Projekt z przedmiotu Przetwarzanie Rozproszone

13 Maja 2024 r.

Bartosz Skorowski | Kamil Raubo | Jakub Bot

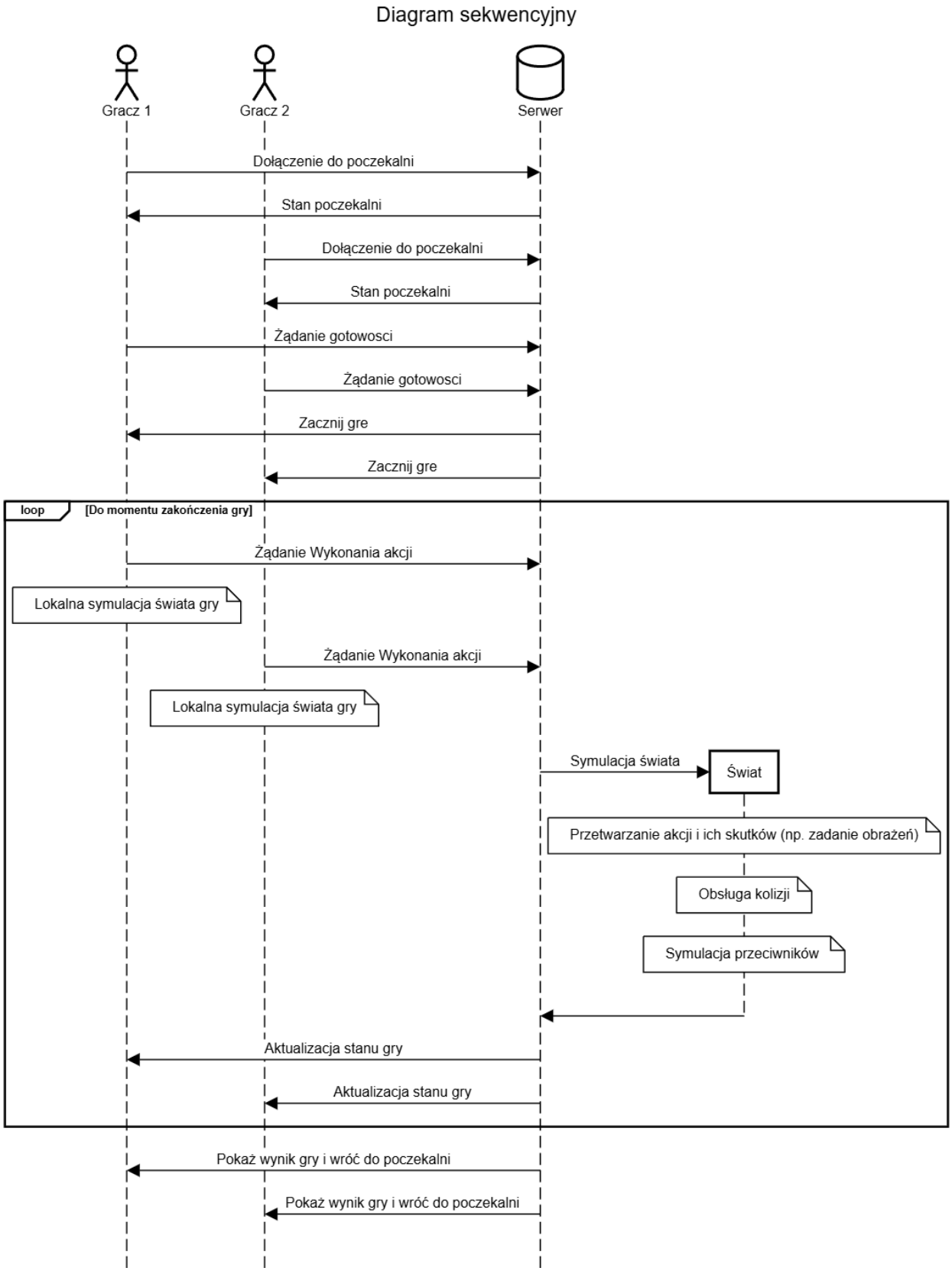
1. Wprowadzenie

Stworzona na potrzeby części drugiej projektu gra jest typu “Tower Defense”. Gracze wspólnie bronią bazy przed falami przeciwników (AI). Przeciwnicy tworzą się określonych na mapie punktach i kierują się na bazę, której zadają obrażenia, gdy się do niej wystarczająco zbliżą. W celu pokonania przeciwników gracze mogą strzelać kulami ognia zadającymi obrażenia przeciwnikom. Gra kończy się zwycięstwem w momencie pokonania wszystkich przeciwników i porażką w momencie, gdy przeciwnicy zniszczą bazę graczy. Serwer jest autorytarny - cała realna gra rozgrywa się na serwerze. Gracze wykonują lokalne symulacje świata gry.

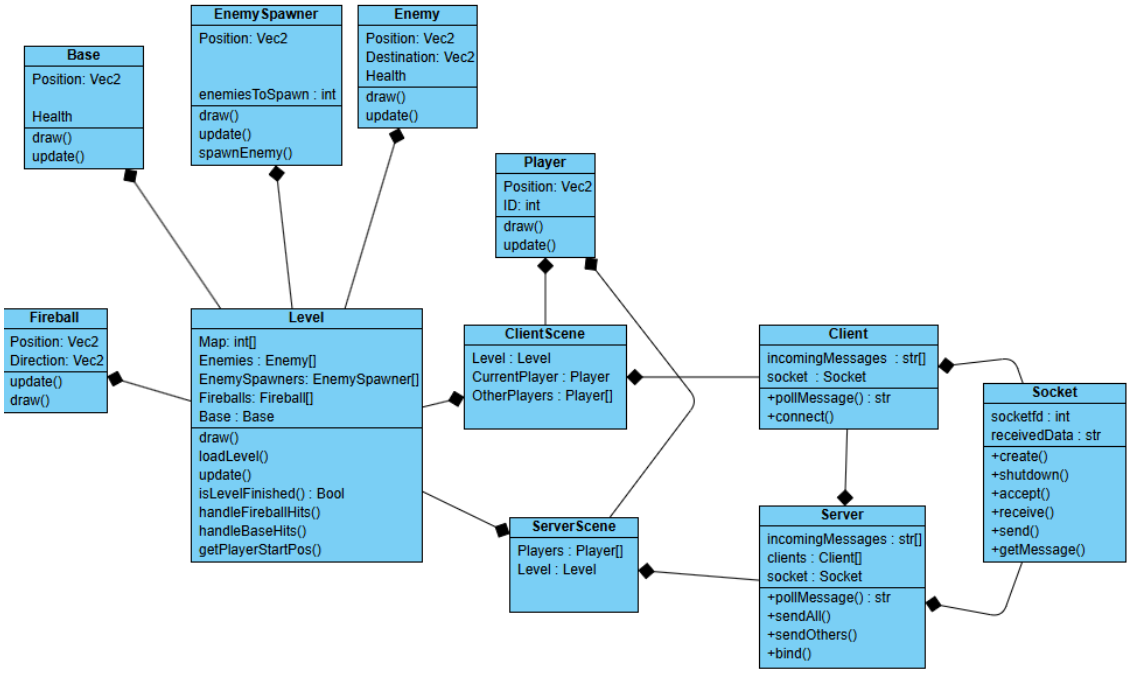
2. Schemat działania

1. Gracze łączą się z serwerem i dołączają do poczekalni.
2. Gracze wysyłają żądanie gotowości do serwera.
3. Gdy wszyscy gracze będą gotowi serwer rozpoczyna grę i wysyła każdemu klientowi informacje inicjalizujące (jego pozycję startową, i mapę).
4. Gracze wysyłają żądania przy wykonaniu akcji (ruch, strzał) do serwera i wykonują lokalną symulację rozgrywki.
5. Serwer wykonuje symulacje (obsługuje zadawanie obrażeń, tworzenie przeciwników) i na bieżąco wysyła zaktualizowany stan gry do graczy.
6. Kroki 6 oraz 7 wykonywane są do momentu zakończenia rozgrywki (porażki lub zwycięstwa).
7. Gdy rozgrywka dobiegnie końca gracze są o tym informowani oraz wracają do poczekalni, gdzie mogą zacząć kolejną rozgrywkę.

3. Diagram sekwencyjny



4. Diagram klas



Najważniejszym aspektem gry stworzonej w ramach projektu z przedmiotu jest wieloosobowa kooperacja graczy poprzez komunikacje z serwerem. W implementacji tego modelu wykorzystaliśmy sieciowe gniazda systemu Linux.

Klasa **Socket** stanowi minimalną abstrakcję nad systemowym gniazdem, pozwalające na dwukierunkową komunikację z serwerem oraz na poprawne rozróżnianie wysłanych komunikatów.

Klasa **Serwer** zawiera informacje o podłączonych klientach oraz jest odpowiedzialna za wysyłanie i odbieranie informacji od klientów. Obsługuje ona również przypadki nagłych rozłączeń klientów od serwera.

Klasa **Client** to dodatkowa abstrakcja na klasie **Socket** ułatwiająca odczytywanie komunikatów od serwera.

Klasy **ClientScene oraz ServerScene** są to główne pętle gry klienta oraz serwera. Przetwarzane są w nich komunikaty od serwera/klientów, sprawdzane są warunki zakończenia gry oraz w przypadku klasy **ClientScene** znajduje się również obsługa klawiatury i myszy.

Klasa **Level** zarządza całym światem gry – aktualizuje wszystkie elementy oraz zajmuje się ich wyświetlaniem.

Klasa **Player** przedstawia obiekt sterowany za pomocą klawiatury i myszy przez graczy.

Klasa **Enemy** dsa przedstawia strukturę przeciwnika. Kieruje się on w stronę bazy, a gdy wystarczająco sie do niej zbliży zaatakuje ją zmniejszając jej poziom życia. Przeciwnicy posiadają życie, które zmniejsza się po kolizji przeciwnika z **Fireball.**

Klasa **EnemySpawner** Klasa ta odpowiedzialna jest za tworzenie instancji **Enemy** co zadany okres. Jeżeli na mapie nie znajdują się żadni przeciwnicy a wszystkie **EnemySpawner** skończyły tworzenie przeciwników, gra jest wygrana przez graczy.

Klasa **Base** główny punkt, którego muszą bronić gracze przed atakami przeciwników. W momencie, gdy jego punkty życia spadną poniżej minimalnej wartości gracze przegrywają.