Итоговый проект по дисциплине “Основы алгоритмизации и программирования”

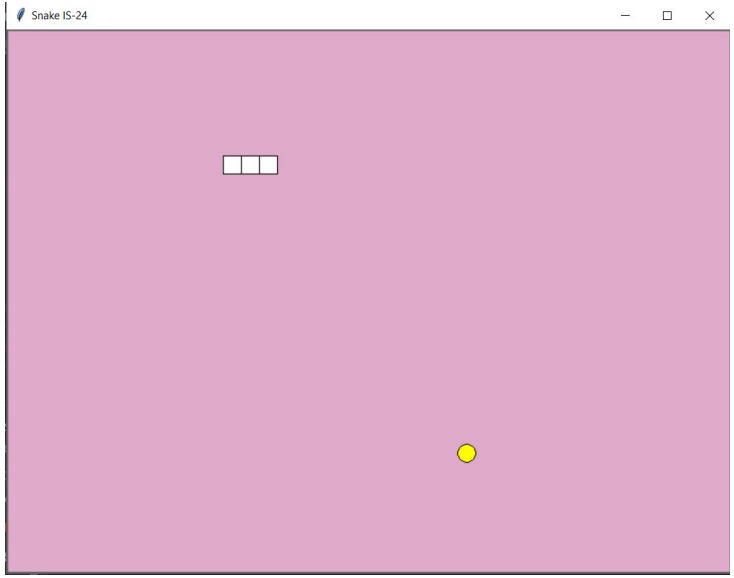
Тема: Разработка игры “Змейка” на Python

Руководитель проекта: Манакова Ольга Петровна

Разработчики: ст-ки гр. ИС-24
Королева Карина и Тарасевич Анна

Цель проектной работы

Разработать игру “Змейка” на языке Python с графическим интерфейсом

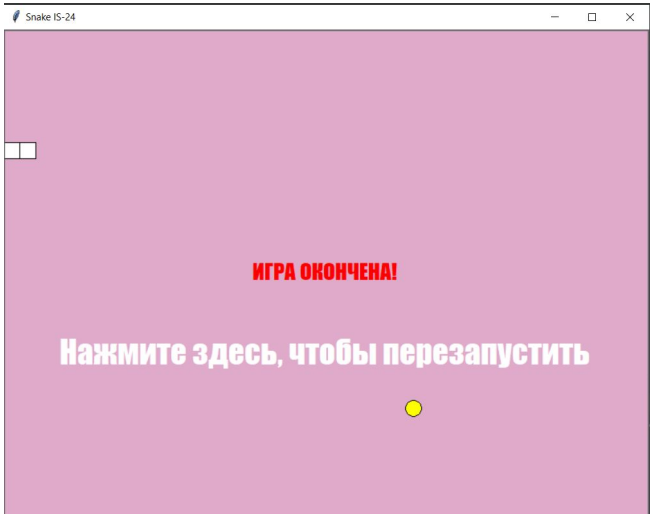


Задачи проектной работы

1. Обсуждение идеи и дизайна
2. Изучение вспомогательной литературы
3. Написание кода
4. Тестирование
5. Устранение ошибок
6. Создание презентации
7. Защита

Обсуждение

При обсуждении идеи, мы решили, что хотим создать игру “Змейка”, так как это показалось нам наиболее интересным вариантом в плане разработки. Дизайн выбрали пастельный, минималистичный.



Изучение

Далее мы принялись изучать материал, который нам пригодится для создания проекта. Первым делом мы познакомились с библиотекой Tkinter - основа графического интерфейса игры. Вспомнили основы языка и ООП.

Написание кода

Создание графического окна приложения:
Мы будем работать с библиотеками Random и Tkinter.
Объявление переменных:
Создали переменные для чтобы задать размеры игрового поля, сегмента змейки и состояния игры.

```
from tkinter import Tk, Canvas
import random

# Размеры
WIDTH = 800
HEIGHT = 600
SEG_SIZE = 20
# Переменная отвечающая за состояние игры
IN_GAME = True
```

Написание кода

Функция create_block создает поинт, который поедает змейка и увеличивается в размере.

```
# Вспомогательные функции
AnnaTarasevich
def create_block():
    """ Создание поинта"""
    global BLOCK
    posx = SEG_SIZE * random.randint(1, (WIDTH-SEG_SIZE) / SEG_SIZE)
    posy = SEG_SIZE * random.randint(1, (HEIGHT-SEG_SIZE) / SEG_SIZE)
    BLOCK = c.create_oval(posx, posy,
                          posx+SEG_SIZE, posy+SEG_SIZE,
                          fill="yellow")
```

Написание кода

Функция main нужна для управления игровым процессом

AnnaTarasevich

```
def main():
    """ Управление игровым процессом """
    global IN_GAME
    if IN_GAME:
        s.move()
        head_coords = c.coords(s.segments[-1].instance)
        x1, y1, x2, y2 = head_coords
        # Проверка, нет ли столкновения с краями игрового поля
        if x2 > WIDTH or x1 < 0 or y1 < 0 or y2 > HEIGHT:
            IN_GAME = False
        # Поедание поинтов
        elif head_coords == c.coords(BLOCK):
            s.add_segment()
            c.delete(BLOCK)
            create_block()
        # Самоедство
        else:
            for index in range(len(s.segments)-1):
                if head_coords == c.coords(s.segments[index].instance):
                    IN_GAME = False
        root.after(100, main)
    # Если не в игре выводим сообщение о проигрыше
    else:
        set_state(restart_text, 'normal')
        set_state(game_over_text, 'normal')
```

Написание кода

Создание классов сегмента и змейки:

Класс сегмента змейки.

Сегмент змейки будет простым прямоугольником, созданным при помощи метода create_rectangle класса Canvas модуля tkinter.

```
AnnaTarasevich
class Segment(object):
    """ Класс сегмента змейки """
    AnnaTarasevich
    def __init__(self, x, y):
        self.instance = c.create_rectangle(x, y,
                                           x+SEG_SIZE, y+SEG_SIZE,
                                           fill="white")
```

Написание кода

Класс змейки.
Змейка у нас будет набором сегментов. У нее будут методы движения, изменения направления и добавления сегмента.

```
AnnaTarasevich
class Snake(object):
    """ Класс змейки """
    AnnaTarasevich
    def __init__(self, segments):
        self.segments = segments
        #список доступных направлений движения змейки
        self.mapping = {"Down": (0, 1), "Right": (1, 0),
                        "Up": (0, -1), "Left": (-1, 0)}
        #изначально змейка двигается вправо
        self.vector = self.mapping["Right"]
```

Написание кода

Функция move позволяет управлять змейкой.
Она задает направления, в котором будет ползти змейка и как она будет двигаться.

AnnaTarasevich

```
def move(self):
    """Двигает змейку в заданном направлении"""
    # перебираем все сегменты кроме первого
    for index in range(len(self.segments)-1):
        segment = self.segments[index].instance
        x1, y1, x2, y2 = c.coords(self.segments[index+1].instance)
        # задаем каждому сегменту позицию сегмента стоящего после него
        c.coords(segment, x1, y1, x2, y2)

    # получаем координаты сегмента перед "головой"
    x1, y1, x2, y2 = c.coords(self.segments[-2].instance)
    # помещаем "голову" в направлении указанном в векторе движения
    c.coords(self.segments[-1].instance,
             x1+self.vector[0]*SEG_SIZE, y1+self.vector[1]*SEG_SIZE,
             x2+self.vector[0]*SEG_SIZE, y2+self.vector[1]*SEG_SIZE)
```

Написание кода

Функция add_segment создает сегмент змейки

```
def add_segment(self):
    """ Создает сегмент змейки """
    # определяем последний сегмент
    last_seg = c.coords(self.segments[0].instance)
    # определяем координаты куда поставить следующий сегмент
    x = last_seg[2] - SEG_SIZE
    y = last_seg[3] - SEG_SIZE
    # добавляем змейке еще один сегмент в заданных координатах
    self.segments.insert(0, Segment(x, y))
```

Написание кода

Функция change_direction меняет направление движения змейки

AnnaTarasevich

```
def change_direction(self, event):
    """ Меняет направление движения змейки """
    # event передаст нам символ нажатой клавиши
    # и если эта клавиша в доступных направлениях
    # изменяем направление
    if event.keysym in self.mapping:
        self.vector = self.mapping[event.keysym]
```

Написание кода

Сбрасывает змейку при поражении

AnnaTarasevich

```
def reset_snake(self):
    for segment in self.segments:
        c.delete(segment.instance)
```

Устанавливает состояние змейки

AnnaTarasevich

```
def set_state(item, state):
    #устанавливаем состояние змейки
    c.itemconfigure(item, state=state)
```

Написание кода

Функция clicked перезапускает игру при нажатии после поражения

AnnaTarasevich

```
def clicked(event):
    global IN_GAME
    s.reset_snake()
    IN_GAME = True
    c.delete(BLOCK)
    c.itemconfigure(restart_text, state='hidden')
    c.itemconfigure(game_over_text, state='hidden')
    start_game()
```

Написание кода

Функция start_game создает змейку и запускает игру

AnnaTarasevich

```
def start_game():
    global s
    create_block()
    s = create_snake()
    # Реакция на нажатие клавиши
    c.bind("<KeyPress>", s.change_direction)
    main()
```

Написание кода

Функция create_snake создает сегменты и саму змейку

```
def create_snake():
    # создание сегментов и змейки
    segments = [Segment(SEG_SIZE, SEG_SIZE),
                Segment(SEG_SIZE*2, SEG_SIZE),
                Segment(SEG_SIZE*3, SEG_SIZE)]
    return Snake(segments)
```

Написание кода

После того, как написали все функции, нужно создать окно, где они будут выполняться.

```
#Создаем окно
root = Tk()
# Устанавливаем название окна
root.title("Snake IS-24")
```

Написание кода

Далее устанавливаем дизайн окна и наводим фокус, чтобы змейка ловила нажатия

```
# создаем экземпляр класса Canvas и заливаем все розовым цветом

c = Canvas(root, width=WIDTH, height=HEIGHT, bg="#e0abcb")
c.grid()
# Наводим фокус на Canvas, чтобы мы могли ловить нажатия клавиш
c.focus_set()
```

Написание кода

Последнее, выводим надписи: “Игра окончена!” и “Нажмите, чтоб перезагрузить”, запускаем игру

```
game_over_text = c.create_text(WIDTH/2, HEIGHT/2, text="ИГРА ОКОНЧЕНА!",
                               font='IMPACT 20', fill='red',
                               state='hidden')
restart_text = c.create_text(WIDTH/2, HEIGHT-HEIGHT/3,
                             font='IMPACT 30',
                             fill='white',
                             text="Нажмите здесь, чтобы перезапустить",
                             state='hidden')
c.tag_bind(restart_text, "<Button-1>", clicked)
start_game()
root.mainloop()
```

Тестирование и устранение ошибок

Во время теста была обнаружена лишь ошибка отображения окна, но в процессе повторных запусков она исчезла

Вывод

В результате проделанной работы, мы разработали игру “Змейка” на Python с интерфейсом. Цель работы и задачи полностью выполнены, произведено тестирование и улучшение кода, готовый код можете увидеть на гитхаб.

Литература

https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter

https://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python

https://proglib.io/p/python-oop

Лекции по ООП с пар по алгоритмизации