Szymon Michalak 136769

Gra sieciowa Warcaby wykorzystująca protokół TCP

1. **Opis protokołu:**
   Każdy z graczy ma podczas swojej pracy dwa wątki: jeden do wysyłania danych, a drugi do odbierania. Na samym początku, zaraz po podłączeniu i kliknięciu przycisku koloru serwer rozsyła informacje o kolorze danego gracza. Po kliknięciu przycisku „Move" dane o ruchu zostają konwertowane na wymagany format, czyli "<pole_startu>;<pole_celu>\n", np. "C3;D4\n". Informacje o ruchu jak również wszelkiego rodzaju błędy i komunikaty również mają długość 6 bajtów. Takie informacje są wysyłane do serwera, gdzie jest sprawdzana poprawność ruchu. Jeśli wszystko jest dobrze, serwer przekazuje dalej do obydwóch zawodników niezmienioną wiadomość. W przeciwnym wypadku zwraca do nadawcy numer błędu ("er;00\n"). W oknie klienta pokazuje się napis o złych polach. Dopóki jeden zawodnik nie wykona prawidłowego ruchu, drugi nie otrzymuje żadnego komunikatu i czeka na swoją kolej. Następnie serwer sprawdza stan rozgrywki. Jeśli któryś z zawodników zbije wszystkie pionki przeciwnika lub dojdzie na drugą stronę planszy wtedy każdy klient otrzymuje informację "bl;wh\n" lub "wh;bl\n". Pierwsze pole oznacza wygranego, drugie przegranego. Zostaje wyświetlona wiadomość kto wygrał, a kto przegrał. Dodatkowo, gdy jeden z graczy zamknie okno lub kliknie przycisk „Disconnect" drugi otrzymuje komunikat o zakończeniu gry "ov;er\n". Gra w takim przypadku kończy się, a zwycięzca otrzymuje gratulacje.
2. **Opis implementacji:**
   Serwer podczas łączenia zawodników w pary korzysta z mutexów i zmiennych warunkowych:
   Mutex **players** – aby nie było niejasności i problemów, proces przypisywania ID został zabezpieczony.
   Mutex **room_mut** – synchronizuje pracę w pokoju, odpowiada za bezpieczne, prawidłowe przypisane deskryptorów i połączenie ich w pary.
   Cond **wait_for_another_player** – ta zmienna warunkowa pozwala zatrzymać proces, jeśli nie mamy zawodnika do pary. Gdy pojawi się nowy zawodnik i zostanie przypisany do nas wtedy proces jest uwalniany.
   Funkcje po stronie serwera:
   - **create_board**(int[8][8]) – wypełnia macierz podaną na wejściu: 0 gdy pole jest puste lub identyfikatorem pionka.
   - **find_result**(int[8][8]) – funkcja liczy ilość pionków i zwraca tablicę 2 elementową.
   - **check_move**(int[8][8], kolor, ruch) – tutaj następuje sprawdzenie czy na danej planszy, dany kolor może wykonać ruch. Jeśli tak to zostaje zwrócona 1, a w przeciwnym wypadku 0. Tutaj również następuje aktualizacja macierzy gry.
   - **threadBehavior** – każdy podłączony klient otrzymuje swój własny wątek. To tutaj wysyłane i odbierane są wszystkie komunikaty, oczekiwanie na przeciwnika czy wywoływanie funkcji.

   Klient natomiast może wykonywać następujące operacje:
   - **receive_color** – przy pomocy tej funkcji, na początku rozgrywki pobieramy kolor, którym będziemy później sterowali.

- **receive_data** – odbiera dane od serwera i zwraca listę z odebranymi komunikatami. Jeśli wszystko odebrane jest prawidłowo lista zawiera jeden element. Jej wywołanie mieści się we wnętrzu funkcji read and update.
- **read_and_update** – uruchamiana przy wykorzystaniu nowego wątku. Tutaj natomiast znajduje się pętla while, która wykonuje się dopóki jesteśmy połącznie z serwerem. Odbiera dane, aktualizuje widok gry i wyświetla stosowne informacje.
- **show_information** – proste przetwarzanie na stringach, służące do wypisywania informacji na pseudo konsoli, która znajduje się u dołu ekranu klienta.
- **send_data** – skleja informacje z wejścia (pole startu i celu), konwertuje na ciąg bajtów i wysyła do serwera. Wywoływana przy naduszeniu przycisku „Move".

Dodatkowo takie funkcje jak conect_to_server oraz disconnect służą odpowiednio do połączenia i rozłączenia z serwerem. W dużej mierze jest użyte łapanie wyjątków, które potem wypisują odpowiedni komunikat na 'konsoli'.

3. Opis sposobu kompilacji:

Aby skompilować serwer należy w terminalu przejść do folderu z projektem, a następnie wpisać komendę:

```
gcc -pthread server.c -Wall -o ser.out
```

Po zakończonym procesie kompilacji wystarczy uruchomić plik wynikowy poleceniem:

```
./ser.out
```

Poprawnie skompilowany i uruchomiony serwer to taki, który nic nie wypisze i posiada jedynie mrugający kursor.

4. Uruchomienie klienta:

W celu uruchomienia klienta na samym początku wymagane jest pobranie biblioteki graficznej. Aby tego dokonać należy użyć polecenia:

```
sudo apt-get install python3-tk
```

Ważne, aby pliki z rozszerzeniem .png z pionkami znajdowały się w tym samym katalogu co plik klienta. Z wymagań to tyle. Następnie okno klienta można otworzyć albo z poziomu IDE (np. PyCharm) albo z konsoli wpisując:

```
python3 client.py
```

Sposób użycia aplikacji jest bardzo intuicyjny. Zaczynamy od wpisania adresu IP serwera oraz numeru portu. Serwer jest ustawiony na port numer 1234, więc taką wartość trzeba wpisać w odpowiednie okienko. Klikamy przycisk connect i oczekujemy na przeciwnika. Gdy ten się podłączy wystarczy pobrać z serwera nasz kolor. Klikamy „Set your color!". Wszelkiego rodzaju podpowiedzi czy wskazówki będą wyświetlały się w dolnej części ekranu.