Wydział Elektrotechniki, Automatyki i Informatyki

Projekt zaliczeniowy – Języki skryptowe (Python)

Przedmiot: Języki Skryptowe

Prowadzący: Dr. Inż. Dariusz Michalski

Temat projektu: Gra typu “Wąż”

Autorzy: Aleksandra Stawińska, Adam Szewczyk

Grupa: 2ID14A

Data oddania: 25.06.2025

1. **Opis projektu**
   Celem projektu było stworzenie klasycznej gry „Snake” z wykorzystaniem języka Python i biblioteki Pygame. Gra umożliwia sterowanie wężem, zbieranie jedzenia, zdobywanie punktów oraz zapisywanie wyników. Projekt miał na celu zastosowanie różnych koncepcji programistycznych, takich jak:

   Programowanie obiektowe (OOP)

   Obsługa plików (JSON)

   Programowanie funkcyjne (map, reduce, filter, lambda)

   Testowanie jednostkowe

   Wizualizacja danych (wykresy wyników)

2. **Zakres funkcjonalności**

   Główne funkcje gry:
   Sterowanie wężem za pomocą klawiszy strzałek.

   Zwiększanie długości węża po zjedzeniu jedzenia.

   System punktacji i zapisywanie wyników do pliku scores.json.

   Kolizje ze ścianami i własnym ciałem (koniec gry).

   Możliwość pauzowania (klawisz ESC) i restartu (klawisz R).

   Dynamiczne przyspieszanie gry co 50 punktów.

   Wizualizacja wyników (wykres w matplotlib).

   Testy jednostkowe sprawdzające logikę gry.

3. **Struktura projektu**

   ```
   /snake_game
   │  main.py          # Główny plik uruchamiający grę
   │  requirements.txt    # Wymagane biblioteki
   │  README.txt         # Dokumentacja
   ```

```
│
├──game/             # Główna logika gry
│   │  game.py        # Klasa zarządzająca grą
│   │  snake.py       # Klasa węża
│   │  food.py        # Klasa jedzenia
│   │  high_scores.py   # Zarządzanie wynikami
│   │
│   ├──utils/         # Narzędzia pomocnicze
│   │   │  helpers.py   # Funkcje pomocnicze (np. rysowanie tekstu)
│   │   │  visualizer.py # Generowanie wykresów wyników
│   │
│   └──tests/         # Testy jednostkowe
│      │  test_snake.py # Testy klasy Snake
│      │  test_game.py  # Testy logiki gry
│
├──data/             # Dane gry
│   │  scores.json      # Zapisane wyniki
│
└──assets/           # Zasoby graficzne
   │  plot.png        # Wykres wyników
```

4. **Wykorzystane technologie i biblioteki**

   Python 3.10+ – język programowania

   Pygame – biblioteka do tworzenia gier

   Matplotlib – generowanie wykresów

   JSON – zapis i odczyt wyników

   Pytest – testy jednostkowe

   Flake8 – kontrola jakości kodu

5. Sposób działania programu

➢ Instrukcja uruchomienia

1. **Wymagania wstępne:**

a. Python 3.8 lub nowszy

b. Zainstalowane biblioteki z requirements.txt

2. **Instalacja zależności:**

```
bash
Copy
Download
pip install -r requirements.txt
```

3. **Uruchomienie gry:**

```
bash
Copy
Download
python main.py
```

4. **Sterowanie w grze:**

a. **Strzałki (↑ ↓ ← →)** – kierowanie wężem

b. **ESC** – pauza

c. **R** – restart po przegranej

➢ Przykładowe dane wejściowe/wyjściowe

Dane wejściowe:

- Plik konfiguracyjny (config.json):

```
json
Copy
Download
{
 "width": 800,
 "height": 600,
 "fps": 10,
 "block_size": 20,
 "colors": {
  "background": [0, 0, 0],
  "snake": [0, 255, 0],
```

```
  "food": [255, 0, 0],
  "text": [255, 255, 255]
 }
}
```

Dane wyjściowe:

- Wyniki zapisywane w data/scores.json:

```
json
Copy
Download
[120, 80, 50, 200, 30]
```

- Wykres wyników (assets/plot.png):

  /assets/plot.png

➢ Zrzuty ekranu (GUI)

1. Ekran główny gry

Wąż (zielony) porusza się po planszy, zbierając jedzenie (czerwone).

2. Pauza w grze

Gra zatrzymana – możliwość wznowienia klawiszem ESC.

3. Koniec gry (Game Over)

Komunikat o przegranej i możliwość restartu (klawisz R).

4. Wykres wyników

Wizualizacja najlepszych wyników wygenerowana przez matplotlib.

6. Przykładowe fragmenty kodu

a) Klasa Snake (game/snake.py)

python
Copy
Download

```python
class Snake:
    def __init__(self, grid_width, grid_height, block_size):
        self.grid_width = grid_width
        self.grid_height = grid_height
        self.block_size = block_size
        self.reset()

    def move(self):
        head_x, head_y = self.body[0]
        if self.direction == 'UP':
            new_head = (head_x, head_y - 1)
        elif self.direction == 'DOWN':
            new_head = (head_x, head_y + 1)
        elif self.direction == 'LEFT':
            new_head = (head_x - 1, head_y)
        elif self.direction == 'RIGHT':
            new_head = (head_x + 1, head_y)

        self.body.insert(0, new_head)
        if self.grow_length > 0:
            self.grow_length -= 1
        else:
            self.body.pop()
```

b) Zapisywanie wyników (game/high_scores.py)

python
Copy
Download

```python
class HighScores:
    def __init__(self, filename):
        self.filename = filename
        self.scores = self.load_scores()

    def add_score(self, score):
        self.scores.append(score)
        self.scores.sort(reverse=True)
```

```
    self.scores = self.scores[:10]  # Top 10 wyników
    self.save_scores()
```

c) Test jednostkowy (game/tests/test_snake.py)

```
python
Copy
Download
def test_snake_grow(self):
  initial_length = len(self.snake.body)
  self.snake.grow()
  self.snake.move()
  self.assertEqual(len(self.snake.body), initial_length + 1)
```

7. Testowanie

- **Testy jednostkowe** sprawdzają poprawność ruchów węża i kolizji.
- **Testy funkcjonalne** weryfikują działanie całej gry.
- **Testowanie ręczne** (sprawdzanie reakcji na klawisze, zapis wyników).
- **Kontrola jakości kodu** (flake8).

8. Wnioski

Co się udało?

☑ Pełna implementacja gry „Snake” z wszystkimi wymaganymi funkcjonalnościami.

☑ Poprawna struktura projektu (OOP, moduły, testy).

☑ Zastosowanie różnych koncepcji programistycznych (obsługa plików, wykresy, testy).

Co można poprawić?

- Dodać więcej poziomów trudności.
- Wprowadzić różne rodzaje jedzenia (np. przyspieszające).
- Dodać efekty dźwiękowe.

9. Załączniki

- Kod źródłowy (/snake_game)
- Wykres wyników (assets/plot.png)
- Plik z wynikami (data/scores.json)

Podsumowanie: Projekt został zrealizowany zgodnie z założeniami i spełnia wszystkie wymagania. Gra działa poprawnie, a kod jest czytelny i dobrze zorganizowany.