Dokumentacja projektu
Tanks 2D

Autorzy: Szymon Szostak, Dorota Zaremba

Spis Treści

Informacje wstępne

Opis protokołu dotyczy gry Tank 2D, stworzonej w ramach przedmiotu “Programowanie Aplikacji Sieciowych”.

Celem gry jest zabicie wszystkich przeciwników zanim oni zabiją ciebie. Każdy gracz zaczyna z taką samą, określona liczbą punktów życia.

Features

- Tworzenie pokojów (na nowym porcie)
- Invade po kodzie
- Wysyłanie i odbieranie na oddzielnym wątku
- Możliwość zmiany nicku w (nawet trakcie gry)
- Rejestracja przy pierwszym podłączeniu
- Przeglądanie dostępnych gier listwowanych po nicku graczy
- Szyfrowanie z użyciem ssl
- Powolne tracenie życia po rozłączeniu się, do momentu ponownego podłączenia lub ŚMIECI
- Zajęte miejsce po końcu gry
- SUPER muzyka w tle, której nie da się wyłączyć!
- Skonfigurowane CD (continuous deployment)
- Limit możliwych gier w zależności od udostępnionych portów
-

Serwer jest serwerem zdarzeniowym z szyfrowaniem dwustronnym.

Gra

W grze mamy dwie klasy obiektów:

Tank

- ilość życia
- pozycja x, y
- kąt obrotu
- nickname gracza

Bullet

- pozycja x, y
- kąt obrotu

W momencie gdy Bullet koliduje z graczem (Tank), traci on zdrowie, a Bullet znika. Po rozłączeniu się gracza, zaczyna on powoli tracić zdrowie, do momentu, aż nie wróci do gry lub nie zabraknie mu zdrowia.

Format pakietów i konwencje

Standard message

Typ wiadomości odpowiedzialny za zarządzanie.

Klient (ogólny format)

```
Command:Register\n\rAuth:uuid\n\rData:<data>\n\r\n\r
```

- Auth: kod uuid klienta
- Command: komenda oznaczająca jaka procedura musi zostać wykonana
- <data> w to miejsce wpisane zostaną opcjonalne dodatkowe dane do przesłania

Dostępne komendy

REGISTER

Client:

```
Command:Register\n\r\n\r
```

Server response:

```
{“code”:200,”message”:'Success',”data”:uuid}
```

START_CHANNEL

Client:

```
Command:START_CHANNEL\n\rAuth:uuid\n\r\n\r
```

Server response:

```
{“code”:200,”message”:'Success',”data”:join_code}
```

```
{“code”:400,”message”:”Fail”}
```

JOIN_CHANNEL

Client:

```
Command:START_CHANNEL\n\rAuth:uuid\n\rData:<game_key>\n\r\n\r
```

- <data> to kod dołączenia do gry, wyświetlany w lewym górnym rogu np. `abcd`

Server response:

Server response:

```
{“code”:200,”message”:'Success',”data”:game_port}
```

```
{“code”:400,”message”:”Fail”}
```

game_port np.

```
40420
```

LIST_GAMES

Client:

```
Command:LIST_GAMES\n\rAuth:uuid\n\r\n\r
```

Server response:

```
{“code”:200,”message”:'Success',”data”:list_of_games}
```

list_of_games np.

```
[['nickname', 'abcd'], ['player2', 'abxd']]
```

Status message

Typ wiadomości odpowiedzialny za przesłanie statusu od serwera.

Serwer

```
{"code": code, "message": message, "data": data}\n\r\n\r
```

- code: kod z jakim zakończyła się procedura
  - 200 - OK
  - 400 - Fail/Error
- message: dodatkowa informacja opisująca błąd
- data: dodatkowe dane

Game message

Typ wiadomości odpowiedzialny za przesyłanie akcji gracza i odpowiedzi serwera.

Klient

```
nickname,uuid,<game_data>\n\r\n\r
```

- nickname to
- uuid otrzymane podczas rejstracji
- <game_data> to tablica Booli w której wpisane są dane dotyczące naciśniętych klawiszy. Kolejne indeksy to kolejne klawisze

[K_UP, K_DOWN, K_LEFT, K_RIGHT, K_ESCAPE, K_SPACE]

Serwer

```
<game_data>\n\r\n\r
```

- <game_data> w to miejsce zostaną wpisane dane dotyczące pozycji obiektów w grze lub wiadomości o zajętym miejscu w przypadku śmierci lub wgranej

tablica obiektów gry np.

```
[{
            "type": "player",
            "nickname": entity.nickname,
            "centerx": entity.rect.centerx,
            "centery": entity.rect.centery,
            "angle": entity.angle,
            "health": entity.health
  },
{
            "type": "bullet",
            "centerx": entity.rect.centerx,
            "centery": entity.rect.centery,
            "angle": entity.angle
}]
```

W momencie zakończenia gry lub przegranej gracza serwer wysyła do niego informacje

```
[GAME_OVER,<player_place>]
```

<player_place> to string z zajętym miejscem np. `1st, 2nd`

Krótka instrukcja

Rozszerzona instrukcja znajduje się w naszym repozytorium.

Hostowanie gry

Po wciśnięciu przycisku Host game tworzymy pokój do którego będą mogli dołączyć inni gracze.

Dołączenie do Gry

Kod gry może zostać podany ręcznie lub wybrany z listy dostępnych pokoi.

Po wpisaniu prawidłowego kodu automatycznie dołączamy do gry.

Po wybraniu kodu z listy i naciśnięciu enter, dołączamy do gry

Poruszanie się i strzelanie

Poruszamy się za pomocą strzałek, a strzelamy spacją.

Śmierć

Jeżeli gracz straci wszystkie punkty życia, umiera. :)

Zwycięstwo/Zakończenie gry

Jeżeli umrą wszyscy oprócz jednego gracza, gra się kończy. Każdy gracz dostaje informacje o zajętym miejscu, zaraz po śmierci lub końcu gry.

Wyjście z gry

Jeżeli klient zamknie okno gry (X w prawym górnym rogu), wychodzi z danej gry i wraca do okna menu gry. Po upływie paru sekund socket rozłącza się, a gracz zaczyna powoli tracić zdrowie, do momentu aż nie połączy się ponownie lub zabraknie mu zdrowia.

Zakończenie

W projekcie skonfigurowaliśmy CD, więc przy każdym mergu na main’a na publiczny serwer wgrywa się najnowsza wersja serwera. Do publicznego serwera każdy może się podłączyć.