Sieci Komputerowe 2 - Warcaby

Jakub Garus, 145241

Opis protokołu komunikacyjnego:
Dane wymieniane między klientem, a serwerem składają się z 4 znaków.
W przypadku informacji o ruchu, pierwsze dwa znaki odpowiadają położeniu pionka, natomiast dwa kolejne, odnoszą się do miejsca docelowego, na które użytkownik przesunął pionek. W sytuacji rozłączenia któregoś z graczy, do przeciwnika wysyłana jest wiadomość “LOST”.

Opis implementacji:
Aplikacja klienta:
Po wyborze pionka, którym planujemy wykonać ruch, na planszy wyświetlane są zielone oznaczenia symbolizujące dostępne, poprawne ruchy. Wybieramy pole docelowe na planszy, przy czym możemy poruszać się jedynie swoimi pionkami. Po wykonanym ruchu aplikacja czeka na ruch przeciwnika. Użycie wątków sprawia, że okno gry pozostaje aktywne w czasie oczekiwania. Jeśli któryś z graczy wygra, każdemu z nich prezentowana jest informacja o zwycięstwie lub porażce.
Aplikacja klienta składa się z sześciu plików:

- main.py - główna petla gry
- board.py - metody rysujące planszę, wyszukujące poprawne ruchy oraz wykrywające zwycięzcę
- constants.py - stałe wykorzystywane w aplikacji
- game.py - metody odpowiedzialne za wybór pola, wykonanie ruchu oraz rysowanie dostępnych, poprawnych ruchów
- network.py - funkcjonalność łączenia się z serwerem
- pawn.py - metody rysujące pionki na planszy

Aplikacja serwera:
Po uruchomieniu, serwer oczekuje na połączenie. Łączy on graczy w pary, tworząc nowy wątek oraz rozsyła informacje dotyczące przypisania użytkowników do kolorów jakimi będą grać. W dalszej części serwer oczekuje na dane z jednej strony, zaczynając od gracza czarnego, i przesyła je do przeciwnika. Po rozłączeniu się któregoś z graczy lub zwycięstwie, program wysyła odpowiednie komunikaty.

Sposób kompilacji, uruchomienia i obsługi programów:
W celu uruchomienia klienta należy przejść do folderu zawierającego plik main.py i w terminalu wpisać linię: python -u main.py. Aplikacja korzysta z biblioteki pygame, która użyta została do implementacji graficznego interfejsu użytkownika. Kolejną biblioteką, dzięki której możemy komunikować się z serwerem, jest socket.
Serwer zaimplementowany jest dla systemu GNU/Linux. Należy skompilować plik server.c:
gcc server.c -lpthread -o server
, a następnie go uruchomić:
./server