Wisielec

Radosław Burdziński 141195
Adrian Krzyżanowicz 141070

1. Wstęp

Tematem naszego projektu była gra – wisielec. Podstawową funkcjonalnością programu jest stworzenie konta a następnie rozpoczęcie rozgrywki z innym graczami po zalogowaniu. Możliwy jest podgląd szubienic rywali oraz podgląd tabeli wyników. Serwer został napisany w C++, natomiast klient w Javie

2. Funkcjonalność klienta

Po uruchomieniu programu pierwszy ukaże się ekran łączenia z serwerem. Podajemy tutaj adres IP serwera oraz port. Jeżeli nie uda się połączyć z serwerem wyświetlony zostanie komunikat ”Can’t reach server”, w przeciwnym razie zostaniemy przeniesieni do ekranu logowania. Tutaj możemy zalogować się na istniejące konto wpisując odpowiednie dane i zatwierdzając przyciskiem sign in. Jeżeli nie mamy konta przechodzimy do ekranu rejestracji klikając przycisk sign up, a następnie wypełniamy odpowiedni formularz. Gdy zalogowanie już się powiedzie, wówczas pokaże się główny ekran . Zaczynając od góry widzimy wyświetlony nasz nick, niżej znajduje się pole, w którym po wciśnięciu przycisku “Play” pojawi sie liczba oczekujących graczy. Jeszcze niżej znajdują się dwa przyciski “Play” – wyrażający chęć dołączenia do gry oraz “Start” – rozpoczynający rozgrywkę. Wciśnięcie przycisku “Start” możliwe jest jeżeli co najmniej dwóch graczy jest w kolejce. Jednak aby gra się rozpoczęła nie może obecnie rozgrywać się inna gra.

3. Funkcje sieciowe klienta

Klient do połączenia wykorzystuje klasę Socket. Wiadomości wysyłane są za pośrednictwem klasy PrintWriter zaś odczytywane za pomocą BufferedReader. Jeżeli będziemy chcieli połączyć się, zalogować lub zarejestrować, a serwer będzie niedostępny, to wówczas zostanie wyświetlony odpowiedni komunikat. Przy połączeniu występuje pięciosekundowy timeout.

4. Serwer

Przy uruchomieniu, stan serwera, tj. listy klientów, jest deserializowany. Administrator serwera może używać konsoli do wyświetlania bieżącej listy zarejestrowanych klientów i ich usuwania poprzez wpisywanie odpowiednich liter (odpowiednio k i d). Drugi z wątków serwera w pętli oczekuje na nowe połączenia. Po otrzymaniu połączenia tworzy nowy wątek Responder.cpp – programu obsługującego zapytania - i przypisuje go do odpowiedniego gniazda. Wątek ten w pętli odczytuje wiadomości z gniazda i odpowiednio reaguje na przychodzące kody. Aby chronić program przed niepożądanymi efektami współbieżności, globalna lista użytkowników i lista grających użytkowników oraz globalne flagi umożliwiające gre (Game::Enable) są chronione mutexami. Wątek Responder nie ma przypisanego jednego klienta – może się on zmieniać w trakcie działania wątku, tak długo jak istnieje połączenie. Gdy połączenie zostaje zerwane, Responder upewnia się, że przypisany do niego klient nie jest zalogowany i kończy działanie, a jego zasoby są zwalniane. Serwer należy wyłączyć poprzez wpisanie w konsolę litery c. Serwer uruchamia serializację poprzez klasę ServerSerializer. Po zakończeniu serializacji program jest wyłączany

5. Kody

Komunikacja miedzy klientem a serwerem opiera się na poniższych kodach:

a) 101 - logowanie

b) 102 - wylogowanie
c) 103 - rejestracja
d) 104 - dodanie punktu po zgadnięciu litery
e) 105 - obsługa odbioru informacji o błędzie gracza
f) 106 - wyslanie informacji o tym ze gracz X jest gotów do gry
g) 107 - wysylanie nowego słowa
h) 108 - sprawdzenie czy ktos jeszcze gra
i) 109 - rozpoczecie gry

Wiadomości mają przykładowo postać: 101|login|password.