Dawid, Nowakowski 151868

Sprawozdanie sk2

Reversi – sieciowa gra turowa

1. Opis projektu (0.5 strony)

* Projekt to implementacja znanej turowej gry strategiczno-logicznej. Gracze dobieranie są kolejno w pary (nie ma możliwości wyboru „planszy”) i łączeni razem do sesji. Po podłączeniu do sesji gra się rozpoczyna i gracze naprzemiennie pytani są o ruchy. Ruchy następnie walidowane są po stronie serwera – jeżeli ruch jest możliwy to jest wykonywany (sprawdzane są warunki zakończenia gry) – stan planszy jest aktualizowany oraz przesyłany do przeciwnika. Serwer wspiera kilka rozgrywek jednocześnie. Po wykryciu rozłączenia jednego z graczy w sesji wysyłana jest wiadomość do gracza, który w sesji jeszcze pozostał.
* Do implementacji klienta wykorzystany został Python (nie ma GUI – wszystko odbywa się w terminalu). Do implementacji serwera użyty został C.

2. Opis komunikacji pomiędzy serwerem i klientem (0.5 strony, może być schemat/rysunek)

* W wątku głównym serwera zainicjalizowane są dwa liczniki – jeden to licznik graczy, drugi to licznik sesji. Gracze nie mają możliwości wybierania, do której planszy chcą dołączyć – są dobierani kolejno w pary przez serwer.
* Jeżeli serwer przyjmie nowe połączenie to sprawdza czy w obecnej sesji jest wolny którykolwiek socket. Jeżeli jest, to pierwszy gracz dołącza do sesji oraz zostaje poinformowany o tym, że musi poczekać na drugiego gracza. Jego socket jest zapisywany w strukturze, która przechowuje informacje o sesji
* Gdy kolejny gracz się połączy zostanie przydzielony do tej samej sesji – jego socket również zostanie zapisany do struktury. Po dołączeniu do gry drugiego gracza sesja się zamyka (zwiększa się licznik sesji). Gracz drugi zostaje poinformowany przez serwer, że musi poczekać na ruch pierwszego gracza – w tym momencie gracz pierwszy otrzymuje informacje z początkowym układem planszy oraz może wykonać ruch.
* Gdy serwer otrzyma ruch następuje walidacja – jeśli ruch jest możliwy to jest wykonywany – jeśli gra po tym ruchu się nie zakończyła, a przeciwnik ma jeszcze legalne ruchy, to następuje przesłanie do niego informacji ze zaktualizowaną planszą. Gdy przeciwnik nie ma legalnych ruchów to możliwość ruchu znowu dostaje gracz, który ruch wykonał przed chwilą i może wykonywać ruchy tak długo, aż zabraknie mu legalnych ruchów – wtedy gra się kończy, podliczane są punkty oraz wysyłane informacje o zakończeniu gry i wyniku.

3. Podsumowanie (0.5-1 strona)

* Zaimplementowano dwie struktury do kontrolowania informacji o graczach i rozgrywce: game\_session oraz client.
* Kierunki ruchu używane przy walidacji oraz egzekucji ruchu są przechowywane w strukturze wyliczeniowej (enum).
* Funkcja socketThread jest odpowiedzialna za komunikację między serwerem, a klientem oraz jest kanałem komunikacji między oboma graczami połączonymi do sesji.
* Każdy klient jest obsługiwany na osobnym wątku.
* Trudności:
* Największą trudność sprawiła mi asynchroniczne nasłuchiwanie i wysyłanie wiadomości przez klienta. Stworzyłem dwa osobne wątki do tego i byłem w stanie przyjąć wiadomość o rozłączeniu drugiego gracza podczas mojego ruchu, lecz nie byłem w stanie zabić wątku, który czekał na input w łagodny sposób tak, żeby program mi się nie „wysypał”
* Po stronie serwera najwięcej problemów sprawiło mi przekazywanie wartości w odpowiedni sposób (wskaźniki) oraz walidowanie ruchów