Projekt protokołu sieciowego

Cel:

Projekt serwera ma za zadanie przetrzymywać pliki konfiguracyjne gry, które w przypadku wybrania gry online mają być pobrane z serwera i być zdatne do użycia w aplikacji klienta – tutaj w grze Luna Lander. Serwer posiada pliki konfiguracyjne na swój użytek, podczas wymiany informacji z klientem zostają one udostępnione do aplikacji klienckiej.

Przetrzymywane przez serwer pliki konfiguracyjne aplikacji klienckiej to:

- główny plik konfiguracyjny
- pliki zawierające definicje wyglądu poziomów gry
- plik zawierający listę najlepszych wyników

Styl wymiany informacji między hostami:

Wymiana informacji na poziomie klient/serwer zostanie obsłużona z wykorzystaniem **protokołu TCP**. Protokół TCP ustanawia dwukierunkowe połączenie pomiędzy hostami. Połączenie to musi być relacyjne, to znaczy że hostowie wymieniają się informacjami.

Funkcjonalność tego protokołu pasuje do naszej aplikacji klient/serwer, ponieważ między tymi indywidualnymi instancjami będą musiały zajść pewne porozumienia („handshaking"). Klient musi połączyć się z serwerem, a serwer musi zaakceptować to połączenie. Klient wysyła żądanie (request), a serwer odsyła odpowiedz (response) - jest to zatem połączenie dwukierunkowe, podczas którego zachodzi wymiana informacji.

- Informacje wymieniane między klientem, a serwerem są protokołami binarnymi.
- {[CONFIGURATIONS]} – plik o formacie *.json
- {[LEVEL_MODEL_X]} - plik o formacie *.json
- {[RESULTS]} – plik o formacie *.txt

- Każdy z powyższych plików jest sparsowany i zapisany jako pojedynczy obiekt klasy String. Przesyłane do klienta są serializowane obiekty typu String.

Schemat przykładowego połączenia:

Podczas opisywania przykładowego połączenia będziemy się posługiwać następującym schematem relacyjnym:
S – serwer

K – klient

np. K: ... -> S (Klient wysyła zapytanie do serwera)

Po dwukropku znajduje się krótki opis przeprowadzonego działania (request/response) natomiast kierunek strzałki będzie oznaczał od kogo do kogo jest przeprowadzane dane działanie.

❖ Nawiązanie połączenia

Po nawiązaniu połączenia przy pomocy protokołu TCP, klient wysyła informację(request) do serwera.

K: *CONNECT* -> S

a) Serwer, po otrzymaniu informacji, jeżeli akceptuje połączenie odsyła odpowiedź(response) o przesłaniu, iż użytkownik został połączony.

S : *CONNECTED* -> K

Po nawiązaniu połączenia między hostami, możliwa jest dalsza wymiana informacji między nimi – pobranie plików konfiguracyjnych z serwera.

❖ Pobranie głównego pliku konfiguracyjnego:

W celu pobrania głównego pliku konfiguracyjnego klient przesyła informację do serwera(request):

K: *GET_CONFIG* -> S

- Serwer akceptuje przychodzące żądanie i odsyła(response) podstawowe ustawienia gry:

S: {[CONFIGURATIONS]} -> K

❖ Pobranie pliku opisującego wygląd poziomu X:

W celu pobrania pliku opisującego wygląd poziomu X klient przesyła informację do serwera(request):

K: *GET_LEVEL_MODEL_X* -> S

- Serwer otrzymał żądanie, następnie odsyła do klienta model poziomu X (response):

S: {[LEVEL_MODEL_X]} -> K

❖ Pobranie listy wyników:

W celu pobrania pliku z aktualną listą wyników, klient przesyła informację do serwera(request):

K: *GET_RESULTS* -> S

- Serwer akceptuje żądanie i odsyła listę wyników do klienta(response):

S: {[RESULTS]} -> K

❖ Wysłanie wyniku do serwera

Jeżeli użytkownik podał swój ‘nick’ po zakończonej grze, to do tej informacji zostanie dołączony **uzyskany wynik**(liczba). Taka pojedyncza informacja zostanie zapisana jako obiekt klasy String : (**nick** + **uzyskany wynik**) i zostanie wysłana do serwera. Lista wyników zostaje zaktualizowana po stronie serwera.

K: SEND_ NICK_RESULT -> S

a) Serwer zaakceptował żądanie i przyjął informację/wynik (String). Odsyła informację o prawidłowym otrzymaniu danych.

S: RESULT_RECEIVED -> K

b) Serwer nie zaakceptował żądania (przesłanych danych) – wynik nie zostanie zapisany na serwerze. Serwer odsyła informację odmowie zapisu wyniku.

S: RESULT_SAVE_REJECTED -> K

- W przypadku wystąpienia jakiegokolwiek błędu po stronie serwera, serwer przesyła informację o zamknięciu połączenia:

S: CONNECTION_CLOSING -> K

- W przypadku wystąpienia błędu, który sterminował serwer lub go zawiesił zanim zdążył odesłać informację (np. dane o pliku konfiguracyjnym w postaci Stringa) lub informacji o zamknięciu połączenia – klient będzie miał ustalony czas oczekiwania na odpowiedź od serwera.