Snake

Programowanie I - projekt zaliczeniowy
Wydział Matematyki Stosowanej
Politechnika Śląska

Bartłomiej Pacia

Styczeń 2022

1 Introduction

"Snake" jest klonem popularnej "gry w węża". Rozgrywka toczy się na kwadratowej planszy. Na początku gracz startuje z wężem o długości 1 (tzn. zajmującym 1 kratkę). Jednocześnie w losowej kratce na mapie pojawia się punkt. Zadaniem gracza jest takie kierowanie swoim wężem przy użyciu klawiszy strzałek na klawiaturze, by zebrać punkt. Zebrany punkt znika, a długość węża gracza zwiększa się o 1. Następnie pojawia się nowy punkt i cały proces zaczyna się od nowa. Gracz przegra, jeśli uderzy głową węża w jego własne ciało lub wyjdzie poza mapę. Uniknięcie tej pierwszej sytuacji staje się coraz trudniejsze wraz ze wzrostem długości węża.

2 Wymagania

~~Przekreślone~~ elementy listy oznaczają wymagania zdefiniowane w pierwotnych Założeniach do Projektu, które ostatecznie nie zostały zrealizowane. **Pogrubione** elementy listy oznaczają nowe wymagania, niezdefiniowane w pierwotnych Założeniach do Projektu, które zostały zrealizowane.

- ~~ekran menu z 2 przyciskami ("Graj" i "Wyjdź")~~
- ekran gry z ~~2 przyciskami ("Pauza" i "Wyjdź"),~~ licznikiem obecnej liczby punktów oraz najwyższej liczby punktów

- zbieranie punktów i w konsekwencji wydłużanie się węża
- zapisywanie najwyższej liczby punktów (high score) do pliku i możliwość pobicia tego rekordu
- **możliwość podania rozmiaru planszy i interwału czasowego między ruchami jako argumenty do programu**

3 Przebieg realizacji

Starałem pisać się kod z użyciem nowych funkcjonalności zapewnianych przez standardy C++11, C++14 i C++17. W szczególności użyłem *smart pointers* zamiast *raw pointers* oraz przekazywania argumentów do funkcji i metod przez referencję. Użyłem też kontenera *std::vector* oraz funkcji z biblioteki *algorithm*.

3.1 Użyte biblioteki

Użyłem biblioteki standardowej C++ oraz biblioteki SFML, która udostępnia podstawowe funkcje do obsługi grafiki.

3.2 Użyte narzędzia

Użyłem systemu kontroli wersji *git* oraz serwisu *GitHub* do hostowania repozytorium projektu.

Użyłem też narzędzia *clang-format*, aby utrzymywać spójny styl kodu. Użyty przeze mnie styl to *Chromium*.

3.3 Skomplikowanie

W trakcie implementacji okazało się, że występuje sporo przypadków krańcowych, o których nie pomyślałem na początku.

Przykładowy taki przypadek występuje, gdy wąż zbierze punkt i jego długość zwiększa się. Gdzie wtedy powinien pojawić się nowy fragment węża?

Od dawna nie zajmowałem się pisaniem gier, i przypomniało mi się, jak niesamowicie szybko rośnie poziom skomplikowania kodu gry, szczególnie gdy pisze go niedoświadczona osoba, taka jak ja. Z powodu ograniczonego czasu oraz wiedzy (i ograniczonego czasu na jej uzupełnienie) nie użyłem żadnych

wzorców projektowych, które na pewno pomogłyby mi lepiej zdefiniować zależności między różnymi elementami gry i w konsekwencji zmniejszyć liczbę błędów oraz czasu wymaganego na testowanie. Jestem świadom bardzo sporego miejsca na poprawę.

4 Instrukcja użytkownika

4.1 Budowanie projektu

Aby skompilować projekt, potrzebny jest:

- kompilator C++, np. *g++* lub *clang++*
- program *make*
- źródła biblioteki SFML, którą można zainstalować stworzonym przeze mnie skryptem *install_sfml* dostępnym w projekcie

Po spełnieniu powyższych wymagań możemy w głównym katalogu projektu uruchomić program *make*

```
$ make snake
```

4.2 Uruchamianie gry

Grę należy uruchamiać z linii komend. Pozwala to na wyświetlanie pomocy oraz przekazanie mu argumentów wpływających na wygląd i gameplay.

Aby uruchomić grę z domyślnymi ustawieniami (ruch węża co 500 ms i plansza o rozmiarze 16 x 16 kratek), nie trzeba podawać żadnych argumentów.

```
$ ./snake
```

Aby uruchomić grę z wężem poruszającym się co 100 ms i planszą o rozmiarze 128 x 128 kratek, należy podać następujące argumenty:

```
$ ./snake --interval 100 --grid-size 128
```

Aby poruszać wężem należy używać strzałek.

5 Podsumowanie i wnioski

Z ważniejszych rzeczy, których nie zaimplementowałem, trzeba wymienić interfejs użytkownika, np. w postaci menu.

Kiedy nabędę więcej doświadczenia z tzw. nowoczesnym C++ oraz przeczytam książkę Game Programming Patterns, planuję zrefaktoryzować kod gry, tak by nie był zagmatwaną plątaniną wzajemnych, nieoczywistych zależności i zachowań pomiędzy komponentami.

Chciałbym również zainteresować się tematem testów jednostkowych w C++ z użyciem biblioteki Google Test, i (gdy już zrefaktoryzuję kod i będzie on możliwy do testowania), planuję takowe napisać. Sprawi to, że będę pewniejszy tego, że gra działa poprawnie i nie będę testować wszystkich jej funkcji po wprowadzeniu jakiejś zmiany.

Mam nadzieję rozwijać dalej grę, tak by nadawała się na również jako projekt zaliczeniowy na przedmiot Programowanie II.

Repozytorium projektu na GitHubie: github.com/bartekpacia/snake