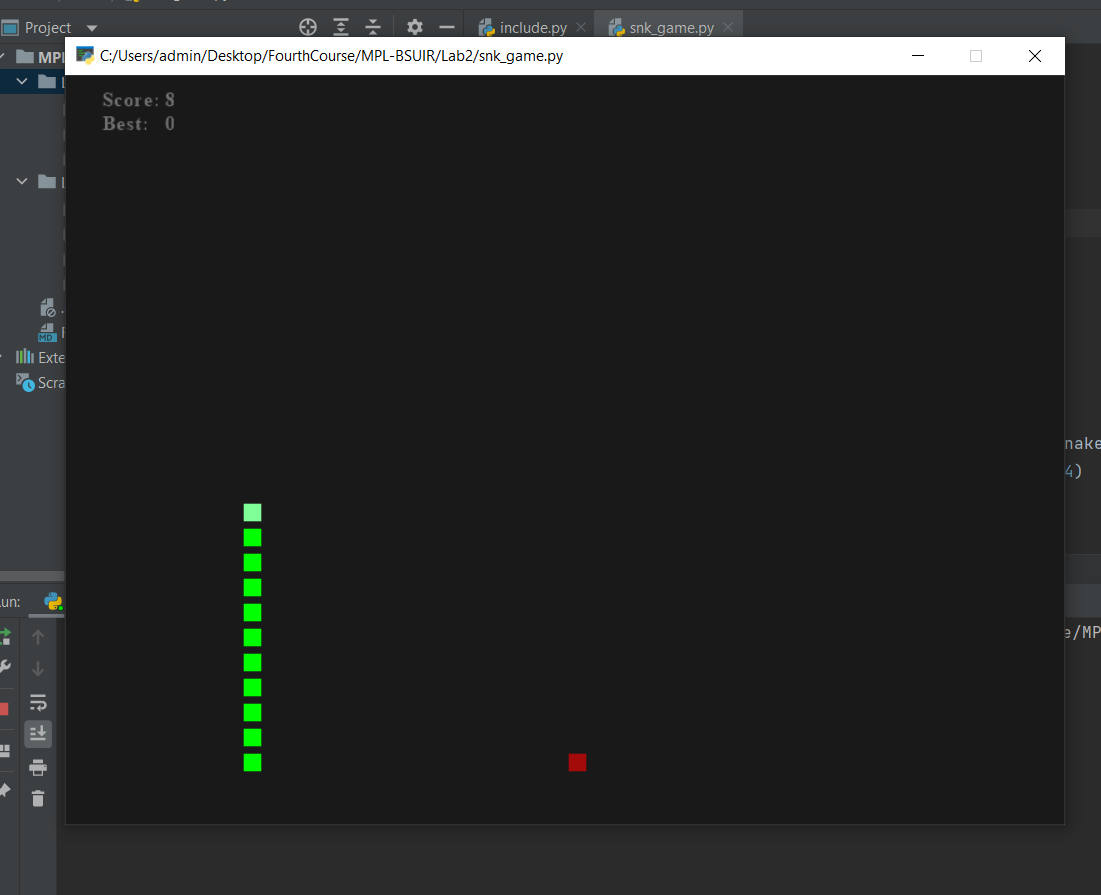
* змейка (упорядоченный набор связанных звеньев с явно выделенными концами – головой
* и хвостом) передвигается по полю «N» x «M»;
* в начале игры змейка состоит из одного звена;
* перемещение змейки состоит в добавлении одного звена к ее голове в требуемом
* направлении (в направлении ее движения) и 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 – Змекйка работает.*

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