Лабораторная работа №23
Разработка оконного приложения

1 Цель работы

1.1 Формирование умения работать в среде программирования IDLE.

1.2 Формирование умения оформлять код программы в соответствии со стандартом кодирования.

1.3 Формирование умения выполнять проверку, отладку кода программы.

2 Литература

2.1 Колдаев, В. Д. Основы алгоритмизации и программирования : учебное пособие / В.Д. Колдаев ; под ред. проф. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 414 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0733-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1735805. – Режим доступа: по подписке.

3 Основное оборудование

3.1 Персональный компьютер.

4 Подготовка к работе

4.1 Прочитать конспект и рекомендуемую литературу.

5 Задание

5.1 Выполнить задания п. 6.

5.2 Выполнить проверку разработанных программ.

5.3 Отобразить полученный листинг программ и проделанные тесты в отчете.

6 Порядок выполнения работы

6.1 Разработать игровое приложение “Змейка”

Описание игрового процесса: игрок управляет змейкой при помощи кнопок направления — вверх, вниз, влево, вправо. Ползая, змейка должна собирать еду, за которую начисляются очки. Цель игры — набрать как можно больше очков. Игра заканчивается, если змейка врезается в стену или в себя.

Для написания игры необходимо проделать следующие шаги:

- импортировать модули в программу,
- создать экран игры с помощью модуля Turtle,
- задать кнопки направления движения змеи,
- реализовать игровой процесс.

6.2 Импортировать модули: turtle, time и random.

6.3 Задать следующие значения по умолчанию:

- начальный счет — score,
- наибольший счет — high_score,

- время задержки — delay (см. Рисунок 1).

```
1 import turtle
2 import time
3 import random
4
5 score = 0
6 high_score = 0
7 delay = 0.1
```

Рисунок 1 – Импорт модулей и начальные значения переменных

6.4 **Создайте экран игры**. Укажите размер экрана, цвет фона и название программы. В функции wind.tracer установите значение задержки обновления экрана (см. Рисунок 2).

```
1 # Creating a window screen
2 wind = turtle.Screen()
3 wind.title("Snake Maze")
4 wind.bgcolor("green")
5
6 # the width and height can be put as user's choice
7 wind.setup(width=600, height=600)
8 wind.tracer(0)
```

Рисунок 2 – Создание экрана игры

6.5 **Создайте главного персонажа - змейку**. Метод penup() тут нужен для того, чтоб змейка не рисовала линию при движении, а goto(x,y) задает координаты, которые перемещают змею в абсолютное положение (см. Рисунок 3).

```
# head of the snake
head = turtle.Turtle()
head.shape("square")
head.color("white")
head.penup()
head.goto(0, 0)
head.direction = "Stop"
```

Рисунок 3 – Создание игрового персонажа

6.6 **Добавьте еду и счетчик**, который будет отображать текущий и рекордный счет (см. рисунок 4).

```
# food in the game
food = turtle.Turtle()
colors = random.choice(['red', 'green', 'black'])
shapes = random.choice(['square', 'triangle', 'circle'])
food.speed(0)
food.shape(shapes)
food.color(colors)
food.penup()
food.goto(0, 100)


pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 250)
pen.write("Score : 0  High Score : 0", align="center",
font=("Arial", 24, "bold"))
```

Рисунок 4 – Создание счетчика и игровых объектов

6.7 Запустите программу, сделайте скриншот и вставьте в электронный

отчет.

6.8 Задайте клавиши направления, при нажатии на которые змейка будет двигаться. В нашем случае кнопки будут следующими: “W”— вверх, “A”— влево, “S”— вниз, “D”— вправо. Используйте функции (см. рисунок 5).

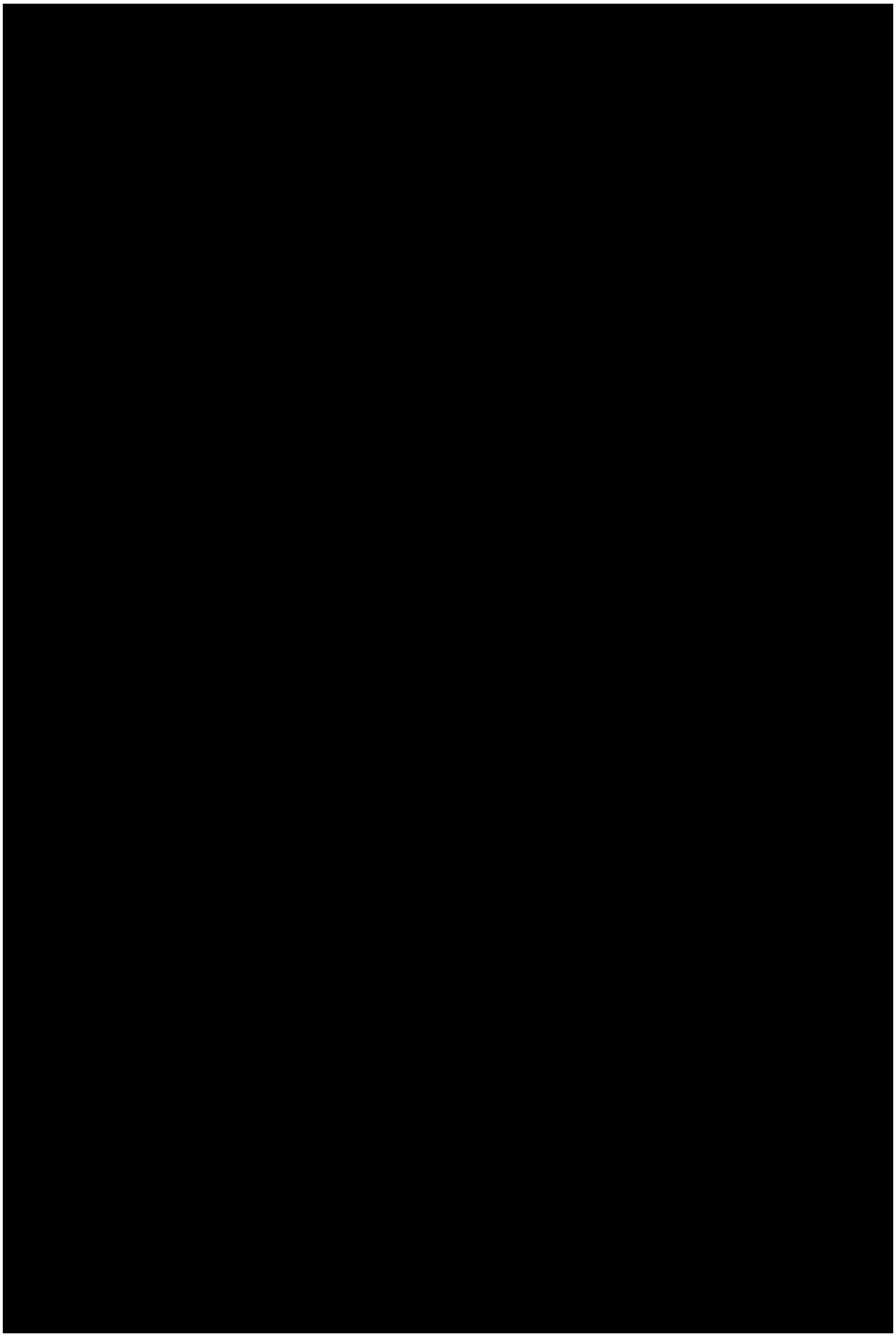

Рисунок 5 – Создание элементов управления

6.9 **Реализовать геймплей.** Он включает в себя следующие пункты:

1) Длина змейки увеличивается каждый раз, когда она собирает еду. При этом хвост и голова змейки должны отличаться по цвету.

2) Счет увеличивается каждый раз, когда змея подбирает еду. Наибольший результат записывается.

3) Должна быть добавлена проверка на столкновение головы змейки с телом или стеной.

4) Цвет и форма еды должны меняться при каждом перезапуске игры.

5) Игра начинается заново, если змея столкнулась с собой или стеной.

6) В случае столкновения счетчик обнуляется, наибольший результат сохраняется до перезапуска.

6.9.1 Добавить оставшуюся часть кода (см. рисунок 6).

```
segments = []

# Main Gameplay
while True:
    wind.update()
    if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor()
        time.sleep(1)
        head.goto(0, 0)
        head.direction = "Stop"
        colors = random.choice(['red', 'blue', 'green'])
        shapes = random.choice(['square', 'circle'])
        for segment in segments:
            segment.goto(1000, 1000)
        segments.clear()
        score = 0
        delay = 0.1
        pen.clear()
        pen.write("Score : {} High Score : {} ".format(
            score, high_score), align="center", font=("Arial", 24, "bold"))
    if head.distance(food) < 20:
        x = random.randint(-270, 270)
        y = random.randint(-270, 270)
        food.goto(x, y)

        # Adding segment
        new_segment = turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape("square")
        new_segment.color("orange")  # tail colour
        new_segment.penup()
        segments.append(new_segment)
        delay -= 0.001
        score += 10
        if score > high_score:
            high_score = score
        pen.clear()
        pen.write("Score : {} High Score : {} ".format(
            score, high_score), align="center", font=("Arial", 24, "bold"))
    # Checking for head collisions with body segments
    for index in range(len(segments)-1, 0, -1):
        x = segments[index-1].xcor()
        y = segments[index-1].ycor()
        segments[index].goto(x, y)
    if len(segments) > 0:
        x = head.xcor()
        y = head.ycor()
        segments[0].goto(x, y)
    move()
    for segment in segments:
        if segment.distance(head) < 20:
            time.sleep(1)
            head.goto(0, 0)
            head.direction = "stop"
            colors = random.choice(['red', 'blue', 'green'])
            shapes = random.choice(['square', 'circle'])
            for segment in segments:
                segment.goto(1000, 1000)
            segment.clear()

            score = 0
            delay = 0.1
            pen.clear()
            pen.write("Score : {} High Score : {} ".format(
                score, high_score), align="center", font=("Arial", 24, "bold"))
    time.sleep(delay)

wind.mainloop()
```

Рисунок 6 – Реализация геймплея

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Условия задач
7.4 Подробное содержание работы
7.5 Вывод по проделанной работе.

8 Контрольные вопросы

8.1 Какие модули вы использовали для разработки игрового приложения?
8.2 Как реализовать механику перемещения объекта по экрану?
8.3 Как сделать счетчик?