

Projekt Sieci Komputerowe 2

Paweł Błoch

1 Opis Projektu

Projekt został zrealizowany samodzielnie. Tematem projektu jest turowa ¹ gra logiczna Reversi. Serwer i klient został zaimplementowany w języku C++. Klient został zaimplementowany w języku C++, ponadto korzysta z biblioteki graficznej SFML. Do komunikacji sieciowej wykorzystywane jest API `bsd-socket`. Projekt jest kompilowany na systemie operacyjnym linux.

2 Protokół komunikacji

Komunikacja jest prowadzona przy pomocy API `BSD Socket`. Komunikacja między serwerem i klientami polega na przesyłaniu stanu rozgrywki. Możliwe informacje przesyłane między serwerem a klientem to:

- Kolor zawodnika - 1 bajtowa wiadomość zawierająca informację o tym, jakim kolorem gra dany klient.
- tura - 1 bajtowa wiadomości mówiąca o tym, który gracz ma następny wykonać ruch. Informacja o turze jest przesyłana po każdym ruchu gracza, ponieważ zasady gry dopuszczają przypadki, gdy jeden z graczy wykonuje dwa lub więcej ruchów pod rząd.
- plansza - 64 bajtowa informacja zawierająca obecny stan gry. Stan gry jest przechowywany i obliczany na serwerze. Stan gry jest przesyłany po każdym prawidłowym ruchu jednego z graczy.
- zwycięzca - 1 bajtowa informacja przekazująca zwycięzcę rozgrywki
- poprawność ruchu - 1 bajtowa informacja która określa czy ruch wysłany do serwera jest poprawny z zasadami dla aktualnego stanu gry.
- błąd połączenia - 1 bajtowa informacja mówiąca o tym, że przeciwnik utracił połączenie z serwerem.

Informacje przesyłane między klientem a serwerem to:

- ruch - 2 bajtowa informacja opisująca ruch wykonany przez gracza.

¹Zasady gry dopuszczają sytuację, kiedy jeden z graczy wykonuje dwa lub więcej ruchów pod rząd.

3 Opis implementacji

Implementacja po stronie klienta przypomina implementację zgodną z paradygmatem obiektowym. Główną klasą jest klasa **Game**. Metoda **Game::run** odpowiada za przebieg i kontrolę rozgrywki. Klasa **BSDsocket** odpowiada za metody przeprowadzające komunikację sieciową. Dzięki niej możemy stworzyć połączenie, wysłać i odebrać dane oraz zamknąć połączenie. Implementacja klienta korzysta z wielowątkowości. W programie tworzone są 2 wątki. Pierwszy odpowiedzialny za przebieg rozgrywki i komunikację sieciową. Drugi odpowiada za obsługę zdarzeń. Wątki są tworzone z użyciem **std::thread**.

Implementacja po stronie serwera opiera się w głównej mierze na funkcji **game_room** która obsługuje rozgrywkę pomiędzy dwoma graczami. Klasa **Board** odpowiada za logiczny przebieg rozgrywki. W niej jest przetrzymywany stan gry. Klasa ta również odpowiada za sprawdzanie poprawności ruchu oraz przydzielanie następnej tury. Funkcje **my_read** i **send** odpowiadają za wysyłanie/odbieranie informacji.

4 Kompilacja i uruchomienie projektu

Aby skompilować projekt należy pobrać projekt z repozytorium. Kompilacji klienta i serwera dokonujemy oddzielnie. Do kompilacji będziemy potrzebowali biblioteki SFML (link do pobrania) oraz narzędzie Makefile. Aby skompilować klienta/serwer należy przejść do katalogu **reversi_client/reversi_server**. Uruchomienie klienta/serwera następuje przez komendę **./bin/client / ./bin/server**. Ponadto można uruchomić klienta podając mu jako argument adres ip. Gdy nie jest podany adres jako argument to domyślenie jest wykorzystywany adres 127.0.0.1.