Gra sieciowa- statki

Kamil Stróżyk, 136805

1. Opis protokołu

Serwer przyjmuje i wysyła wiadomości tekstowe od/do klienta- każda wiadomość ma strukturę <słowny kod wiadomosci><(opcjonalnie) współrzędne pola planszy(x,y dla każdego pola; niektóre wiadomości mogą wysyłać kilka pól>. Wiadomości są zarówno wysyłane jak i odbierane bajt po bajcie, znakiem na zakończenie odczytu/wysłki jest znak nowej linii '\n'. W projekcie mam mutex na wysyłce. W kliencie odczyt również zachodzi bajt po bajcie. Stan gry oraz bufory do wysyłania/ odbierania wiadomości na serwerze są przechowywane w strukturze wątku- plansza gry to tablica dwuwymiarowa intów.

Tabela komunikatów oraz krótki opis:

Treść	**Argumenty**	**Wysyłany/odbierany przez serwer**	**Opis**
Player	Nazwa gracza	Odbierany	Klient wysyła swoją nazwę gracza, serwer wysyła ją przeciwnikowi
enemy:	Nazwa gracza	Wysyłany	Reakcja na komunikat player- wysłanie nazwy przeciwnika
board	Wartości poszczególnych pól planszy	Odbierany	Klient przesyła początkowy stan planszy, serwer przesyła go do klienta przeciwnika
boare	Wartości poszczególnych pól planszy	Odbierany	Klient przesyła stan planszy przeciwnika, serwer zapisuje go w swojej tablicy
start	Brak	Wysyłany	Serwer daje sygnał startowy graczom
our	Brak	Wysyłany	Sygnał tury dla klienta, klient uaktywnia swoją planszę (pozwala na nią klikać)
turn	Brak	Wysyłany	Sygnał, że przeciwnik ma turę- klient dezaktywuje swoją planszę (pozwala na nią klikać)
check	Koordynaty pola planszy	Odbierany	Użytkownik kliknął na pole w kliencie- klient wysyła komunikat do serwera, aby sprawdził jego stan
miss/ Omiss	Koordynaty pola planszy	Wysyłany	Oznaczenie pola w kliencie na szaro (miss- pole planszy przeciwnika, Omiss- pole własnej planszy w kliencie przeciwnika- analogicznie w każdym innym komunikacie tego typu)- użytkownik nie trafił- tura dla przeciwnika

hit/ Ohit	Koordynaty pola planszy	Wysyłany	Oznaczenie pola w kliencie na żółto użytkownik trafił- tura dla użytkownika, dodatkowo uruchomienie sprawdzania, czy jakiś statek został zatopiony w całości
sink/Osink	Koordynaty pola planszy	Wysyłany	Oznaczenie pola w kliencie na czerwono użytkownik zatopił cały statek
lose	Brak	Wysyłany	Gracz przegrał
won	Brak	Wysyłany	Gracz wygrał
end	Brak	Odbierany	Klient odebrał wiadomość o wygranej/porażce- serwer kończy wątek
error	Brak	Odbierany	Bład klienta- serwer kończy wątek

2. Opis struktury projektu

Serwer w pętli akceptuje połaczenia graczy- gdy jest ich dwóch, tworzy im oddzielne wątki, do których podaje ich deskryptory. Następnie, w pętli nieskończonej mamy odczytywanie wiadomości przychodzących do serwera I odpowiednia reakcja na nie, zazwyczaj wysłanie wiadomości do klienta. W tej samej pętli następuje sprawdzanie stanu gry (i wysłanie odpowiednich komunikatów po wykryciu przegranej) oraz rozpoczęcia gry.

Klient działa analogicznie- po podłączeniu się do gry również w pętli odczytuje wiadomości i reaguje na nie- przy czym tutaj reakcją jest tak naprawdę tylko zmiana koloru kafelka na planszy bądź wyświetlenie komunikatu. Poza tym, w kliencie na kliknięcie niezaznaczonego wcześniej kafelka na planszy jest wysyłana wiadomość, aby sprawdzić jego stan na serwerze.

3. Sposób uruchomienia

Zarówno do klienta, jak I do serwera dołączyłem skrypty uruchamiające w ich katalogach. W przypadku serwera to standardowe polecenie `gcc -pthread -Wall Server.cpp -o Server.out` a następnie `./Server.out` , a w przypadku klienta polecenie dołączenia modułów javafx do pliku jar przed uruchomieniem- należy podmienić w nim ścieżkę bezwzględną do javafx sdk- ja działałem na wersji 11, do pobrania stąd: *https://gluonhq.com/products/javafx/?fbclid=IwAR2f8XnRFcfVzdkT8cy_65lmGY0IRo7ER0t0lnqpQhQudMxotozwj_A5Xhs* . Klienta można też odpalić bezpośrednio z poziomu IntelliJ.