**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 ДЗ

Выполнил: Проверил:

студент группы ИУ5-34Б преподаватель каф. ИУ5

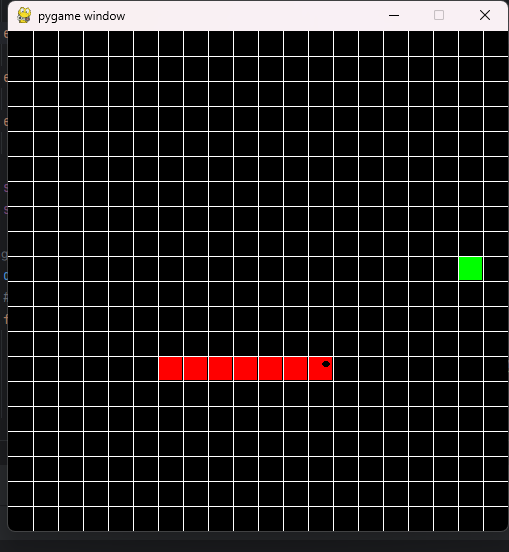
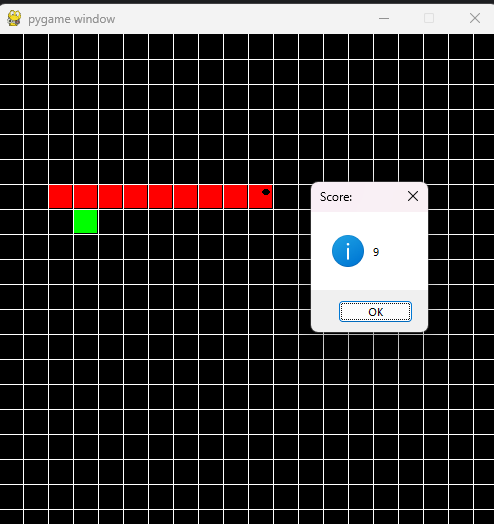
Кожевников М. А. Гапанюк Ю.Е. Подпись и дата: Подпись и дата:

Москва, 2023 г.

Задание:

Реализовать игру с использованием библиотеки pygame

Текст программы: #Управлять змейкой на клавиши: 'w' 'a' 's' 'd'   
import pygame  
import random  
from tkinter import messagebox  
  
width = 500  
height = 500  
  
cols = 25  
rows = 20  
  
  
# Класс Cube представляет собой отдельный кубик на игровом поле  
class Cube():  
 rows = 20  
 w = 500  
  
 def \_\_init\_\_(self, start, dirnx=1, dirny=0, color=(255, 0, 0)):  
 # Инициализация кубика  
 self.pos = start  
 self.dirnx = dirnx  
 self.dirny = dirny # Направление движения ("L", "R", "U", "D")  
 self.color = color  
  
 def move(self, dirnx, dirny):  
 # Обновление позиции кубика в зависимости от направления  
 self.dirnx = dirnx  
 self.dirny = dirny  
 self.pos = (self.pos[0] + self.dirnx, self.pos[1] + self.dirny)  
  
 def draw(self, surface, eyes=False):  
 # Отрисовка кубика на игровом поле  
 dis = self.w // self.rows  
 i = self.pos[0]  
 j = self.pos[1]  
  
 pygame.draw.rect(surface, self.color, (i \* dis + 1, j \* dis + 1, dis - 2, dis - 2))  
 if eyes:  
 # Если указан параметр eyes, рисуем глазки на кубике  
 centre = dis // 2  
 radius = 3  
 eye\_offset = 0  
 if self.dirnx == 1:  
 eye\_offset = 5  
 elif self.dirnx == -1:  
 eye\_offset = -5  
 circleMiddle = (i \* dis + centre + eye\_offset, j \* dis + 8)  
 circleMiddle2 = (i \* dis + dis - radius \* 2, j \* dis + 8)  
 pygame.draw.circle(surface, (0, 0, 0), circleMiddle, radius)  
 pygame.draw.circle(surface, (0, 0, 0), circleMiddle2, radius)  
  
  
# Класс SpecialSnack представляет собой специальную закуску с уникальными свойствами  
class SpecialSnack(Cube):  
 def \_\_init\_\_(self, start, color=(255, 255, 0), special\_property="DoublePoints"):  
 super().\_\_init\_\_(start, color=color)  
 self.special\_property = special\_property  
  
  
# Класс Snake представляет собой змею, состоящую из кубиков  
class Snake():  
 body = []  
 turns = {}  
  
 def \_\_init\_\_(self, color, pos):  
 # Инициализация змеи  
 self.color = color  
 self.head = Cube(pos)  
 self.body.append(self.head)  
 self.dirnx = 0  
 self.dirny = 1  
  
 def move(self):  
 # Обработка событий нажатия клавиш и изменение направления движения  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 keys = pygame.key.get\_pressed()  
  
 for key in keys:  
 if keys[pygame.K\_a]:  
 self.dirnx = -1  
 self.dirny = 0  
 self.turns[self.head.pos[:]] = [self.dirnx, self.dirny]  
 elif keys[pygame.K\_d]:  
 self.dirnx = 1  
 self.dirny = 0  
 self.turns[self.head.pos[:]] = [self.dirnx, self.dirny]  
 elif keys[pygame.K\_w]:  
 self.dirny = -1  
 self.dirnx = 0  
 self.turns[self.head.pos[:]] = [self.dirnx, self.dirny]  
 elif keys[pygame.K\_s]:  
 self.dirny = 1  
 self.dirnx = 0  
 self.turns[self.head.pos[:]] = [self.dirnx, self.dirny]  
  
 for i, c in enumerate(self.body):  
 p = c.pos[:]  
 if p in self.turns:  
 # Если на пути змеи есть поворот, выполняем его  
 turn = self.turns[p]  
 c.move(turn[0], turn[1])  
 if i == len(self.body) - 1:  
 self.turns.pop(p)  
 else:  
 # Иначе продолжаем движение в текущем направлении  
 c.move(c.dirnx, c.dirny)  
  
 def reset(self, pos):  
 # Сброс состояния змеи до начального  
 self.head = Cube(pos)  
 self.body = []  
 self.body.append(self.head)  
 self.turns = {}  
 self.dirnx = 0  
 self.dirny = 1  
  
 def addCube(self):  
 # Добавление нового кубика к змее  
 tail = self.body[-1]  
 dx, dy = tail.dirnx, tail.dirny  
  
  
 if dx == 1 and dy == 0:  
 self.body.append(Cube((tail.pos[0] - 1, tail.pos[1])))  
 elif dx == -1 and dy == 0:  
 self.body.append(Cube((tail.pos[0] + 1, tail.pos[1])))  
 elif dx == 0 and dy == 1:  
 self.body.append(Cube((tail.pos[0], tail.pos[1] - 1)))  
 elif dx == 0 and dy == -1:  
 self.body.append(Cube((tail.pos[0], tail.pos[1] + 1)))  
  
 self.body[-1].dirnx = dx  
 self.body[-1].dirny = dy  
  
 def draw(self, surface):  
 # Отрисовка змеи на игровом поле  
 for i, c in enumerate(self.body):  
 if i == 0:  
 c.draw(surface, True) # Отрисовка головы с глазами  
 else:  
 c.draw(surface)  
  
  
def redrawWindow():  
 # Обновление игрового окна  
 global win  
 win.fill((0, 0, 0))  
 drawGrid(width, rows, win)  
 s.draw(win)  
 snack.draw(win)  
 pygame.display.update()  
  
  
def drawGrid(w, rows, surface):  
 # Отрисовка сетки на игровом поле  
 sizeBtwn = w // rows  
  
 x = 0  
 y = 0  
 for l in range(rows):  
 x = x + sizeBtwn  
 y = y + sizeBtwn  
  
 pygame.draw.line(surface, (255, 255, 255), (x, 0), (x, w))  
 pygame.draw.line(surface, (255, 255, 255), (0, y), (w, y))  
  
  
def randomSnack(rows, item):  
 # Генерация случайного положения закуски на поле  
 positions = item.body  
  
 while True:  
 x = random.randrange(1, rows - 1)  
 y = random.randrange(1, rows - 1)  
 if len(list(filter(lambda z: z.pos == (x, y), positions))) > 0:  
 continue  
 else:  
 break  
  
 return (x, y)  
  
def randomSpecialSnack(rows, item):  
 # Генерация случайного положения специальной закуски на поле  
 positions = item.body  
  
 while True:  
 x = random.randrange(1, rows - 1)  
 y = random.randrange(1, rows - 1)  
 if len(list(filter(lambda z: z.pos == (x, y), positions))) > 0:  
 continue  
 else:  
 break  
  
 return (x, y)  
  
  
def handleSpecialSnack(s, snack):  
 # Обработка действий при съедании закуски  
 if isinstance(snack, SpecialSnack):  
 if snack.special\_property == "DoublePoints":  
 s.addCube()  
 s.addCube()  
 else:  
 # Действия при съедании обычной закуски (по умолчанию)  
 s.addCube()  
  
  
def randomSnackOrSpecial(rows, item):  
 # Генерация случайного типа закуски (обычной или специальной)  
 chance\_special\_snack = 20 # Шанс появления специальной закуски (например, 20%)  
  
 if random.randrange(1, 101) <= chance\_special\_snack:  
 return SpecialSnack(randomSpecialSnack(rows, item), color=(255, 255, 0))  
 else:  
 return Cube(randomSnack(rows, item), color=(0, 255, 0))  
  
def main():  
 # Основная функция игры  
 global s, snack, win  
 win = pygame.display.set\_mode((width, height))  
 s = Snake((255, 0, 0), (10, 10))  
 s.addCube()  
 flag = True  
 clock = pygame.time.Clock()  
  
 # Инициализируем первую закуску перед началом цикла  
 snack = randomSnackOrSpecial(rows, s)  
  
 while flag:  
 pygame.time.delay(50)  
 clock.tick(10)  
 s.move()  
 headPos = s.head.pos  
 if headPos[0] >= 20 or headPos[0] < 0 or headPos[1] >= 20 or headPos[1] < 0:  
 messagebox.showinfo("Score:", len(s.body))  
 s.reset((10, 10))  
  
 # Проверяем, съедена ли текущая закуска  
 if s.body[0].pos == snack.pos:  
 handleSpecialSnack(s, snack)  
 # Создаем новую закуску (обычную или специальную) с заданным шансом  
 snack = randomSnackOrSpecial(rows, s)  
  
 for x in range(len(s.body)):  
 if s.body[x].pos in list(map(lambda z: z.pos, s.body[x + 1:])):  
 messagebox.showinfo("Score:", len(s.body))  
 s.reset((10, 10))  
 break  
  
 redrawWindow()  
  
main()

Результат работы программы: