Politechnika Wrocławska

Platformy programistyczne .Net i Java

Snake AI

Imię, nazwisko, nr indeksu	**Anatolii Shevchuk 261049** **Nikodem Swakoń 263522**
Termin zajęć	**Wtorek, 18:55**
Data wykonywania	**10.06.2025**

1 Wprowadzenie

1.1 Cel projektu

Celem projektu było stworzenie gry komputerowej typu Snake w języku Java, z dodatkiem przeciwników sterowanych przez sztuczną inteligencję. Projekt miał na celu praktyczne zastosowanie wiedzy z zakresu programowania obiektowego, grafiki komputerowej, obsługi zdarzeń oraz wielowątkowości.

1.2 Zakres projektu

Projekt obejmował:

- implementację ruchu gracza i przeciwników AI,
- wykrywanie kolizji i mechanikę punktacji,
- dodanie dwóch poziomów trudności: `NORMAL` i `HARD`,
- zapis i odczyt wyników do pliku,
- uruchamianie AI w osobnych wątkach,

1.3 Motywacja

Wybór tematu gry Snake wynikał z jego prostoty oraz możliwości rozszerzenia o bardziej zaawansowane elementy, takie jak przeciwnicy sterowani przez AI i obsługa wielu wątków. Projekt stanowił dobre ćwiczenie w integracji różnych aspektów programowania: logiki gry, interfejsu, AI i dokumentacji.

2 Opis rozwiązania

2.1 Ogólny opis systemu

Gra rozpoczyna się od wyświetlenia wyników najwyższych graczy i wczytania konfiguracji planszy. Gracz steruje wężem za pomocą klawiatury, zbierając jabłka i unikając przeszkód oraz przeciwników. Na planszy znajdują się dwa węże AI. AI działają niezależnie w osobnych wątkach, a ich logika podejmowania decyzji bazuje na kierunku do gracza i do jabłka.

2.2 Technologie

- **Java 17** – język programowania,
- **Swing** – biblioteka graficzna do GUI,
- **Doxygen** – narzędzie do generowania dokumentacji technicznej,
- **IntelliJ IDEA** – środowisko IDE,
- **JOptionPane** – do interakcji z użytkownikiem (np. wpisanie nicku).

3 Szczegóły implementacji

3.1 Główne komponenty aplikacji

- **GamePanel** – główna klasa zarządzająca logiką gry, rysowaniem planszy, ruchem i kolizjami.
- **Snake, Apple** – klasy modelujące węża i jabłko.
- **SimpleSnakeAI, AggressiveSnakeAI** – klasy implementujące logikę sztucznej inteligencji. Simple porusza się do jabłka, Aggressive reaguje na gracza.
- **HighScoreManager** – zapis i odczyt wyników z pliku tekstowego.

- **LevelType** – enum definiujący poziom trudności (NORMAL lub HARD).
- **Timer** – do synchronizacji animacji gry.
- **Wątki (Thread)** – dwa niezależne wątki obsługujące ruch AI.

4 Testowanie

4.1 Opis testów

Testy przeprowadzono ręcznie poprzez:

- weryfikację działania sterowania klawiszami,
- sprawdzenie poprawności wykrywania kolizji ze ścianą, wężem i przeszkodami,
- testy logiki AI w zależności od pozycji gracza i jabłka,
- testy działania wątków – poprawność ich synchronizacji i brak błędów współbieżności.

4.2 Wyniki testów

Wszystkie testy zakończyły się sukcesem. Gra działa płynnie, a zachowanie AI jest zgodne z oczekiwaniami. Nie zaobserwowano kolizji danych między wątkami. Wyniki zapisywane są prawidłowo, a interfejs działa poprawnie.

5 Podsumowanie

5.1 Osiągnięcia

- Stworzono kompletną grę z GUI i AI,
- AI działa niezależnie w osobnych wątkach,
- Obsługa poziomów trudności i zapis wyników do pliku,

5.2 Wnioski

Projekt pozwolił utrwalić wiedzę z zakresu:

- obsługi GUI w Swingu,
- programowania współbieżnego i synchronizacji,
- projektowania logiki AI,

6 Dokumentacja

Dokumentacja została wygenerowana za pomocą narzędzia **Doxygen**. Wszystkie klasy, pola i metody zawierają odpowiednie komentarze w stylu Doxygen. Po uruchomieniu ‘doxygen Doxyfile‘, program wygenerował dokumentację HTML, którą można otworzyć w przeglądarce w pliku ‘index.html‘.

7 Załączniki

7.1 Kod źródłowy

Kod dostępny jest W repozytorium projektu GitHub