**Zaawansowane programowanie w języku C++**

Dokumentacja projektu

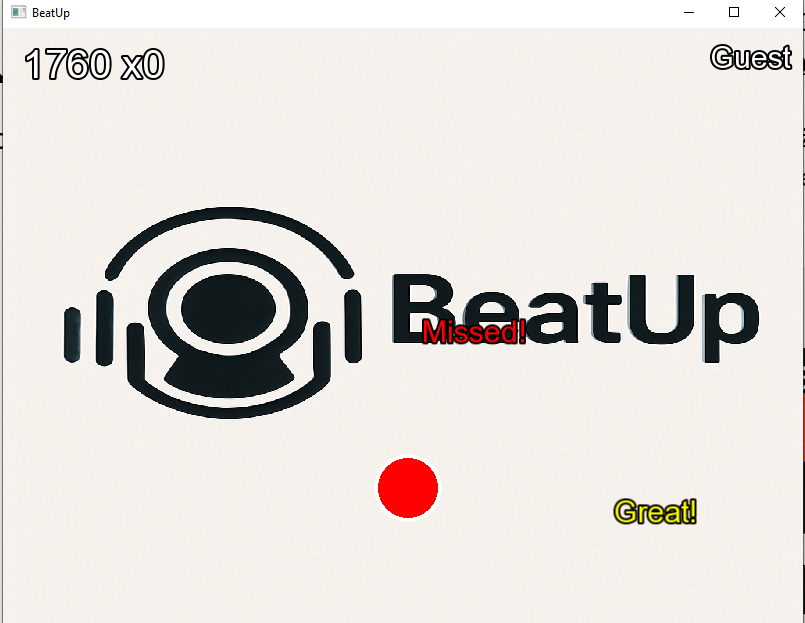
**Miłosz Hart**

**177080 L03**

1. **Wprowadzenie**

Beatup to prosta gra rytmiczno-zręcznościowa. Polega na klikaniu w nuty zsynchronizowane z podkładem muzycznym. Gracz jest wynagradzany za celne trafienie w każdą nutę w postaci punktów, ostateczny wynik wysyłany jest do bazy danych, dzięki czemu możliwa jest rywalizacja z innymi graczami. Gra zapewnia łatwy system tworzenia własnych map.





**Technologie:**

Język: C++17

User Interface: SFML

Sieć: libcurl

Środowisko: Visual Studio

1. Podstawy

*Pętla główna w Game::run()*

processEvents() – obsługa wejść

update(delta) – aktualizacja nut, odtwarzanie muzyki, logika

render() – rysowanie tła, menu, HUD, nut

*Struktura NoteInfo*

struct NoteInfo {

float x; // pozycja w osi X

float y; // pozycja w osi Y

float time; //idealny moment naciśnięcia nuty

};

*Format pliku beatmap*

#linia 1: ścieżka do pliku muzycznego

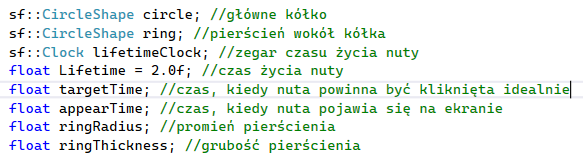
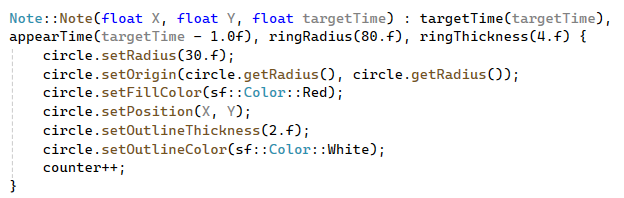
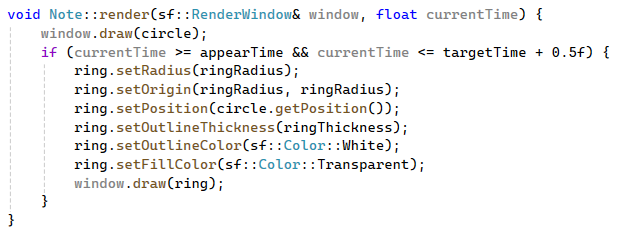
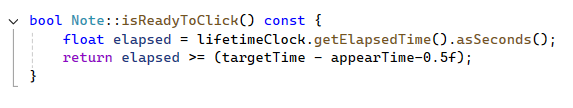
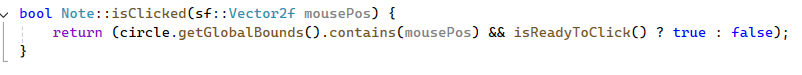
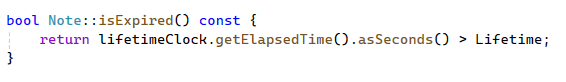
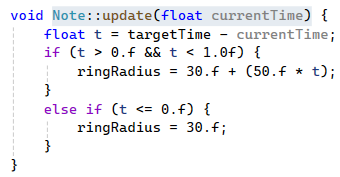
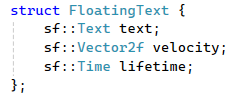
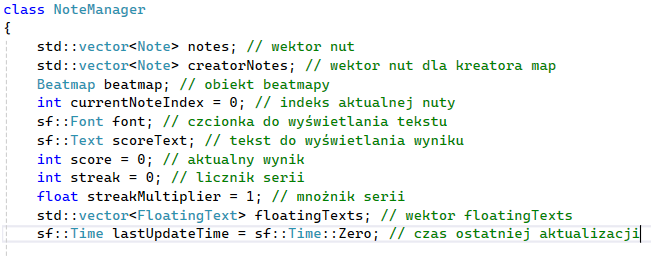
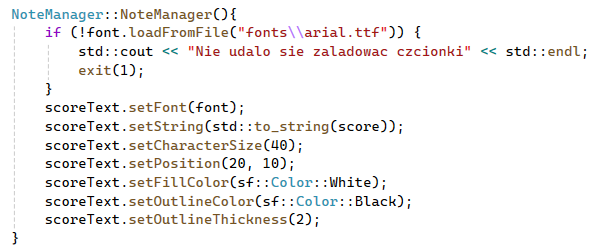
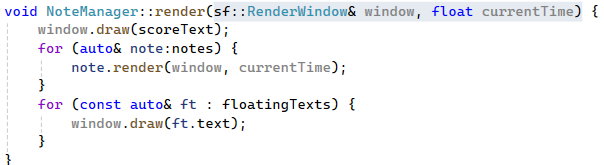
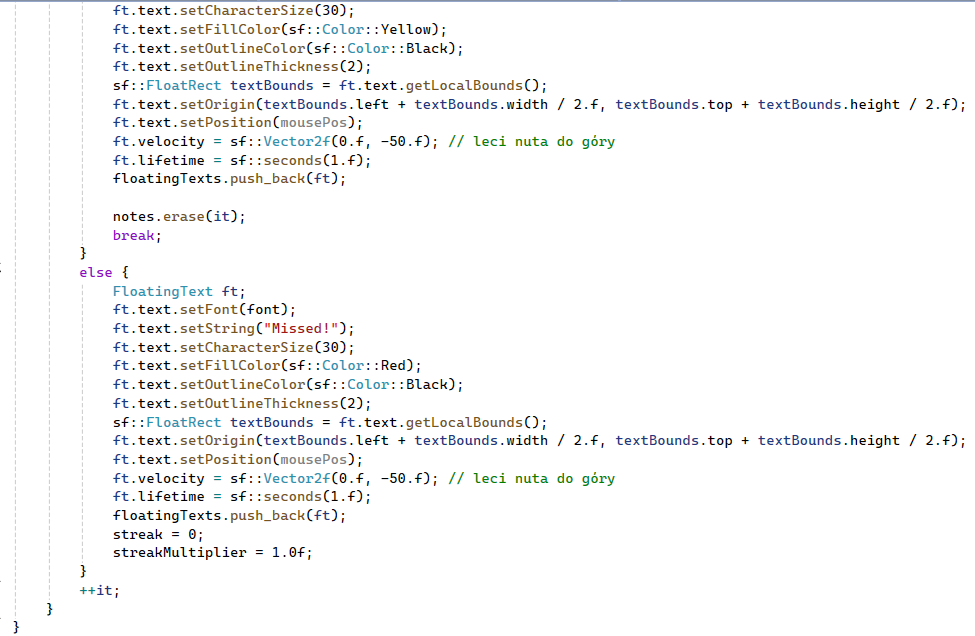
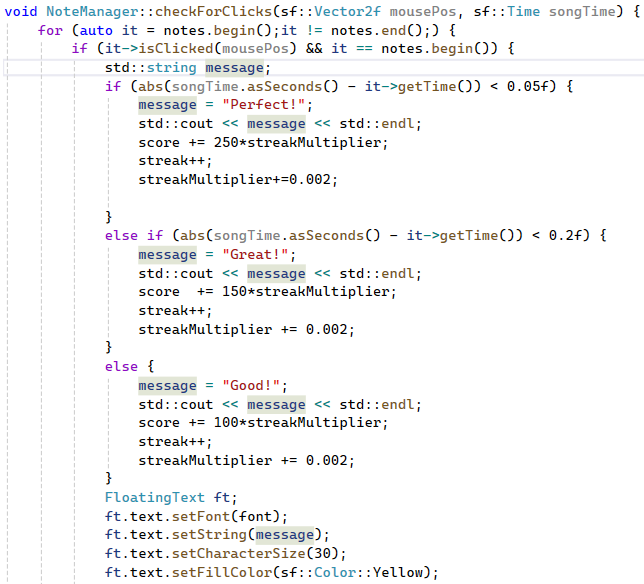
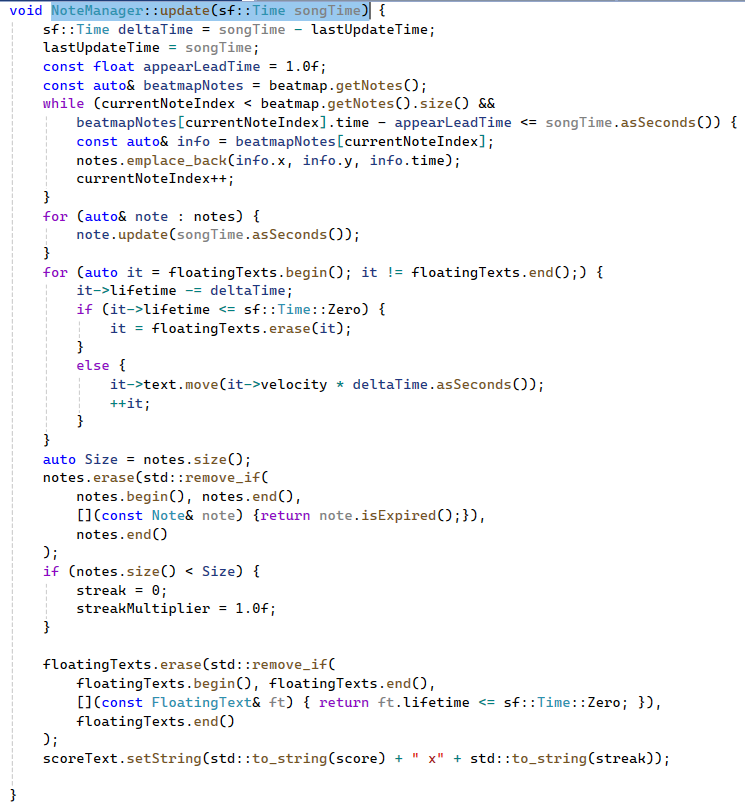
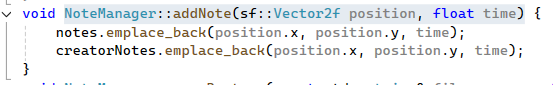
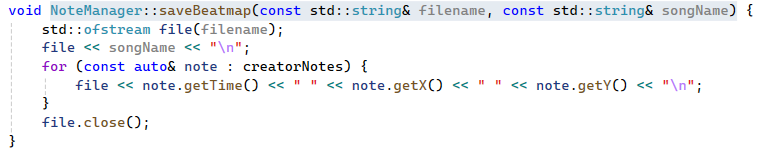
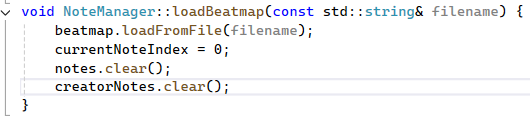
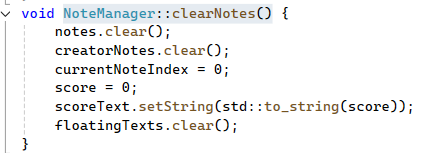
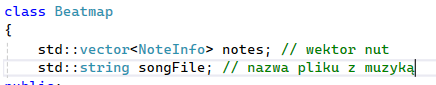
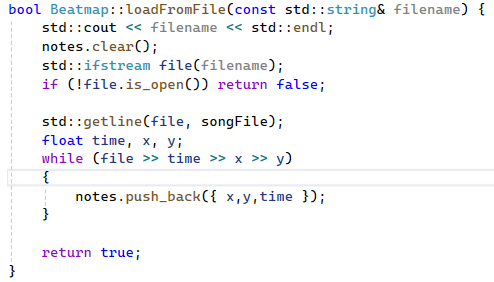
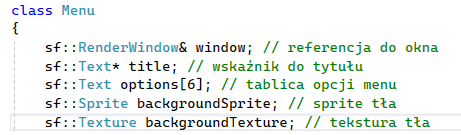
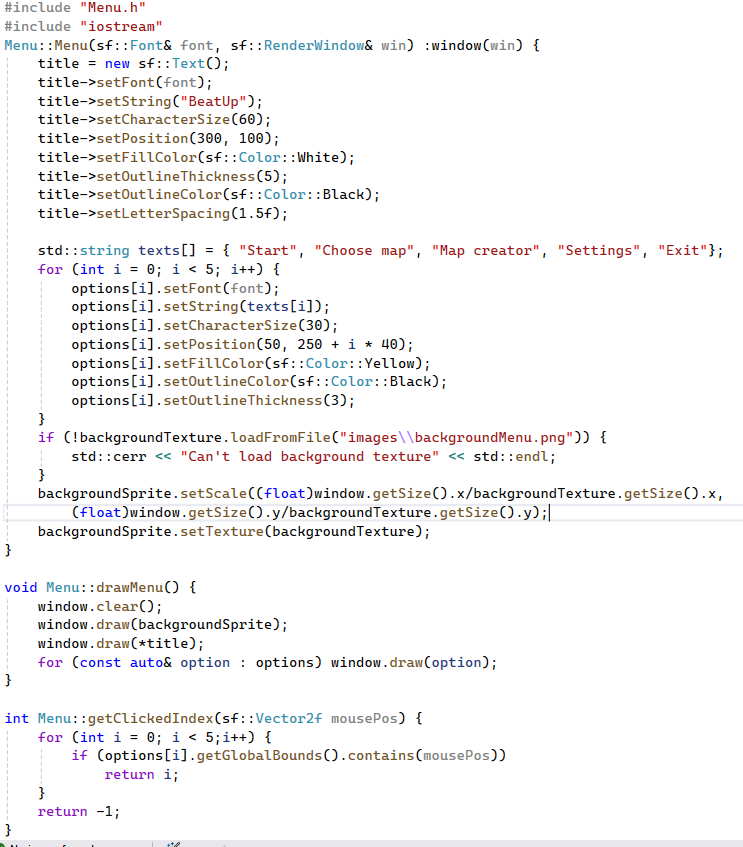
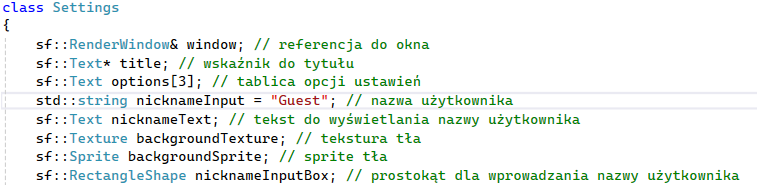
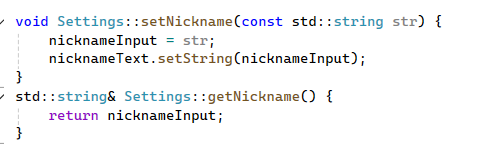
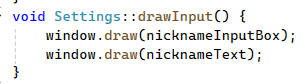
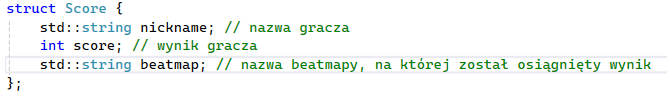
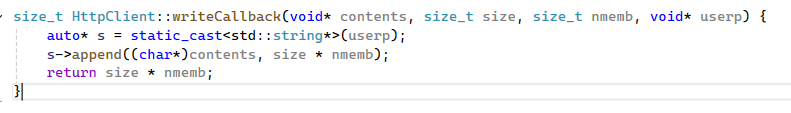
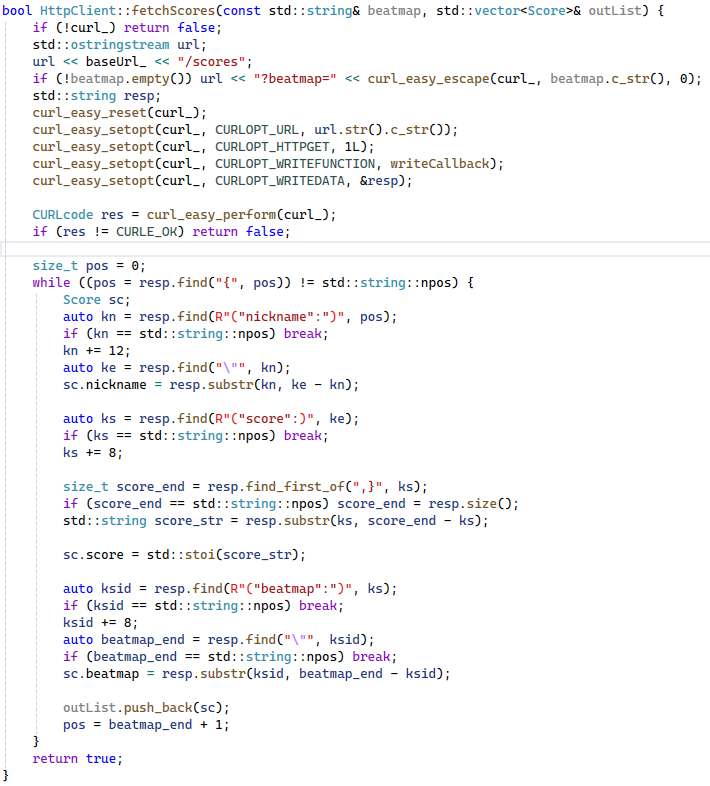
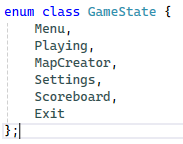
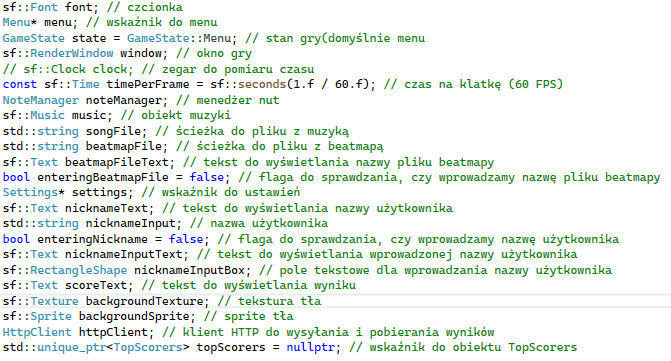
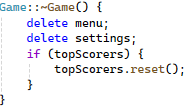
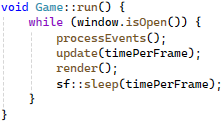
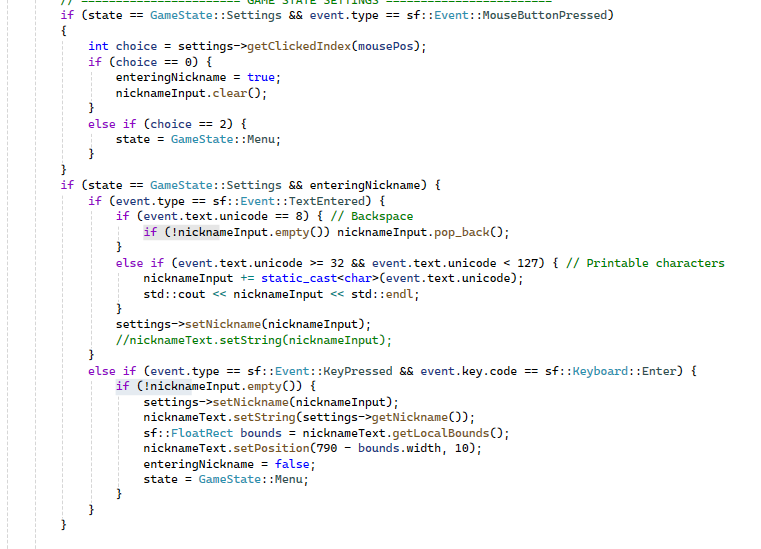
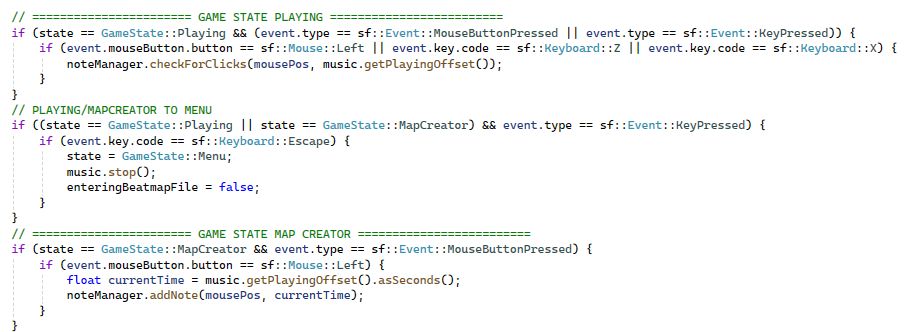
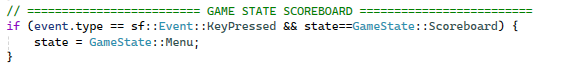
your\_song.ogg

#linia 2+: każda kolejna nuta: time x y

1.07 366 287

2.07 537 348

3.1 345 455

1. Klasa Note
   1. Pola  
      
   2. Konstruktor  
      
   3. Note::render(sf::RenderWindow& window, float currentTime)  
        
      Aktualizuje na bieżąco pierścień
   4. Note::isReadyToClick()const  
        
      Sprawdza, czy nuta jest gotowa do kliknięcia
   5. Note::isClicked(sf::Vector2f mousePos)  
        
      Na podstawie pozycji kursora sprawdza, czy obecna nuta jest gotowa na kliknięcie
   6. Note::isExpired() const  
        
      Sprawdza, czy nuta wygasła
   7. Note::update(float currentTime)  
        
      Aktualizuje wartość promienia dla pierścienia
2. Klasa NoteManager
   1. Struktura FloatingText  
        
      Odpowiada tekstowi, który pojawia się przy trafionym/nietrafionym kliknięciu
   2. Pola  
      
   3. Kreator  
        
      Ładuje czcionkę i ustawia wynik do wyświetlenia
   4. NoteManager::render(sf::RenderWindow& window, float currentTime)  
        
      Rysuje wynik, nuty, tekst “pływający”
   5. NoteManager::checkForClicks(sf::Vector2f mousePos, sf::Time songTime)Sprawdza kolejne nuty czy i jak dokładnie są naciśnięte, usuwa nutę z wektora i wyświetla napis odpowiadający dokładności.
   6. NoteManager::update(sf::Time songTime)  
        
      Sprawdza, czy nuta powinna się pojawić, usuwa przedawnione nuty I teksty, aktualizuje kolejne nuty. Jeśli nuta się przedawni, seria gracza się kończy, a mnożnik resetuje.
   7. NoteManager::addNote(sf::Vector2f position, float time)  
        
      Metoda służąca do wypełniania wektora nut dla kreatora.
   8. NoteManager::saveBeatmap(const std::string& filename, const std::string& songName)  
        
      Zapisuje dane o nutach z wektora dla kreatora do pliku podanego przez użytkownika.
   9. NoteManager::loadBeatmap(const std::string& filename)  
        
      Metoda wczytująca beatmapę.
   10. NoteManager::clearNotes()  
         
       Czyści pola zmieniane podczas gry/kracji mapy.
   11. NoteManager::getBeatmap()  
         
       Zwraca załadowaną beatmapę.
3. Beatmap
   1. Pola  
      
   2. Metody  
        
      Metoda ładująca dane z pliku do wektora notes  
        
      Metoda pobierająca wektor nut i metoda pobierająca string ze ścieżką do pliku muzycznego
4. Menu
   1. Pola  
      
   2. Metody  
        
      Metody do rysowania i obsługi menu
5. Settings
   1. Pola  
      
   2. Metody  
        
      Metody do ustawienia/pobrania nazwy gracza  
        
      Metoda do rysowania Inputa dla nazwy gracza
6. HttpCleint
   1. Struktura Score  
        
      Struktura wykorzystywana do przesłania danych do bazy danych
   2. Pola  
      
   3. Metody  
        
      Konstruktor i destruktor  
        
      Przekazuje libcurlowi sposób zapisu otrzymanych danych. Przyjmuje wskaźnik bufora z odebranymi bajtami oraz ich rozmiar i dopisuje je do podanego w userp obiektu string. size \* nmemb informuje, czy wszystkie bajty zostały poprawnie przetworzone.  
        
      Metoda wysyłająca żądanie POST do endpointa /score.  
        
      Metoda wykonująca żądanie GET do endpointa /scores.
   4. Klasa TopScorers
      1. Pola  
         
      2. Metody  
         
   5. Klasa Game
      1. Enum Class GameState  
           
         Na jego podstawie wykonane są odpowiednie operacje.
      2. Pola  
         
      3. Konstruktor  
         
      4. Destruktor  
         
      5. beatmap/songFileChoice()  
           
         Funkcja pomocnicza i metody do wybierania plików
      6. Game::run()  
           
         Główna pętla
      7. Game::procesEvents() – metoda obsługująca wejścia
         1. Obsługa w menu  
            
         2. Obsługa w opcjach  
            
         3. Obsługa wejść w grze i kreatorze  
            
         4. Zapis do beatmapy w kratorze  
            
         5. Przerwanie wyświetlania tabeli wyników  
            
      8. Game::update(sf::Time)  
           
         Tutaj m.in. wysyłany jest wynik gracza do bazy danych I aktualizowane wartości dla nut
      9. Game::render()  
           
         Tutaj rysowane są tła, przyciski, nuty.
7. Wnioski

Udało się osiągnąć w projekcie względną estetykę, intuicyjny interfejs, ale przede wszystkim spełnione zostały funkcjonalności, które są podstawą do tego typu gier. Przejrzystość kodu była dużym wyzwaniem, jednak ją ostatecznie też udało się spełnić do minimum.