|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

Кузнецов Егор ИУ5-32Б

Парадигмы и конструкции языков программирования

**ОТЧЁТ ПО**

**Домашнему заданию**

Москва

2023

Задание

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Задача

Задачей является написание простейшей игры змейки на языке Python.

Структура программы

DZ.py – Основной файл, в котором выполнена вся работа.

Текст программы

from tkinter import \*  
from enum import Enum  
import time  
import random  
  
  
class python\_snake:  
 def \_\_init\_\_(self, window, canv\_x, canv\_y, canv\_width, canv\_height):  
 self.\_\_started = 1  
 self.\_\_spped = 10  
 self.\_\_window = window  
 self.\_\_canv\_x = canv\_x  
 self.\_\_canv\_y = canv\_y  
 self.canv\_width = canv\_width  
 self.canv\_height = canv\_height  
 self.\_\_snake\_x = self.canv\_width // 2 # Координата старта змеи  
 self.\_\_snake\_y = self.canv\_height // 2 # Координата старта змеи  
 self.canv = Canvas(self.\_\_window, width=self.canv\_width,  
 height=self.canv\_height,  
 bg=self.CONST.CANVAS\_BGCOLOR.value)  
 self.canv.place(x=self.\_\_canv\_x, y=self.\_\_canv\_y)  
 self.create\_head\_food()  
  
 self.\_\_window.bind('<d>', self.right)  
 self.\_\_window.bind('<D>', self.right)  
 self.\_\_window.bind('<Right>', self.right)  
 self.\_\_window.bind('<s>', self.down)  
 self.\_\_window.bind('<S>', self.down)  
 self.\_\_window.bind('<Down>', self.down)  
 self.\_\_window.bind('<a>', self.left)  
 self.\_\_window.bind('<A>', self.left)  
 self.\_\_window.bind('<Left>', self.left)  
 self.\_\_window.bind('<w>', self.up)  
 self.\_\_window.bind('<W>', self.up)  
 self.\_\_window.bind('<Up>', self.up)  
  
 self.\_\_window.bind('<e>', self.move)  
 self.\_\_window.bind('<q>', self.quit)  
 self.\_\_window.bind('<Destroy>', self.quit)  
 self.\_\_window.bind('<plus>', self.speed\_key)  
 self.\_\_window.bind('<minus>', self.speed\_key)  
 self.\_\_window.bind('<KP\_Add>', self.speed\_key) # Клавиша + на боковой клаве  
 self.\_\_window.bind('<KP\_Subtract>', self.speed\_key) # Клавиша - на боковой клаве  
  
 class CONST(Enum): # Список возможных направлений движения и других констант  
 RIGHT = 1  
 DOWN = 2  
 LEFT = 3  
 UP = 4  
 SNAKE\_HCOLOR = 'red' # Цвет головы змейки  
 SNAKE\_BCOLOR = 'green' # Цвет тела змейки  
 CANVAS\_BGCOLOR = '#f75394' # Цвет фона холста  
 SNAKE\_THICKNESS = 11 # Толщина тела змейки (нечётное число)  
 FOOD\_THICKNESS = 15 # Толщина еды (нечётное число)  
 FOOD\_COLOR = '#aced95' # Цвет тела еды  
 EXPLOSIVE = 15 # Диаметр взрыва при столкновении змеи с препятствием (нечётное число)  
 EXPLOSIVE\_BORD = 10 # толщина контура взрыва при столкновении змеи с препятствием  
 EXPLOSIVE\_BCOLOR = '#ff9999' # Цвет тела взрыва  
 EXPLOSIVE\_CCOLOR = '#881a1a' # Цвет контура взрыва  
  
 # обработчики клавиш изменения направления движения:  
 def right(self, event):  
 self.\_\_vector = self.CONST.RIGHT.value  
  
 def down(self, event):  
 self.\_\_vector = self.CONST.DOWN.value  
  
 def left(self, event):  
 self.\_\_vector = self.CONST.LEFT.value  
  
 def up(self, event):  
 self.\_\_vector = self.CONST.UP.value  
  
 def speed\_key(self, event):  
 # print(event.keysym)  
 if event.keysym == 'KP\_Add' or event.keysym == 'plus':  
 self.speed('+')  
 elif event.keysym == 'KP\_Subtract' or event.keysym == 'minus':  
 self.speed('-')  
  
 def create\_head\_food(self):  
 rand\_vect = random.randint(1, 4)  
 if rand\_vect == 1:  
 self.\_\_vector = self.CONST.RIGHT.value  
 elif rand\_vect == 2:  
 self.\_\_vector = self.CONST.DOWN.value  
 elif rand\_vect == 3:  
 self.\_\_vector = self.CONST.LEFT.value  
 else:  
 self.\_\_vector = self.CONST.UP.value  
 self.head = self.element\_square(self, self.\_\_snake\_x,  
 self.\_\_snake\_y,  
 self.CONST.SNAKE\_THICKNESS.value,  
 self.CONST.SNAKE\_HCOLOR.value)  
 self.food.add(self)  
 self.body = []  
 self.body.append({'id': self.head.draw(),  
 'x': self.\_\_snake\_x,  
 'y': self.\_\_snake\_y})  
 self.step('add')  
 self.step('add')  
 self.step('add')  
 self.step('add')  
  
 def speed(self, way):  
 if way == '+' and self.\_\_spped > 1:  
 self.\_\_spped -= 1  
 elif way == '-' and self.\_\_spped < 20:  
 self.\_\_spped += 1  
  
 def reload(self):  
 self.quit = 'n'  
 self.\_\_started = 1  
 self.\_\_spped = 10  
 self.canv.delete('all')  
 del self.body  
 self.body = []  
 self.create\_head\_food()  
 self.start()  
  
 def quit(self, event):  
 self.quit = 'y'  
  
 def move(self, event):  
 if self.quit != 'n':  
 self.start()  
  
 def start(self): # Бесконечный цикл движения змейки  
 if self.\_\_started == 1:  
 self.quit = 'n'  
 i = 0  
 add = 'del'  
 while i == 0:  
 self.step(add)  
 if self.food.eat(self) == 1:  
 add = 'add'  
 self.speed('+')  
 elif add == 'add':  
 add = 'del'  
 if self.bump\_wall() == 'the end':  
 break  
 if self.bump\_body() == 'the end':  
 break  
 for x in range(1, (self.\_\_spped + 1)):  
 time.sleep(0.05)  
 self.\_\_window.update()  
 if self.quit == 'y':  
 i = 1  
 break  
  
 def bump\_wall(self): # Проверка на столкновение со стеной  
 \_\_head\_x = self.body[-1]['x']  
 \_\_head\_y = self.body[-1]['y']  
 if ((\_\_head\_x < ((self.CONST.SNAKE\_THICKNESS.value // 2) + 1))  
 or (\_\_head\_y < ((self.CONST.SNAKE\_THICKNESS.value // 2) + 1))  
 or (\_\_head\_x > (self.canv\_width  
 - (self.CONST.SNAKE\_THICKNESS.value // 2) + 1))  
 or (\_\_head\_y > (self.canv\_height  
 - (self.CONST.SNAKE\_THICKNESS.value // 2) + 1))):  
 self.explosive()  
 return 'the end'  
 else:  
 return 0  
  
 def bump\_body(self): # Проверка на столкновение с телом змеи  
 \_\_head\_x = self.body[-1]['x']  
 \_\_head\_y = self.body[-1]['y']  
 bump = 0  
 for i in range(0, (len(self.body) - 1)):  
 if ((\_\_head\_x == self.body[i]['x'])  
 and (\_\_head\_y == self.body[i]['y'])):  
 self.explosive()  
 bump = 'the end'  
 return bump  
  
 def explosive(self):  
 self.\_\_started = 0  
 self.canv.create\_oval((self.body[-1]['x']  
 - self.CONST.EXPLOSIVE.value),  
 (self.body[-1]['y']  
 - self.CONST.EXPLOSIVE.value),  
 (self.body[-1]['x']  
 + self.CONST.EXPLOSIVE.value),  
 (self.body[-1]['y']  
 + self.CONST.EXPLOSIVE.value),  
 fill=self.CONST.EXPLOSIVE\_BCOLOR.value,  
 outline=self.CONST.EXPLOSIVE\_CCOLOR.value,  
 width=self.CONST.EXPLOSIVE\_BORD.value)  
  
 def step(self, add): # Двигать тело змеюки в текущую сторону на 1 шаг  
 # При этом тело может увеличиться (add='add') в размерах или нет  
 if self.\_\_vector == self.CONST.RIGHT.value:  
 deltax = self.CONST.SNAKE\_THICKNESS.value  
 deltay = 0  
 elif self.\_\_vector == self.CONST.DOWN.value:  
 deltax = 0  
 deltay = self.CONST.SNAKE\_THICKNESS.value  
 elif self.\_\_vector == self.CONST.LEFT.value:  
 deltax = -self.CONST.SNAKE\_THICKNESS.value  
 deltay = 0  
 elif self.\_\_vector == self.CONST.UP.value:  
 deltax = 0  
 deltay = -self.CONST.SNAKE\_THICKNESS.value  
 self.head.x += deltax  
 self.head.y += deltay  
 self.head = self.element\_square(self, self.head.x, self.head.y,  
 self.CONST.SNAKE\_THICKNESS.value,  
 self.CONST.SNAKE\_HCOLOR.value)  
 self.body.append({'id': self.head.draw(), 'x': self.head.x,  
 'y': self.head.y})  
 self.canv.itemconfig(self.body[-2]['id'],  
 fill=self.CONST.SNAKE\_BCOLOR.value)  
 if add != 'add':  
 self.canv.delete(self.body[0]['id'])  
 self.body.pop(0)  
  
 class food:  
 def add(self):  
 self.food.x = random.randint(self.CONST.FOOD\_THICKNESS.value  
 // 2, self.canv\_width  
 - self.CONST.FOOD\_THICKNESS.value // 2)  
 self.food.y = random.randint(self.CONST.FOOD\_THICKNESS.value  
 // 2, self.canv\_height  
 - self.CONST.FOOD\_THICKNESS.value // 2)  
 self.food.body = self.element\_square(self, self.food.x,  
 self.food.y,  
 self.CONST.FOOD\_THICKNESS.value,  
 self.CONST.FOOD\_COLOR.value)  
 self.food.id = self.food.body.draw()  
  
 def eat(self):  
 head\_x = self.body[-1]['x']  
 head\_y = self.body[-1]['y']  
 eat = 0  
 if ((head\_x  
 + self.CONST.SNAKE\_THICKNESS.value // 2 > (self.food.x  
 - self.CONST.FOOD\_THICKNESS.value // 2))  
 and (head\_x  
 - self.CONST.SNAKE\_THICKNESS.value // 2 < (self.food.x  
 + self.CONST.FOOD\_THICKNESS.value // 2))  
 and (head\_y  
 + self.CONST.SNAKE\_THICKNESS.value // 2 > (self.food.y  
 - self.CONST.FOOD\_THICKNESS.value // 2))  
 and (head\_y  
 - self.CONST.SNAKE\_THICKNESS.value // 2 < (self.food.y  
 + self.CONST.FOOD\_THICKNESS.value // 2))):  
 self.canv.delete(self.food.id)  
 self.food.add(self)  
 eat = 1  
 return eat  
  
 class element\_square:  
 def \_\_init\_\_(self, self\_glob, x, y, d, color):  
 self.self\_glob = self\_glob  
 self.x = x  
 self.y = y  
 self.d = d  
 self.color = color  
 if (self.d % 2) == 0:  
 self.d += 1 # сторону квадрата делаю нечётной  
  
 def draw(self):  
 x = self.x - (self.d // 2)  
 y = self.y - (self.d // 2)  
 return self.self\_glob.canv.create\_rectangle(x, y, x + self.d,  
 y + self.d,  
 fill=self.color,  
 width=2)  
  
  
def main():  
 image1\_data = '''R0lGODlhSgBKAOf/AABBAxJBCgBLAgdKBQRPABVKCwBTAAVRBwBYBRBXCQBfAQxcAAdeDAVhAABnAABoCwpmCQZnFQRsBQBtEwBvABtnFhNqEQByAAByCwJ1AglxFg9xCwB6AAB6CAt4BwB9ACVxJQeAAgCDBRh6GwCEACB3JCV3HQCGAAeCFACHCwWJ'''  
 image2\_data = '''R0lGODlhSgBKAOf/ABBHBwBOBARPAABTAABXBABbAAdZAAtXDgBeAA9dAQleDQJiAxheGgdlCABoAA1lAA5lEwRtBQJuEwBxAAB0AQlyDBhtExptHAl3BwB7AAB7CRB4AAB9ACNwJi1sMRp3EwiAAwGBEgCDBSB3HgCEAACGAACHAACGCy12JQKIABiAFyt6MAGKEQeKADt3OCd/JQCPBxiHECp/LBKNBiKGIQ+PFxiQCwaWEh+OITGJJjGJLj2FMzSINBiTHB2TEDqIPUKGQR2VHySUL0mHSR2YKiGYISmVKFGJUj6RQD2TMCOcLiicJjeWO356hTKbLkqRTIN8eC+ '''  
  
 def button\_press(a):  
 reload\_button['image'] = reload\_button\_img2  
 snake.reload()  
  
 def button\_unpress(a):  
 reload\_button['image'] = reload\_button\_img1  
  
 root = Tk()  
 root.title('Программа Змейка на питоне')  
 root.geometry('800x600+150+150')  
  
 frame = Frame(root, width=740, height=90, bg='#67E300')  
 frame.place(x=30, y=5)  
 text = Label(root, text='''Игра Змейка Правила: Змейка должна кушать  
зелёные плоды. При съедании плода, скорость змейки возрастает. Скорость можно  
отрегулировать вручную клавишами "+" и "-". Нельзя выползать за границы поля и есть себя.''',  
 bg='#67E300', width=79)  
 text.place(x=30, y=10)  
 reload\_button\_img1 = PhotoImage(data=image1\_data)  
 reload\_button = Label(image=reload\_button\_img1, bg='#f2ffe0')  
 reload\_button.place(x=675, y=13)  
 reload\_button\_img2 = PhotoImage(data=image2\_data)  
 reload\_button.bind('<Button-1>', button\_press)  
 reload\_button.bind('<ButtonRelease-1>', button\_unpress)  
  
 snake = python\_snake(root, 30, 100, 740, 470)  
 snake.start()  
  
 root.mainloop()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Экранные формы

