Tanks - sprawozdanie

Stanisław Nieradko Bartłomiej Krawisz Jakub Bronowski

1. Wprowadzenie

Celem projektu było stworzenie gry w języku Python, która będzie symulacją bitwy czołgów obsługującą kilku graczy. Gracze sterują czołgami, które poruszają się po planszy i strzelają do siebie. Gra kończy się gdy wszyscy poza 1 graczem zostaną zniszczeni. Gracz, który pozostał na planszy wygrywa.

2. Model Komunikacji

Gra wykorzystuje protokół TCP do komunikacji między klientami a serwerem. Wybór ten został dokonany ze względu na wystarczającą wydajność względem UDP oraz prostotę implementacji.

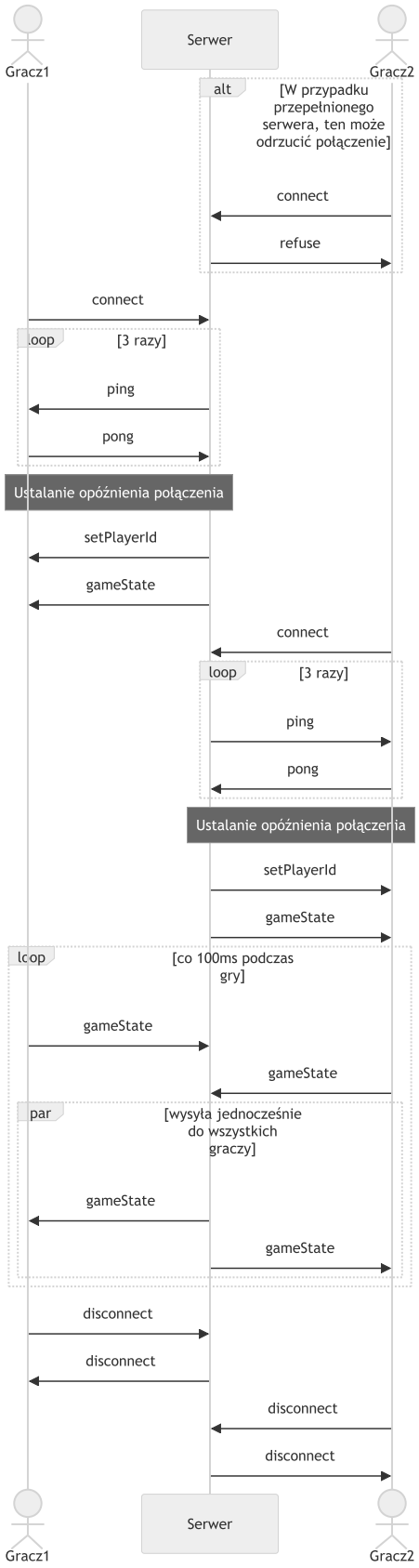
Zdarzenie przesyłane z oraz do serwera są w formacie JSON, wg. poniższego schematu:

Nazwa	Typ	Opis
`event_type`	string	Typ zdarzenia (`connect`, `refuse`, `ping`, `pong`, `setPlayerId`, `gameState`, `disconnect`)
`time`	uint64	Czas zdarzenia [unix timestamp]
`data`	object	Dane zdarzenie (zależne od `event_type`)
	char	NULL kończący wiadomość

Przykładowe zdarzenia:

```
{"eventType":"connect","time":1711291958}

{"eventType":"refuse","time":1711291958}

{"eventType":"ping","time":1711293829}

{"eventType":"pong","time":1711295934,"data":{ "serverTime":1711293829 }}

{"eventType":"setPlayerId","time":1711301024,"data": 2}

{"eventType":"gameState","time":1711293829,"data": {
    "tanks": {
        "1": { "x": 115.2, "y": 254.32, "direction": 1, "speed": 100, "score": 50,
"alive": true },
        "2": { "x": 34.2, "y": 74.32, "direction": 0, "speed": 0, "score": 0,
"alive": false }
    },
    "bullets": [
        { "x": 115.2, "y": 254.32, "direction": 1, "speed": 100, "playerId": 2 },
        { "x": 34.2, "y": 74.32, "direction": 0, "speed": 0, "playerId": 1 }
    ],
    "isGameOver": false
}}

{"eventType":"disconnect","time":1711895043}
```

2.2 Diagram Sekwencji



Źródło: mermaid.live

2.3 Schemat działania aplikacji

Gracze dołączają do serwera poprzez wysłanie zdarzenia `connect`. Po 3 krotnej wymianie zdarzeń `ping` (ze strony serwera) oraz `pong` (ze strony użytkonika) w celu ustalenia opóźnienia (z ang. *latency*) połączenia. Po ustaleniu opóźnienia serwer odpowiada zdarzeniem `setPlayerId` z przypisanym identyfikatorem gracza oraz zdarzeniem `gameState` z aktualnym stanem gry.

W przypadku przepełnienia serwera, serwer wysyła zdarzenie `refuse` i zamyka połączenie.

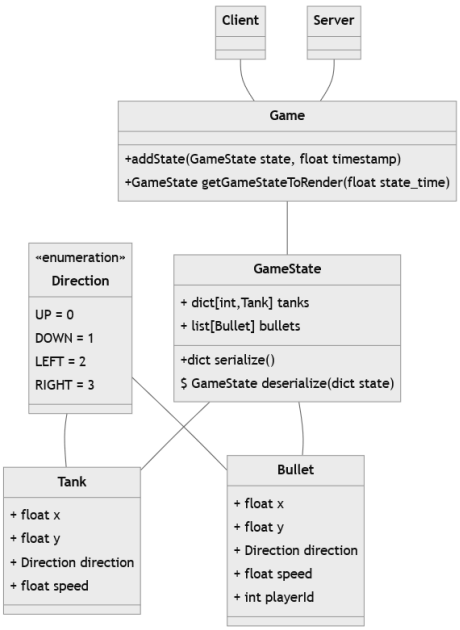
Podczas gry co 100ms zarówno serwer wysyła każdemu graczowi zdarzenie `gameState` z aktualnym stanem gry, jak i każdy gracz stara się wysłać zdarzenie `gameState` z aktualnym stanem swojego czołgu.

Gdy gra kończy się (po zniszczeniu wszystkich graczy poza jednym), serwer wysyła zdarzenie `gameState` z `isGameOver` ustawionym na `true`, co informuje klientów iż gra się zakończyła. Po 3s serwer rozpoczyna nową grę.

W celu rozłączenia się z serwerem gracz wysyła zdarzenie `disconnect` (w wyniku którego otrzymuje on także zwrotne zdarzenie `disconnect` od serwera w celu potwierdzenia).

W przypadku zakończenia pracy serwera, zostaje nadaane zdarzenie `disconnect` dla każdego gracza.

2.3 Diagramy klas



Źródło: mermaid.live

3. FAQ

TCP vs UDP
Po przetestowaniu obu protokołów, zdecydowaliśmy się na TCP ze względu na mniejszą ilość problemów związanych z przesyłem danych oraz wystarczającą wydajność do naszych zastosowań.

Mimo, iż UDP jest szybszy, to implementacja gry w tym protokole wymagałaby dodatkowego nakładu pracy. W przypadku TCP, odnotowane opóźnienia względem implementacji korzystającej z UDP były na tyle małe, że nie miały one wpływu na rozgrywkę.

W jaki sposób radzimy sobie z sytuacją w której klient, przestał przesyłać informacje?
W przypadku gdy klient przestaje przesyłać informacje, serwer po 5s od ostatniego zdarzenia wysyła zdarzenie `disconnect` w celu rozłączenia klienta. W momencie wysyłania zdarzenia czołg klienta zostaje natychmiast usunięty z gry.

W międzyczasie serwer będzie "symulował" zachowanie czołgu poprzez ekstrapolację przez co najwyżej 0,5s. Po tym czasie czołg klienta zatrzyma się w miejscu.

W jaki sposób radzimy sobie z sytuacją w której serwer, przestał przesyłać informacje?
Klient symuluje zachowania czołgów innych graczy dzięki ekstrapolacji (przez maks. 0,5s). Jeżeli do tego czasu nie uda się przywrócić połączenia z serwerem, klient zakończy grę (lokalnie) i pokaże komunikat o problemach z połączeniem.

W jaki sposób radzimy sobie z opóźnieniami przesyłu i/lub zakolejkowanymi wiadomościami?
Serwer wykorzystuje asynchroniczny model przetwarzania zdarzeń, co pozwala na obsługę kolejnych zdarzeń podczas wysyłania i oczekiwania na ukończenie połączenia. Wszystkie zdarzenia są przetwarzane w kolejności ich otrzymania, także w przypadku opóźnień po stronie klienta, tak szybko jak nadejdą opóźnione zdarzenia, zostaną one obsłużone.

Podczas gry, gdy zdarzenia klienta nie dochodzą do serwera, ten ekstrapoluje ruch czołgu (przez 0,5s) klienta na podstawie ostatnich danych otrzymanych od klienta i przesyła je do pozostałych graczy. Po oknie ekstrapolacji ale przed wyrzuceniem gracza z powodu nieaktywności, czołg klienta nie porusza się. W przypadku gdy klient znów zacznie przesyłać dane, serwer cofnie czołg klienta do pozycji sprzed maks. 0,5s i zaakceptuje dane od klienta tak długo jak nie są one starsze niż 0,5s. W innym wypadku pozycja czołgu klienta będzie taka sama jak po cofnięciu.

W jaki sposób werfyikujemy, że dane zostały przesłane w całości?
Dzięki wykorzystaniu JSON'a jako medium przesyłu danych, serwer jest w stanie sprawdzić czy dane zostały przesłane w całości. W przypadku błędu w przesłaniu danych, serwer odrzuca zdarzenie i oczekuje na kolejne. Każda wiadomość musi być zakończona znakiem `NULL` w celu odseparowania poszczególnych JSON'ów.

W jaki sposób radzimy sobie z sytuacją gdy pakiet "zagubi się" i nie dotrze poprawnie do/z serwera?
TCP zapewnia, że dane zostaną dostarczone w całości i w odpowiedniej kolejności. W przypadku gdy pakiet zostanie "zagubiony", TCP ponownie wyśle dane. W przypadku gdy pakiet nie dotrze do serwera, klient ponownie wyśle dane.

Nawet gdy w magiczny sposób było to możliwe lub dokonalibyśmy zmiany na protokół UDP, to i tak dane przesyłane są na tyle często (co 100ms) że nie ma to wpływu na rozgrywkę.