Kolegium Nauk Przyrodniczych
Uniwersytet Rzeszowski

Przedmiot:

Programowanie w C++

Dokumentacja projektu:

Arkanoid

Wykonali:

Ireneusz Kiełtyka

Dariusz Niemczycki

Damian Samek

Rzeszów 2019

1. Specyfikacja projektu

1.1. Opis programu / systemu

1.1.1. Cel projektu

Celem projektu jest odtworzenie słynnej gry *Arkanoid*, która została stworzona w 1986r. i wydana w 1986r. przez firmę Taito. Gra zyskała ogromną popularność na całym świecie.

1.1.2. Szczegóły oryginalnej gry

Celem gry jest niedopuszczenie do wydostania się piłeczki poza prostokąt oraz odbijanie jej w taki sposób, aby zbić jak najwięcej klocków rozmieszczonych wewnątrz prostokąta. Na początku gracz ma do dyspozycji określoną liczbę piłeczek lub platform (w zależności od wersji gry), które traci kolejno, jeśli piłeczka ucieknie poza prostokąt. Za zbijane klocki przyznawane są punkty. Po zbiciu wszystkich klocków lub uzyskaniu odpowiedniej liczby punktów następuje przejście do kolejnego poziomu, co oznacza nowe ułożenie klocków na planszy. Jako kontroler gry służy zazwyczaj myszka lub klawiatura.

1.2. Wymagania stawiane grze

- Gra powinna zawierać kilka poziomów
- Gra powinna umożliwiać nawigację pomiędzy zakładkami
- Gra powinna współpracować z bazą danych, gdzie przechowywane i zapisywane będą najlepsze wyniki
- Gra powinna cechować się stabilnością i płynnością

1.3. Panele, które będą oferowały potrzebne funkcjonalności

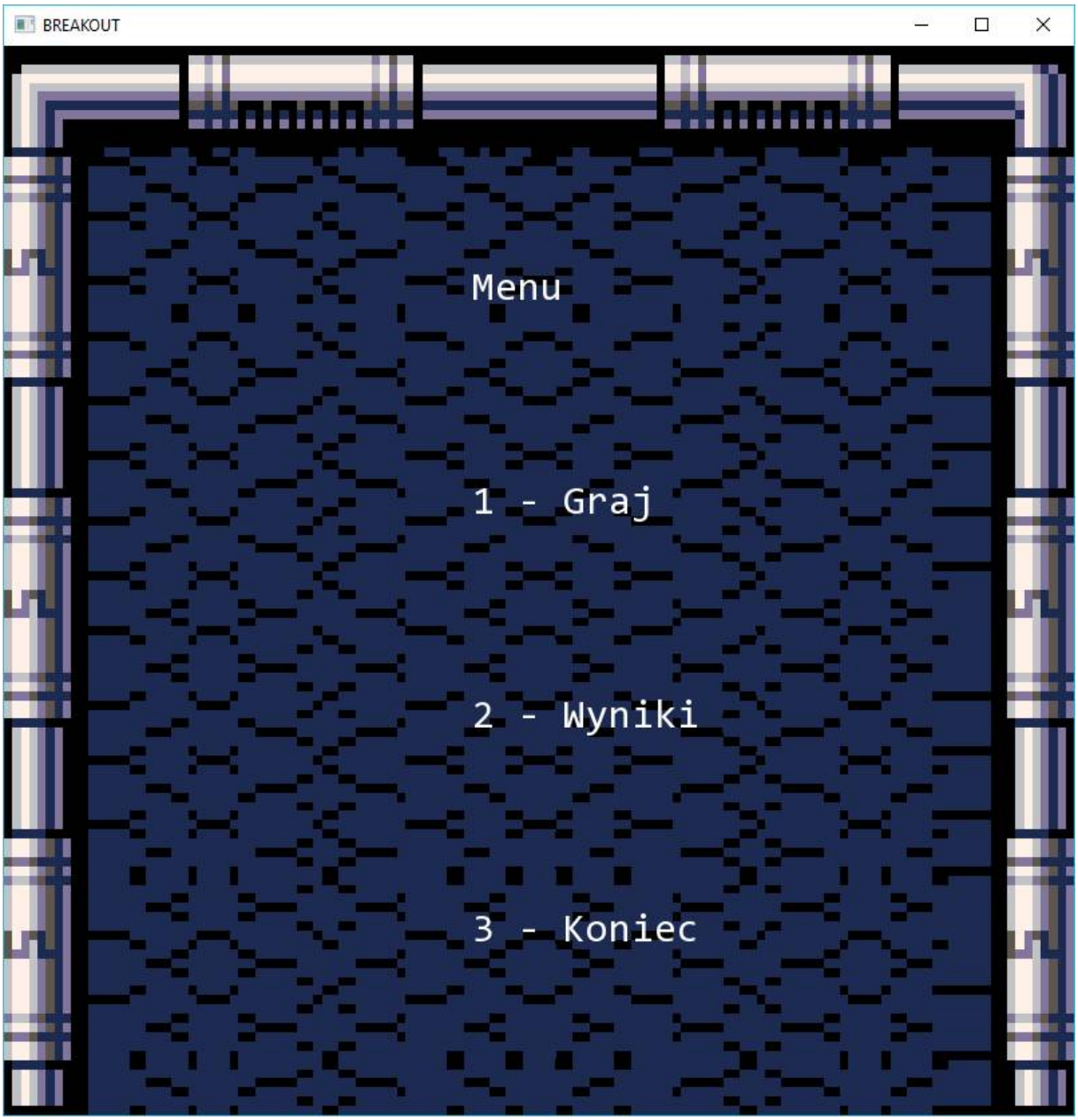
- Panel rozgrywki –Główna część projektu. Składa się on z planszy, klocków, piłeczki oraz dolnej platformy, która służy do odbijania piłeczki. Dodatkowo wyświetlane są informacje o wyniku i ilości żyć, jakie pozostały graczowi.
- Panel najlepszych wyników – Składa się z tabeli pięciu najlepszych wyników.
- Menu główne – Składają się na nie 3 opcje – graj, najlepsze wyniki i koniec

2. Wykorzystane technologie

- C++ - wieloparadygmatowy język programowania
- SFML - biblioteka programistyczna ułatwiająca tworzenie gier oraz programów multimedialnych
- MySQL - system zarządzania relacyjnymi bazami danych

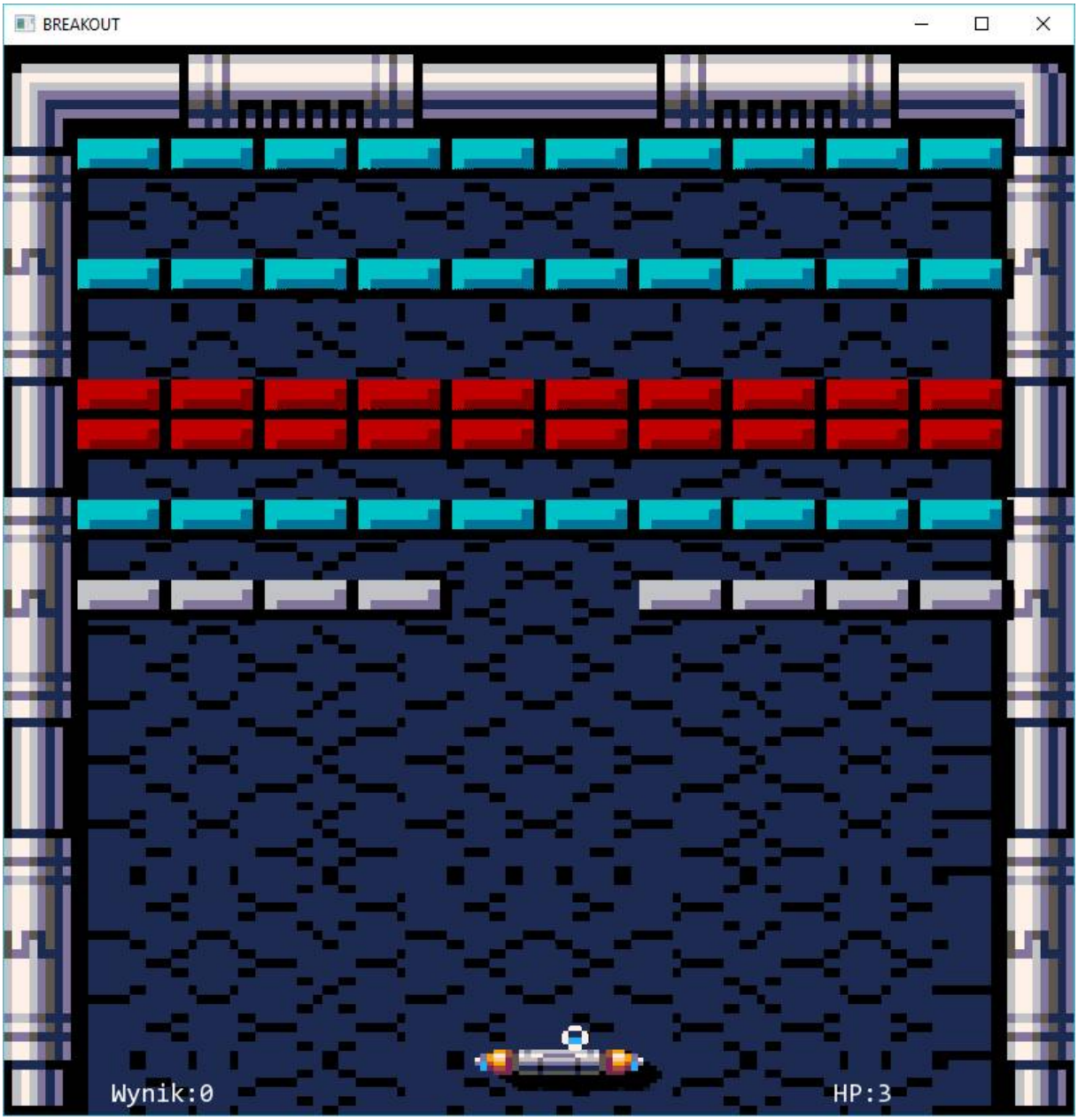
3. Interfejs gry

Po uruchomieniu gry gracz zostaje przeniesiony do menu głównego, które składa się z trzech opcji.

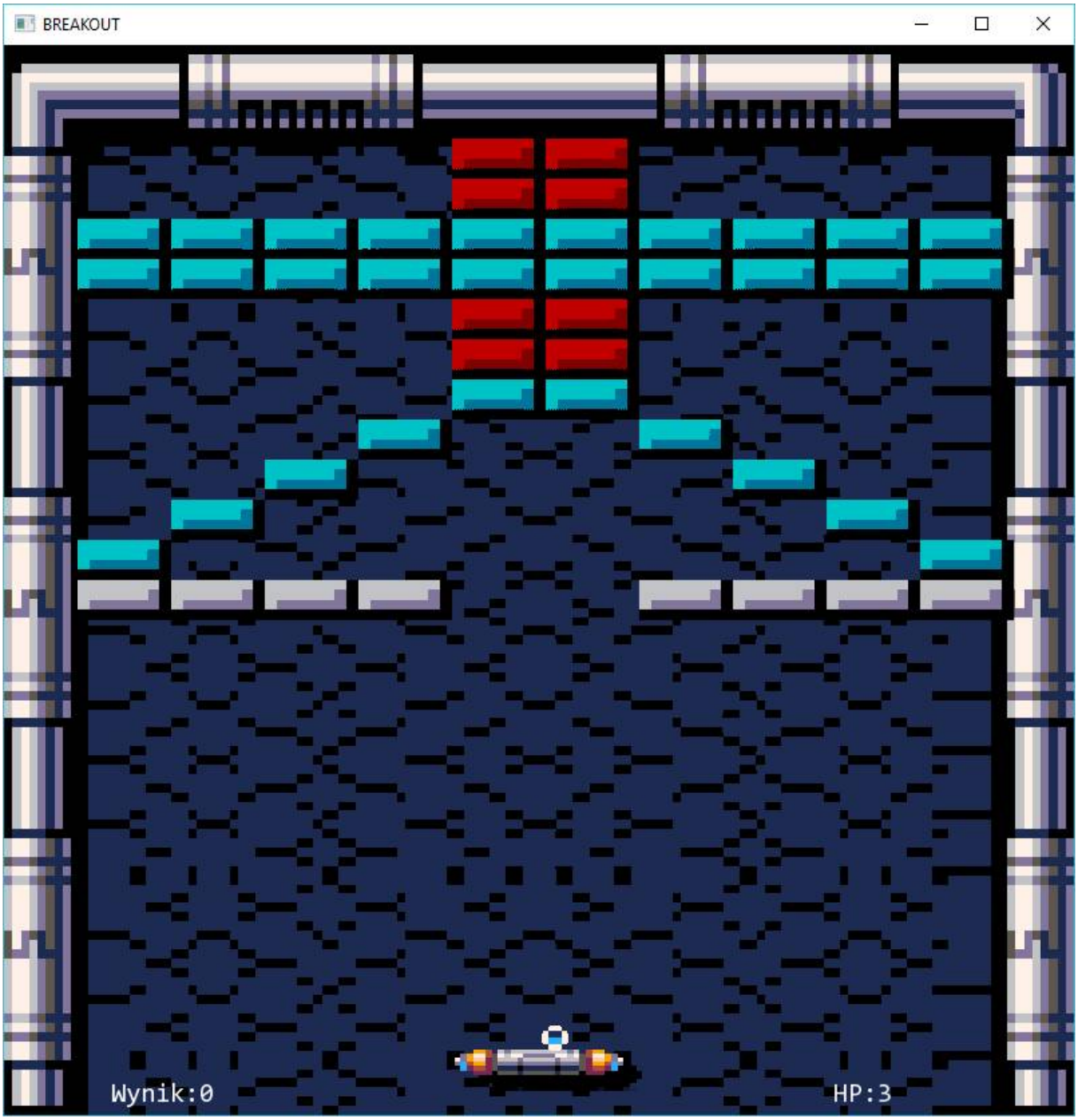


Sterowanie opcjami menu głównego jest możliwe poprzez wybór jednej z trzech opcji:

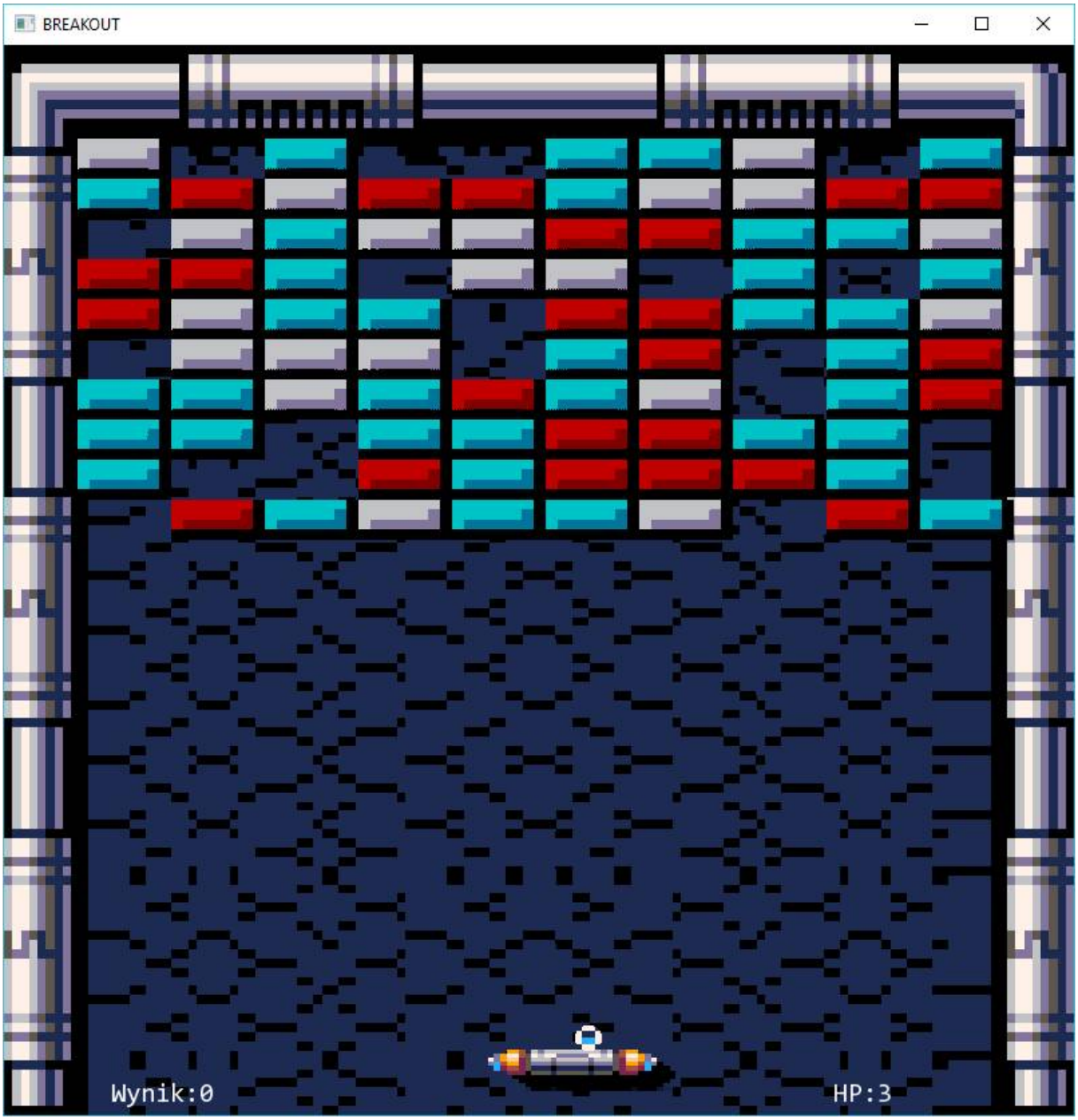
1- Graj
2- Wyniki
3- Koniec



Panel rozgrywki – poziom pierwszy. Na początku każdego poziomu platforma z piłeczką ustawiona jest w centralnej części ekranu. Sterowanie odbywa się poprzez ruchy myszki lub klawiatury (sterowanie strzałkami). W lewym dolnym rogu znajduje się licznik punktów, natomiast w prawym dolnym rogu liczba żyć. Do zniszczenia niebieskiego klocka potrzebne jest jedno uderzenie.
Do zniszczenia czerwonego klocka potrzebujemy dwóch uderzeń. Szare klocki są niezniszczalne.
Dla urozmaicenia rozgrywki w grze zastosowano ruchome przeszkody.



Panel rozgrywki – poziom drugi



Panel rozgrywki – poziom trzeci

W grze punkty naliczane są wg następującego wzoru:

$$wynik = wynik + combo * 10$$

Gdzie combo jest wartością inkrementowaną przy każdym uderzeniu klocka. Przy uderzeniu piłeczki w dolną platformę wartość combo jest zerowana.

Dźwięki gry

W grze zastosowano efekty dźwiękowe dla:

- uderzenia piłeczki o platformę,
- zniszczenia klocka,
- uderzenia klocka,
- odbicia piłeczki o ścianę
- wygranej
- przegranej
- dźwięk w tle podczas rozgrywki

Kod:

```
void menuHandleInput() {
    Event event;
    while (window.pollEvent(event)) {
        if (Keyboard::isKeyPressed(Keyboard::Num1))
        {
            currentState = GAME_STATE::GAME;
        }
        if (Keyboard::isKeyPressed(Keyboard::Num2)) {
            currentState = GAME_STATE::HIGHSCORE;
        }
        if (Keyboard::isKeyPressed(Keyboard::Num3)) {
            currentState = GAME_STATE::EXIT;
        }
    }
}
```

Przykładowy event handler (w tym przypadku dla panelu Menu)

```
void getHighScores() {

    MYSQL* conn;
    MYSQL_ROW row;
    MYSQL_RES* res;
    conn = mysql_init(0);
    conn = mysql_real_connect(conn, "localhost", "root", "", "cpp", 3306, NULL, 0);

    if (conn) {
        puts("success");

        string query = "SELECT * FROM cpp.test ORDER BY test DESC LIMIT 5";
        const char* q = query.c_str();
        int qstate = mysql_query(conn, q);
        if (!qstate) {
            res = mysql_store_result(conn);
            int i = 0;
            while (row = mysql_fetch_row(res)) {
                highscores[i] = row[1];
                i++;
            }
        }
    }
}
```

Metoda pobierająca najlepsze wyniki z bazy danych MySQL

```
void gameHandleGameOver() {
    if (life <= 0)
    {
        gameover = true;
        BGMSound.pause();
        loseSound.play();

        MYSQL* conn;
        MYSQL_ROW row;
        MYSQL_RES* res;
        conn = mysql_init(0);
        conn = mysql_real_connect(conn, "localhost", "root", "", "cpp", 3306, NULL, 0);

        string query1 = "INSERT INTO cpp.test (`id`, `test`) VALUES (NULL, '" + std::to_string(score) +"')";
        const char* d = query1.c_str();
        int state = mysql_query(conn, d);

        getHighScores();

        currentState = GAME_STATE::MENU;
    }
}
```

Metoda obsługująca koniec gry, zapis wyniku do bazy danych

```
void gameHandleKeyboardPaddleMovement() {
    if (gameover) {
        return;
    }
    if ((Keyboard::isKeyPressed(Keyboard::Left) || Keyboard::isKeyPressed(Keyboard::A)) &&
        (paddle.picture.getPosition().x - paddle.picture.getSize().x / 2.f > 50.f))
    {
        paddle.picture.move(-paddle.speed * deltaTime, 0.f);
    }
    if ((Keyboard::isKeyPressed(Keyboard::Right) || Keyboard::isKeyPressed(Keyboard::D)) &&
        (paddle.picture.getPosition().x + paddle.picture.getSize().x / 2.f < frameWidth - 50.f))
    {
        paddle.picture.move(paddle.speed * deltaTime, 0.f);
    }
    if (Keyboard::isKeyPressed(Keyboard::Space) || Mouse::isButtonPressed(Mouse::Left))
    {
        isPlaying = true;
    }
    if (!isPlaying)
    {
        ball.picture.setPosition(paddle.picture.getPosition().x, paddle.picture.getPosition().y - paddle.picture.getSize().y / 2 - ball.picture.getRadius());
    }
}
```

Metoda obsługująca uderzenie piłeczki o platformę dolną

```
bool Brick::hit()
{
    hp--;
    if (hp == 0)
    {
        enable = false;
        return true;
    }
    else
    {
        return false;
    }

}
```

Metoda obsługująca cykl życia poszczególnych klocków

4. Instrukcja uruchomienia

1. Zainstalować Visual Studio 2019
2. Pobrać projekt z repozytorium
3. Zainstalować serwer MySQL
4. Utworzyć bazę danych na serwerze lokalnym, port 3306. Nazwa bazy danych: cpp, użytkownik: root
5. Zainstalować bibliotekę SFML. Przebieg instalacji i konfiguracja projektu oraz konsolidatora zostały udokumentowane na oficjalnej stronie Frameworka SFML w zakładce Creating and configuring a SFML Project. Link: https://www.sfml-dev.org/tutorials/2.5/start-vc.php
6. Należy skonfigurować MySQL dla Visual Studio według instrukcji ze strony https://github.com/marchyl/MySQLExample. Wszystkie kroki konfiguracji są opisane i udokumentowane w pliku Readme.md
7. Po konfiguracji należy przejść do zakładki Debugowanie -> Rozpocznij debugowanie. Gra zostanie skompilowana i uruchomiona.