Protokół sieciowy - projekt:

Typ protokołu: Tekstowy

- Klient wysyła żądanie o listę 5 najlepszych wyników (String mapName):
  C: getTop5Ranking ***mapName***\n => S

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

  Przykład wiadomości jaką otrzyma serwer:
  a) From client: getTop5Ranking ***level1***
     getTop5Ranking - wiadomość, dzięki której serwer wie, że ma przesłać listę 5 najlepszych wyników
     level1 - oznacza numer mapy, dla której klient chce otrzymać listę najlepszych wyników

- Serwer wysyła do klienta odpowiedź z listą najlepszych wyników (String nick) (int score)
  S: replyForTop5Ranking **nick1 score1 …. nick5 score5**\n =>C

  gdzie:
  nick1 ... nick5 - nazwy graczy
  score1 … score5 - uzyskany wynik danego gracza

  Przykładowe dane otrzymane przez klienta:
  a) From server: replyForTop5Ranking **Asia 623 …. Genowefa 89**

  Opis działania: Klient otrzymuje dane rozdzielone spacją, w formacie nick-wynik.

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

  Przykład wiadomości jaką otrzyma serwer:
  a) From client: getNumberOfAvailableMaps

  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 maps)
  S: numberOfAvailableMaps ***maps***\n => C

Przykład wiadomości jaką otrzyma klient:

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

- Klient wysyła żądanie do serwera o dane na temat konkretnej mapy (String mapName)
  C: sendMap ***mapName***\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 (String enemy) (int spawnTime)
  S:replyForSendMap **enemy1 spawnTime1.... enemyN spawnTimeN**\n=>C

  gdzie:
  enemy1 … enemy n - nazwy przeciwników, którzy będą się spawnować
  spawnTime1 - spawnTime n - czas spawnowania poszczególnych przeciwników

  Przykładowe dane otrzymane przez klienta:

  a) From server: replyForSendMap **SmallBall 2.... LargeBall 14**\n=>C

     Opis działania: Klient otrzymuje dane rozdzielone spacją, w formacie przeciwnik-czas spawnu.

- Klient wysyła żądanie o listę configów dla wybranego poziomu trudności (int difficulty):
  C: getConfigForDifficulty ***difficulty***\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: getConfigForDifficulty ***2***; 2-oznacza poziom trudności

- Serwer odpowiada klientowi przesyłając dane konfiguracyjne dla wybranego poziomu trudności w postacii: (double health), (double damage), (double speed), (double height), (double width), (double score)

S: sendConfigForDifficulty ***health damage speed height width score***\n=>C

gdzie:
health - życie gracza
damage - obrażenia jakie zadaje gracz przeciwnikowi
speed - szybkość poruszania się gracza
height - wysokość gracza
width - szerokość gracza
score - mnożnik punktów dla danego poziomu

Przykład wiadomości jaką otrzyma klient:

a) From server: sendConfigForDifficulty ***10 1.5 2 39.8 27.4 7***

Opis działania: Klient otrzymuje ciąg znaków rozdzielonych spacjami, 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 spacją
  C: savePlayerScore ***mapName nick score***\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: savePlayerScore ***level5 Janek 312***

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

- Serwer odpowiada klientowi, że zapisał dane na temat jego wyniku (String mapName)
  S:scoreSaved **mapName**\n => C

  gdzie:
  mapName - nazwa mapy, dla której został zapisany wynik

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