Krzysztof Smal 155990

Sprawozdanie SK2

Sieciowa turowa gra logiczna - Reversi

1. Opis projektu:

Projekt polega na stworzeniu dwuosobowej gry Reversi w architekturze klient-serwer. Serwer pełni rolę centralnego elementu koordynującego grę, zarządzając stanem planszy, logiką rozgrywki oraz komunikacją z klientami. Klient natomiast umożliwia użytkownikowi interakcję z grą, poprzez wyświetlanie planszy i umożliwienie wykonywania ruchów. Gra toczy się na planszy o wymiarach 8x8, na której gracze na zmianę ustawiają swoje pionki. Pionki przeciwnika są przejmowane, jeśli znajdą się pomiędzy pionkami gracza w linii pionowej, poziomej lub ukośnej. Rozgrywka kończy się, jeżeli gracze zapełnią planszę, żaden z graczy nie może wykonać ruchu, jeden z graczy straci wszystkie swoje piony. Celem gry jest posiadanie większej liczby pionków niż przeciwnik.

Zarówno serwer jak i klient został zaimplementowany w języku C. Do obsługi komunikacji sieciowej wykorzystane zostały standardowe gniazda (sockets) z biblioteki sys/socket.h i netinet/in.h. Dodatkowo użyto biblioteki arpa/inet.h do konwersji adresów IP między formatami tekstowymi a binarnymi, co pozwala na łatwą manipulację adresami w protokołach IPv4. Do zarządzania wielowątkowością użyto biblioteki pthread.h. Kod wykorzystuje również standardowe biblioteki C, takie jak stdio.h, stdlib.h, string.h, unistd.h, stdbool.h oraz ctype.h do obsługi danych, alokacji pamięci, manipulacji stringami, komunikacji z konsolą, typów boolowskich i konwersji znaków. Kolorowanie tekstu w konsoli klienta zrealizowano za pomocą sekwencji ANSI escape codes. Serwer działa w trybie wielowątkowym, gdzie każda para klientów rozgrywająca partię gry jest obsługiwana w oddzielnym wątku, co zapewnia możliwość obsługi wielu gier jednocześnie. Komunikacja między klientami a serwerem opiera się o gniazda strumieniowe (TCP), zapewniając niezawodną transmisję danych.

2. Opis komunikacji pomiędzy serwerem i klientem:

Serwer nasłuchuje na określonym porcie (domyślnie 1100), oczekując na połączenia od klientów. W pętli akceptuje kolejne żądania połączeń. Po nawiązaniu połączenia z dwoma klientami, serwer uruchamia nowy wątek, który obsługuje ich rozgrywkę.

Wątek rozgrywki na serwerze:

1. Losuje symbole pionków (X i O) i wysyła je graczom.
2. Inicjuje planszę i przesyła ją obu graczom.
3. Rozpoczyna pętlę rozgrywki:
    1. Na podstawie numeru tury określa który gracz wykonuje ruch.
    2. Zaznacza na planszy ruchy które gracz może wykonać.
    3. Sprawdza czy rozgrywka dobiegła końca, jeśli tak wysyła do obu graczy komunikat „gameEnded” i kończy pętlę rozgrywki.
    4. Wysyła komunikat „oponentTurn” do gracza który nie wykonuje ruchu w tej turze.
    5. Sprawdza czy gracz może wykonać ruch w tej turze, jeśli nie wysyła do niego komunikat „noValidMoves”, a do jego przeciwnika wysyła aktualną planszę, następnie zwiększa licznik tur i przechodzi do kolejnej iteracji pętli.
    6. Wysyła do gracza planszę z zaznaczonymi ruchami które może wykonać.

7. W pętli odbiera od gracza ruch i sprawdza jego poprawność:
    a) Jeśli ruch jest poprawny wychodzi z pętli.
    b) W przeciwnym przypadku wysyła do klienta komunikat „invalidMove”.
8. Wykonuje ruch, wysyła planszę do obu graczy i zwiększa licznik tur.

4. Zamknięcie połączeń z graczami i zakończenie wątku.

Klient**:**

1. Nawiązuje połączenie z serwerem.
2. Odczytuje symbol pionków wysłany przez serwer i wyświetla go.
3. Odczytuje i wyświetla planszę.
4. Rozpoczyna pętlę rozgrywki:
    1. Odczytuje komunikat o stanie gry.
    2. Jeśli odczytany komunikat to „gameEnded” wyświetla informacje o końcu rozgrywki oraz wynik, przerywa pętlę.
    3. Jeśli odczytany komunikat to „oponentTurn” wyświetla informację o oczekiwaniu na ruch przeciwnika, a następnie odczytuje planszę po jego ruchu i wyświetla ją, przechodzi do kolejnej iteracji pętli.
    4. Jeśli odczytany komunikat to „noValidMoves” wyświetla informacje o braku możliwości wykonania ruchu i przechodzi do kolejnej iteracji pętli.
    5. Wyświetla planszę, otrzymaną w komunikacie.
    6. W pętli pobiera ze standardowego wejścia ruch gracza, jeśli jest równy „quit” wychodzi z obu pętli, wpp. wysyła ruch do serwera i odbiera odpowiedź, jeśli odpowiedź jest równa „invalidMove” wyświetla informację o niepoprawnym ruchu, wpp. wychodzi z pętli.
    7. Wyświetla planszę po wykonanym ruchu.
5. Zamyka połączenie z klientem i wyłącza program.

3. Podsumowanie:

- Wielowątkowość: Serwer jest wielowątkowy, co umożliwia obsługę wielu równoczesnych gier. Każda gra jest obsługiwana w osobnym wątku, co zapewnia skalowalność.

- Gniazda sieciowe: Komunikacja pomiędzy klientami a serwerem odbywa się za pomocą gniazd sieciowych TCP, co zapewnia niezawodny i uporządkowany przepływ danych.

- Obsługa logiki gry: Logika gry, w tym inicjalizacja planszy, zaznaczanie możliwych ruchów, sprawdzanie poprawności ruchów i aktualizacja planszy jest zaimplementowana po stronie serwera.

- Protokół komunikacyjny: Komunikacja opiera się na przesyłaniu tekstowych reprezentacji planszy oraz prostych komunikatów o stanie gry.