Warcaby

Jakub Borowiak

1. Temat zadania
   Tematem zadania było zaimplementowanie gry w warcaby w architekturze klient-serwer. Do współbieżnego serwera, na którym znajdują się pokoje gry, może podłączyć się wielu klientów.
2. Opis protokołu komunikacyjnego
   Klient jest obiektem klasy GameClient, która posiada atrybuty wysyłane serwerowi. Jednym z nich jest komenda. Istnieje 5 rodzajów komend wysyłanych przez klienta:
   0 – klient chce wykonać ruch
   1 – klient chce zostać przydzielony do pokoju
   2 – klient chce zrezygnować z gry
   3 – klient chce zrezygnować z ruchu, jeśli ma do tego prawo
   4 – komenda wysyłana cyklicznie w celu uzyskania stanu serwera
   Ponadto klient wysyła serwerowi swój identyfikator oraz identyfikator pokoju, do którego został przydzielony. W przypadku komendy 0 wysyła także pole początkowe oraz końcowe w postaci bajtów.
   Serwer posiada strukturę odpowiadającą atrybutom klasy GameClient. Po odebraniu wiadomości od klienta serwer wysyła odpowiedź z komendą 255 jeśli wszystko się powiodło lub 254 jeśli nie. W przypadku zalogowania/wylogowania zmieniane są także identyfikatory klienta oraz pokoju. Pole początkowe i końcowe serwer odsyła w stanie niezmienionym.
3. Opis implementacji
   Klient został zaprogramowany w języku Python z wykorzystaniem biblioteki PyQt5 do stworzenia graficznego interfejsu użytkownika. Składa się on z klasy GameClient, która posiada następujące metody:
   - __init__ (serverAddress, serverPort)- do nawiązania połączenia z serwerem
   - getMsgAsBytes() – do przygotowania wiadomości w wygodnym do wysłania formacie
   - msgFromBytes(messageBytes) – do zamiany odpowiedzi serwera na wygodny format
   - sendMsg(cmd, move_from=b"--", move_to=b"--") – do wysyłania i odbierania wiadomości
   W kliencie znajduje się także klasa obsługująca GUI:
   - __init__(), initUI(), initButton(name, left, top, function) – przygotowanie okienka, przycisków, pól tekstowych
   -onClick1() – akcja przycisku wysyłającego komendę 0
   -onClick2() – akcja przycisku wysyłającego komendę 3
   -onClick3() – akcja przycisku wysyłającego komendę 2
   -update() – wysyłanie komendy 4
   W funkcji main znajduje się inicjacja GUI oraz wywoływanie funkcji update().

Serwer został napisany w języku C++ z wykorzystaniem wątków. W nim znajdują się struktury oraz tablice przechowujące informacje o klientach, pokojach, planszach, połączeniach oraz stanie gry.
Do znalezienia klientowi pokoju i ewentualnego zaczęcia gry służy funkcja find_next_client_id().
Void *client_th(void *arg) to wątek klienta, w którym realizowane są komendy, odpowiedzi i wywoływane funkcje sprawdzające poprawność ruchów, a także dokonujące zmian w stanie gry:
-int if_valid_king(char move_f[2], char move_k[2], uint16_t c_id) – sprawdza czy podany ruch damki jest prawidłowy, zwraca 0 jeśli nie, 1 jeśli tak, 2 jeśli ruch jest z biciem
-int if_valid(char move_f[2], char move_k[2], uint16_t c_id, uint8_t figure, uint8_t plansza[64]) - sprawdza czy podany ruch piona jest prawidłowy, zwraca 0 jeśli nie, 1 jeśli tak, 2 jeśli ruch jest z biciem
- bool check_move(int ind, uint16_t c_id, uint8_t plansza[64]) – sprawdza czy pion może zbić figurę przeciwnika
- bool check_king_move(int ind, uint16_t c_id) – sprawdza czy damka może zbić figurę przeciwnika
- int check_all(uint16_t c_id) – sprawdza czy gracz ma obowiązek bicia, zwraca -1 jeśli nie lub indeks na planszy figury którą ma obowiązek bić
- bool check_all_moves(uint16_t c_id) – sprawdza czy pionki i damki danego gracza mogą się poruszyć
- bool check_end(uint16_t c_id) – sprawdza warunki końca gry
W funkcji main znajduje się nawiązywanie połączenia oraz tworzenie wątku klienta.

4. Sposób kompilacji i uruchomienia
Klient uruchamiany jest z pomocą interpretera języka Python.
Serwer kompilowany jest komendą g++ -Wall serv.cpp -o serv -pthread w systemie linux oraz uruchamiany poprzez komendę ./serv