Sprawozdanie z gry sieciowej – Shooter Game

Opis projektu

Przedstawiana gra to wieloosobowy shooter sieciowy, w którym połączeni gracze rywalizują ze sobą. Wykorzystuje on rodzaj grafiki dwuwymiarowej (2D), w wersji top-down. Gracze wcielają się w postacie (reprezentowane jako czerwone koła), które poruszają się po mapie i które mogą oddawać strzały w nadanych kierunkach. Każdy gracz ma określoną ilość życia. Jeżeli ta spadnie do zera, gracz ciemnieje oraz traci zdolność poruszania się i strzelania. Rozgrywka kończy się, gdy przy życiu zostaje ostatni gracz. Według zasad gry, to właśnie ostatni żyjący gracz wygrywa grę. Gra zakłada dynamiczne sterowanie postacią za pomocą klawiszy WASD oraz wykorzystuje komunikację sieciową do przesyłania danych o pozycji i stanie gracza.

Moduł **SFML/Network** korzysta z tych samych niskopoziomowych bibliotek sieciowych w C++, co standardowe biblioteki systemów Unix/Linux, w tym nagłówki **arpa/inet**, **sys/socket.h** i inne. Oto kluczowe nagłówki oraz ich zastosowanie w SFML:

Kluczowe nagłówki w systemach Unix-like (Linux/macOS):

1. **sys/socket.h**:
    - SFML korzysta z tego nagłówka dla operacji niskopoziomowych na gniazdach, takich jak socket(), bind(), connect() itp.
2. **arpa/inet.h**:
    - W SFML te funkcje służą do konwersji i manipulacji adresami IP w klasie sf::IpAddress.
3. **netinet/in.h**:
    - Jest używany przez SFML do zarządzania adresami sieciowymi.

Technologie i biblioteki

Gra została napisana w języku C++.

Projekt został zintegrowany z systemem kontroli wersji Git. Wieloosobowa praca nad projektem była możliwa poprzez platformę GitHub obsługującą Gita.

Projekt wykorzystuje bibliotekę SFML (Simple and Fast Multimedia Library), która zapewnia funkcjonalności do obsługi grafiki, zdarzeń, sieci oraz czasu. Przykłady najważniejszych wykorzystanych klas oraz funkcji:

- **sf::RenderWindow** — do tworzenia okna gry, gdzie jest renderowana scena.
- **sf::Mouse** i **sf::Keyboard** — do obsługi wejścia od gracza, sterowanie.
- **sf::Shape** (np. **sf::CircleShape** albo **sf::RectangleShape**) — do rysowania prostych kształtów na scenie.
- **sf::Clock** — do obsługi czasu, timerów, pomiaru czasu między klatkami i implementacji ograniczeń czasowych.
- **sf::Text** — do manipulowania i renderowania komponentów tekstowych.

Projekt dodatkowo rozszerzony jest o moduł SFML Network, który został opisany wcześniej. Do przykładowych najważniejszych klas i funkcji użytych w projekcie należą:

- **sf::UdpSocket** – do deklarowania socketów wykorzystujących komunikację UDP.
- **sf::IpAddress** – do deklarowania adresów IP, potrzebnych do poprawnej implementacji funkcjonalności sieciowych w grze.
- **setBlocking** – ustawia zatrzymywanie się kodu do momentu otrzymania pakietu.
- **Receive** – nasłuchiwanie i oczekiwanie na wiadomość
- **Send** – wysyłanie wiadomości

W projekcie wykorzystano narzędzie CMake do konfiguracji i budowy projektu. Struktura projektu składa się z dwóch głównych folderów: src oraz server.

Dzięki CMake możliwe było utworzenie dwóch niezależnych plików wykonywalnych dla klienta i serwera. CMakeLists.txt w głównym katalogu i w podrzędnych katalogach pełnią rolę zarządzania konfiguracją projektu, importem bibliotek, wstrzykiwaniem potrzebnych czcionek, importem plików cpp. CMake usprawnia i przyspiesza kompilację całego projektu, zwłaszcza w przypadku gry SFML.

Struktura kodu klienta

- **Główna pętla gry**:
    - Obsługa zdarzeń (zamykanie okna, wykrywanie wejścia z klawiatury).
    - Aktualizacja pozycji gracza na podstawie danych wejściowych.
    - Wysyłanie danych gracza do serwera oraz odbieranie aktualizacji o innych graczach.
    - Rysowanie wszystkich elementów gry, w tym pocisków, graczy i wskaźnika myszy.
- **Klasa Player**:
    - Reprezentuje lokalnego i zdalnych graczy.
    - Obsługuje sterowanie określonym graczem.
    - Obsługuje życie gracza, jego paska zdrowia.
    - Obsługuje tekst wyświetlający nazwę gracza.
- **Klasa Bullet:**
    - Reprezentuje pojedynczy pocisk.
    - Wpływa na zadawanie obrażeń graczowi.
- **Klasa CursorManager**:
    - Chowa domyślny kursor i wyświetla zamiast niego spersonalizowany celownik ułatwiający strzelanie.
- **Klasa ClientSideCommunicationManager**:
    - Zarządza komunikacją sieciową z serwerem po stronie klienta.
    - Przesyła dane dotyczące pozycji gracza i pobiera informacje o pozostałych uczestnikach gry.
    - Odpowiada za łączenie się i rozłączanie z serwerem

Struktura kodu po stronie serwera

- **Główna pętla serwera**
  - Odbiór wiadomości w postaci pakietów
  - Obsługa dołączania i rozłączania graczy
  - Obsługa zadawania obrażeń
  - Obsługa aktualizacji poszczególnych graczy
  - Wysyłanie pakietów w odpowiedzi do klientów potrzebujących informacji o aktualnym stanie gry
- **Klasa ServerManager:**
  - Zarządza komunikacją sieciową po stronie serwera
  - Wysyła i odbiera pakiety
  - Zarządza listą aktualnym stanem gry
  - Weryfikuje zwycięzce

Mechaniki rozgrywki

1. **Ruch postaci**:
   - Poruszanie się postacią za pomocą klawiszy WASD.
   - Obracanie się postacią za pomocą myszki.
2. **Strzelanie**:
   - Pociski są wystrzeliwane w kierunku celownika.
   - Odbywa się za pomocą lewego przycisku myszy.
   - Zaimplementowano cooldown zapobiegający nadmiernemu spamowaniu strzałami.
3. **Interakcje z pociskami**:
   - Pociski są usuwane po opuszczeniu obszaru gry lub po trafieniu w cel.
   - Pociski trafione w gracza, zadają mu obrażenia.

Problemy i ograniczenia

- **Brak zaawansowanej obsługi kolizji**: Kolizje mogłyby być bardziej złożone i precyzyjne.

Wnioski

Projekt pokazuje, jak za pomocą SFML można stworzyć funkcjonalną, sieciową grę wieloosobową. W przyszłości możliwe jest rozwinięcie projektu o dodatkowe elementy, takie jak ulepszona grafika, system punktów i rankingów, bardziej skomplikowane kolizje i efekty

dźwiękowe.