ZPR - Adaptacja karcianej gry pt. "Światowy Konflikt"

Kaczmarek Kacper
Szpunar Michał

Styczeń 2020

1 Opis aplikacji

"Światowy Konflikt" jest grą multiplayer działającą w architekturze klient-serwer. Klient i Serwer są dwoma oddzielnymi programami napisanymi w ten sposób, by następowała wymiana komunikacji sieciowej pomiędzy nimi.

Zastosowanym modelem protokołu komunikacyjnego jest TCP. Klienty łączą się do serwera poprzez podanie jego adresu IP (IPv4 lub PIv6). W naszej implementacji jest to dowolny adres, który jest skierowany do komputera na port 1967.

1.1 Aplikacja kliencka

Część kliencka aplikacji jest odpowiedzialna przede wszystkiim za wygląd gry. Przechowuje ona zarówno modele interfejsów użytkownika, jak i również np. grafiki kart użytych w grze. Obsługuje ona również wysyłanie zapytań do serwera (głównie za pośrednictwem przycisków) oraz odbiór i parsowanie otrzymanych wiadomości. Wysyłanie zapytań wykonywane jest w klasie opisanej w pliku **mainwindow.h**, natomiast klasa dokonująca interpretacji otrzymanych odpowiedzi znajduje się w **chatclient.h**. Zarówno zapytania jak i otrzymywane odpowiedzi są w formacie JSON.

Komunikacja klas ChatClient oraz MainWindow odbywa się w głównej mierze za pomocą emitowania *sygnałów* i wywoływania skojarzonych do nich *slotów* - są to mechanizmy frameworku Qt. "Podpięcie" odpowiednich sygnałów pod dane sloty odbywa się w konstruktorze MainWindow.

1.2 Serwer

Serwer jest aplikacją, która przetrzymuje wszelkie informacje o zalogowanych użytkownika oraz stworzonych rozgrywkach, rozsyła do aplikacji klienckich update'y dotyczące chatu oraz gry, w której mogą przebywać. Interpretuje on również otrzymane odpowiedzi (za pomocą klas opisanych w plikach **chatserver.h** oraz **session.h**) korzystając z logiki gry (również w **session.h**). Serwer jest w stanie wysłać wiadomości do:

- wszystkich użytkowników
- użytkowników, którzy dołączyli do danej rozgrywki
- pojedynczego użytkownika

Serwer od klienta otrzymuje wiadomości

- systemowe (np. próba połączenia się, próba stworzenia gry, próba dołączenia do gry - interpretowane są za pośrednictwem obiektu klasy ChatServer)
- akcji (iterpretowane są za pośrednictwem obiektów klasy Session)
- chatu (interpretowane za pośrednictwem obiektu klasy ChatServer)

1.2.1 Sesja

Sesja jest klasą realizującą rozgrywkę. To ona rozsyła graczom wiadomości o wykonanych akcjach, komunikatach, update'ach, timeoutach itp. Zajmuje się ona również interpretowaniem wiadomości o typie *action* i *caunteraction* otrzymanych od klienta. Interpretacja takich wiadomości połączona jest z wbudowaną logiką gry, która decyduje o tym, jaka odpowiedź zostanie odesłana klientowi.

Sesja wysyła wiadomości o typie *sessionmessage*.

2 Zrealizowane funkcje opisane w dokumentacji wstępnej

2.1 Możliwość rozgrywki multiplayer

Zgodnie z opisem zawartym w rozdziale 1, gra odybywa się wyłącznie w trybie wieloosobowym. Gracze po podaniu adresu ip mogą połączyć się do serwera, a następnie stworzyć grę/ dołączyć do gry.

2.2 Możliwość czatu między graczami, portfel gracza, liczenie piniędzy

Czat między graczami jest zrealizowany w sposób wysłania przez klienta wiadomości o odpowiednim typie. Serwer interpretuje tę wiadomość, po czym rozsyła ją do wszystkich graczy zalogowanych.

Portfel gracza oraz liczba jego żyć są polami klasy Player, której obiekty agregowane są przez obiekty klasy ServerWorker - klasy reprezentującej danego klienta. Obiekt Player tworzony jest w momencie wystartowania gry.

Po skończeniu każdej tury, serwer rozsyła do klientów obecnych w danej sesji wiadomość o specjanlym typie *update* wraz z parametrami wszystkich graczy. W ten sposób klient jest w stanie dane te wyświetlić.

2.3 Czas wykonania tury, czas pomiędzy turami na 'sprawdzenie'

Realizacja odmierzania czasu zrealizowana jest z pomocą klasy QTimer, która po uplynięciu zadanego czasu potrafi wysłać sygnał, który sprawia, że wykonywany jest skojarzony slot - w naszym przypadku *callOnTimeout*. Timer jest składową klasy Session. Uruchomiony jest każdorazowo w momencie, w którym wykonana została akcja, którą można sprawdzić lub zablokować. Timer można również zatrzymać poprzez wykonanie sprawdzenia lub bloku.

2.4 Każdy użytkownik może wykonywać dowolną akcję, nawet jeśli nie posiada danej karty

Zrealizowane poprzez dostępny interfejs podczas tury, gdzie wszytskie przyciski są odblokowane. To, czy gracz posiada daną kartę, liczy się jedynie podczas wysłanie rządania sprawdzenia przez innego gracza.

2.5 Przeglądanie, jakie karty mogą daną akcję zablokować

Po kliknięciu przycisku "Blokuj" pojawia się okno dialogowe, które wyświetla możliwe blockery. W momencie, gdy akcja nie może być zablokowana przez wskazaną kartę, wyświetlany jest pop-up z informacją o nieprawidłowym zagraniu.

2.6 Możliwość wyjścia w każdym momencie rozgrywki

Kiedy zajdzie taka sytuacja, to dane gracza zostają usunięte z rozgrywki. Odświeżany jest również widok (poprzez update od serwera). Gra może:

- toczyć się dalej - jeśli gracz nie był właścicielem sesji
- odesłać graczy do stanu początkowego - jeśli gracz był właścicielem sesji

2.7 Lobby

Lobby jest widokiem, który pokazywany jest graczowi po dołączeniu do gry, ale zanim rozgrywka wystartuje. Odblokowany jest chat oraz możliwe jest zgłaszanie swojej gotowości poprzez przycisk "Gotowy".

2.8 Stworzenie rozgrywki

Gracz może stworzyć rozgrywkę zaraz po połączeniu się do wskazanego serwera. Akcja ta dostępna jest w Menu. Po klinknięciu przycisku "Create game", wyskoczy okno dialogowe z polem tekstowym do wpisania liczby przewidywanych graczy - warto zaznaczyć, że gra może wystartować również w momencie, w którym jest mniej graczy niż ustalony limit. Nie można natomiast dołączyć do gry, której limit graczy jest już osiągnięty.

Od momentu wysłania zapytania do serwera z liczbą graczy i otrzymaniu odpowiedzi, że udało się stworzyć sesję, gracz staje się właścicielem sesji. Może ją wystartować w dowolnym momencie.

2.9 Wybór nicku

Nick nie tylko jest możliwy - on jest kluczowy, ponieważ jest również identyfikatorem gracza. O podanie go, użytkownik proszony jest zaraz po nawiązaniu połączenia.

3 Problemy

3.1 Ustalenie relacji pomiędzy serwerem a klientem

To, jak powinna wyglądać relacja pomiędzy serwerem a klientem przez długi czas wywoływało kontrowersje w zespole projektowym. Powtawały tu różne koncepcje, które ciężko było przeforsować. Ostateczny model powstał wraz z rozwojem projektu, gdyż dopiero wtedy zaczęliśmy sobie uświadamiać, jakie kompetencje powinien mieć każdy z podmiotów. Uprzednia niewiedza była czynnikiem skutecznie utrudniającym rozwój aplikacji, gdyż dochodziło do konfliktów koncepcji, a co za tym idzie - niespójności.

3.2 Zbyt duży nakład pracy powiązny z projektem/ złe gospodarowanie czasem

Największym problemem napotkanym na drodze pisania projektu okazało się słabe zorganizowanie w zespole. Praca systematyczna na przestrzeni całego semestru zaowocowała by dużo lepszymi wynikami i lepszym efketem końcowym. Jednak duża część aplikacji powstała w grudniu/styczniu, w wyniku czego m.in. nie wszytskie akcje zostały zaimplementowane ze względu na swoją złożoność.

4 Liczba godzin poświęconych na rozwój projektu

Kacper - około 100h
Michał - około 70h