Protokół sieciowy - projekt:

Typ: Protokół sieciowy

- *Klient nawiązuje połączenie z serwerem (wysyła żądanie o zalogowanie się):*
  C: login\n=> S

  Przykład wiadomości jaką otrzyma serwer:
  a) From client: loginUser
  loginUser - wiadomość, dzięki której serwer wie, że klient chce nawiązać z nim połączenie

- *Serwer potwierdza status zalogowania się klienta(int)*
  S: loggedStatus\n=> C

  Przykłady odpowiedzi, jakie otrzymuje klient:
  a) *From server: loggedIn(1) ; 1-oznacza poprawne logowanie (klient może wysyłać prośby do serwera o mapy, listę najlepszych wyników, itd.)*
  b) *From server: timeout(3) ; 3-oznacza przekroczony timeout*
  c) *From server: error(0) ; 0-oznacza inny błąd, np nieoczekiwane zerwanie połączenia*

  Opis działania: Serwer potwierdzi klientowi zalogowanie tylko wtedy, kiedy doszło do tego w odpowiednio krótkim czasie(Timeout następuję po 3 sekundach), oraz nie nastąpiło nieoczekiwane zerwanie połączenia.

- *Serwer informuje klienta o słabym połączeniu i gdy jakość połączenia się nie pogorszy w określonym czasie (int) to serwer rozłącza się z klientem*
  S:badConnection\n=>C

  Odpowiedź, którą otrzymuje klient:
  a) From server: ConnectionError-10; 10-oznacza czas po jakim serwer rozłączy klienta

  Opis działania: Gdy klientowi nie uda się poprawić jakości łącza, to serwer go rozłączy i klient nie będzie miał możliwości pobierania danych z serwera

- Klient wysyła żądanie o listę 5 najlepszych wyników (String mapName):
  C: requestForTop5Ranking\n => S

gdzie:
mapName - jest nazwą mapy dla której żądamy 5 najlepszych wyników

Przykład wiadomości jaką otrzyma serwer:

b) From client: get5BestPlayers-level1
get5BestPlayers - wiadomość, dzięki której serwer wie, że ma przesłać listę najlepszych wyników
level1 - oznacza numer mapy, dla której klient chce otrzymać listę najlepszych wyników

- Serwer wysyła do klienta odpowiedź z listą najlepszych wyników jako dane w postaci String[] nicks, Double[] scores
S: replyForTop5Ranking\n =>C

gdzie:
nicks - nazwy graczy
score - uzyskany wynik danego gracza

Przykładowe dane otrzymane przez klienta:

a) From server: nicks; scores
String[] nicks = {Asia,Basia,Marysia,Dobromir,Genowefa}
Double[] scores ={623,546,324,123,89}

Opis działania: Klient otrzymuje 2 tablice z danymi do żądanej mapy, jedna odnosi się do nicków graczy, a druga do wyników.

- Klient wysyła żądanie o listę dostępnych map:
C: requestForNumberOfMaps\n =>S

Przykład wiadomości jaką otrzyma serwer:

c) From client: getAvailableMaps

Opis działania: Na podstawie tej wiadomości serwer sprawdza ilość dostępnych map w swojej bazie danych

- Serwer odpowiada klientowi przesyłając dane na temat liczby dostępnych map(int)
S:replyNumberOfMap\n=>C

Przykład wiadomości jaką otrzyma klient:

a) From server: availableMaps-5; 5-oznacza liczbę dostępnych map na serwerze

- Klient wysyła żądanie do serwera o dane na temat konkretnej mapy (String mapName)
  C: requestForNumberOfMaps\n =>S

  gdzie:
  mapName - jest nazwą mapy dla której żądamy listy przeciwników i czas ich spawnu

  Przykład wiadomości jaką otrzyma serwer:
  a) From client: sendMap-level2; level2-oznacza nazwę mapy, na której chce zagrać klient

- Serwer wysyła do klienta odpowiedź z danymi mapy w postaci String[] enemies, Double[] spawnTime
  S:replyForMapData\n=>C

  gdzie:
  enemies - nazwy przeciwników
  spawnTime - czas spawnowania poszczególnych przeciwników

  Przykładowe dane otrzymane przez klienta:
  a) From server: nicks; scores
     String[] enemies = {SmallBall, SmallBall, LargeBall, MegaBall, LargeBall}
     Double[] spawnTime ={2,5,8,10,14}

     Opis działania: Klient otrzymuje 2 tablice z danymi do żądanej mapy, jedna odnosi się do nazw przeciwników, a druga do czasu ich spawnowania na mapie

- Klient wysyła żądanie o listę configów dla wybranego poziomu trudności (int difficulty):
  C: getConfig\n=>S

  gdzie:
  difficulty - poziom trudności (od 1 do 3, gdzie 1 - łatwy, 2 - średni, 3 - trudny)

  Przykład wiadomości jaką otrzyma serwer:
  a) From client: sendLevelConfigForDifficulty-2; 2-oznacza poziom trudności

- Serwer odpowiada klientowi przesyłając dane konfiguracyjne dla wybranego poziomu trudności w postacii String[] attributes, Double[] values
  S: replyForConfigForDifficulty\n=>C

gdzie:
attributes - to nazwa parametru np. życie przeciwnika, lub damage jaki zadaje
values - odpowiadająca danemu parametrowi wartość

Przykład wiadomości jaką otrzyma klient:

a) From server: attributes, values
   String[] attributes = {health,damage,speed,height,width,score}
   Double[] values ={10,1.5,2,39.8,27.4,7}

Opis działania: Klient otrzymuje 2 tablice z danymi, które reprezentują dane konfiguracyjne dla wybranego poziomu trudności

- Po zakończeniu rozgrywki klient wysyła żądanie o zapis nicku gracza i jego wyniku uzyskanego na konkretnej mapie (String mapName) (String nick) (int score): Parametry są rozdzielone myślnikiem
  C: savePlayerScore\n => S

  gdzie:
  mapName - nazwa mapy, dla której gracz uzyskał dany wynik
  nick - nick gracza podany na początku gry
  score - wynik gracza po skończeniu gry

  Przykład wiadomości jaką otrzyma serwer:

  a) From client: saveScore-level5-Janek-312

  Opis działania: Serwer otrzymuje dane, które zapisuje w swojej bazie danych.

Opis działania serwera i klienta:

Zasada działania protokołu:
- serwer tworzy socket i czeka na połączenie od klienta
- klient inicjuje połączenie z serwerem (adres –> zapisany w plikach konfiguracyjnych klienta, port -> dynamicznie przydzielany w momencie odpalania aplikacji / zapisany w configu)
- serwer akceptuje połączenie od klienta
- aby móc obsłużyć nowych klientów serwer tworzy nowy socket

Działanie klienta:
- po odpaleniu gry użytkownik może połączyć się z serwerem, gdy ten jest online
- klient łączy się z serwerem i w zależności od wyboru użytkownika wysyła żądanie o: mapy / listę najlepszych wyników
- gdy gra się zakończy, a użytkownik jest połączony z serwerem to klient wysyła żądanie do serwera, aby ten zapisał wynik gracza do swojej bazy danych

Działanie serwera:
- serwer jest w stanie obsłużyć dowolną liczbę klientów
- serwer przechowuje lisy najlepszych wyników dla danych poziomów, a także nowe poziomy
- adres serwera jest zapisany w aplikacji klienta
- łączenie poszczególnych klientów -> wielowątkowość (aby serwer mógł obsługiwać wielu klientów jednocześnie i oni nie czekali na odpowiedź w nieskończoność)
-serwer przechowuje dane na temat najlepszych wyników dla każdej mapy

Łączenie klienta z serwerem:
- klient może zainicjować połączenie z serwerem
- jest określony maksymalny czas na zainicjowanie połączenia:
    - jeśli zostanie przekroczony to użytkownik dostaje informację o braku połączenia i gra w trybie offline, używając tylko map lokalnych i własnych list wyników (tylko z rozgrywek na danym komputerze)
    - jeśli użytkownik połączy się z serwerem w określonym czasie to wysyła do niego prośbę o przesłanie map oraz list najlepszych wyników do każdej mapy (gramy na mapach serwerowych)
- po skończeniu gry klient przesyła do serwera wynik i nick użytkownika, żeby serwer zapisał go w swojej bazie danych

Przesyłanie informacji:

a) klient -> serwer (klient wysyła informacje do serwera)

- po przejściu w tryb online klient wysyła prośbę do serwera o przesłanie danych na temat dostępnej ilości map i list najlepszych wyników
- po zakończeniu gry, klient wysyła żądanie do serwera, aby ten zapisał nick użytkownika i jego wynik

b) serwer ->klient (serwer wysyła informacje do klienta)

- serwer przesyła mapy / wyniki na prośbę klienta
- po każdym wejściu w listę „highscores” następuje przesłanie aktualnych list wyników z serwera do klienta

Wczytywanie map z serwera:

- po przesłaniu danych na temat ilości map użytkownik wybiera, na której mapie chce zagrać
- po kliknięciu na daną mapę klient wysyła żądanie do serwera, aby ten przesłał mu dane tej mapy (przeciwnicy + czas ich spawn'u)
- po wczytaniu mapy użytkownik jest przenoszony do panelu z grą, gdzie może pograć na wczytanej wcześniej mapie

Wczytywanie list najlepszych wyników z serwera:

-lista wyników jest pobierana po każdym kliknięciu na wynik danej mapy, w celu pobrania najbardziej aktualnych danych
- aplikacja klienta wczytuje dane i wyświetla na ekranie kilka najlepszych wyników dla wybranej mapy