

| WERSJA | DATA | ZMIANY |
|--------|------------|--|
| 0.1 | 16.01.2023 | <i>Powstanie wersji poglądowej</i> |
| 0.2 | 20.01.2023 | <i>Dodanie opisów realizacji i analizy problemów</i> |
| 0.3 | 22.01.2023 | <i>Dodanie diagramów UML</i> |
| | | |

GRA – BASE DEFENSE STRATEGY GAME

Autor: Tomasz Szydłak
Akademia Górniczo-Hutnicza

Spis treści

| | | |
|-------|--|----|
| 1. | WSTĘP..... | 4 |
| 2. | FUNKcjONALNOŚĆ..... | 6 |
| 3. | ANALIZA PROBLEMU | 7 |
| 3.1 | STRONA GRAFICZNA | 7 |
| 3.1.1 | Generacja kształtów | 7 |
| 3.1.2 | Renderowanie tekstur | 7 |
| 4. | PROJEKT TECHNICZNY | 8 |
| 4.1 | ŹRÓDŁA I WYKORZYSTANIE FRAGMENTÓW KODU | 8 |
| 4.2 | ##JESZCZE NIEUŻYWANE##..... | 9 |
| 4.3 | ##JESZCZE NIEUŻYWANE##..... | 9 |
| 5. | OPIS REALIZACJI..... | 10 |
| 6. | OPIS WYKONANYCH TESTÓW - LISTA BUGGÓW, UZUPEŁNIENÍ, ITD..... | 11 |
| 7. | PODRĘCZNIK UŻYTKOWNIKA | 12 |
| 7.1 | URUCHOMIENIE PROGRAMU | 12 |
| 7.2 | OKNO GRY..... | 13 |
| 8. | METODOLOGIA ROZWOJU I UTRZYMANIA SYSTEMU | 14 |
| | BIBLIOGRAFIA..... | 15 |

Lista oznaczeń

| | |
|-----|--|
| SDL | Simple DirectMedia Player |
| UI | User Interface (Interfejs Użytkownika) |
| API | Application Programming Interface |
| Fps | Frames per second (klatki na sekundę) |

1. Wstęp

Dokument dotyczy opracowania prostej gry typu Tower Defense. Celem tego typu gier jest obrona własnej bazy przed wrogimi jednostkami, w przypadku tego projektu za pomocą własnych jednostek. Założeniem projektu jest stworzenie ciekawej i interaktywnej rozgrywki, która wymagałaby od użytkownika zastanowienia ale jednocześnie była przyjemna.

1.1 Założenia projektu

Podstawowe założenia projektu:

1. Przygotowanie założeń rozgrywki – zasady gry, sposób na wygraną, UI.
2. Określenie wymagań UI.
3. Określenie sposobu kontroli grafiki przy wykorzystaniu abstrakcji warstw.
4. Implementacja komunikacji między składnikami oprogramowania (API).
5. Testowanie sytemu graficznego.
6. Implementacja komunikacji między użytkownikiem a interfejsem.
7. Testy końcowe.

2. Funkcjonalność

1. Funkcjonalności rogrywki:

- 1.1. Czytelne UI pozwalające na bezproblemowe użytkowanie programu
- 1.2. Proste zasady i dodatkowe opisy pomagające w ich zrozumieniu
- 1.3. Stałe reguły gry pozwalające na powtarzalne ogólne doświadczenie
- 1.4. Losowe elementy rozgrywki zapewniające wyjątkowość każdej gry

2. Funkcjonalności interfejsu graficznego:

- 2.1. Renderowanie grafiki poszczególnych jednostek i poruszanie nimi
- 2.2. Przyciski odpowiedzialne za stawianie jednostek
- 2.3. Tworzenie mapy określonego rozmiaru
- 2.4. Możliwość przesuwania obszaru widocznej mapy w ograniczonym zakresie przy użyciu strzałek (góra i dół)
- 2.5. Uniwersalność i łatwość obsługi:
 - 2.5.1. Możliwość dowolnego (w zakresach rozsądku) wybrania rozmiaru okna
 - 2.5.2. Wizualne wskazówki, np. Wybrany typ jednostki
 - 2.5.3. Możliwość sprawdzenia liczby klatek poprzez wskaźnik fps w rogu

3. Funkcjonalności silnika gry:

- 3.1. Kontrolowanie i ustawianie parametrów jednostek
- 3.2. Wgrywanie i ustawianie grafik jako tekstur jednostek i przycisków
- 3.3. Kontrolowanie szybkości gry poprzez mierzenie czasu
- 3.4. Stawianie jednostek
- 3.5. Zachowania jednostek

3. Analiza problemu

3.1 Strona graficzna

3.1.1 Generacja kształtów

Biblioteka SDL pozwala na generowanie prostych kształtów takich jak prostokąty. Jest to możliwe przy wykorzystaniu 3 funkcji: `SDL_Rect`, `SDL_SetRenderDrawColor` oraz `SDL_RenderFillRect`.

`SDL_Rect` jako swoje parametry przyjmuje pozycję x lewego górnego rogu rysowanego prostokąta, pozycję y lewego górnego rogu rysowanego prostokąta a także szerokość i wysokość rysowanego prostokąta.

Następna funkcja, `SDL_SetRenderDrawColor`, ustawia kolor tego prostokąta. Przyjmuje 4 parametry: wartości RGB oraz współczynnik Alpha, który oznacza przezroczystość tekstury.

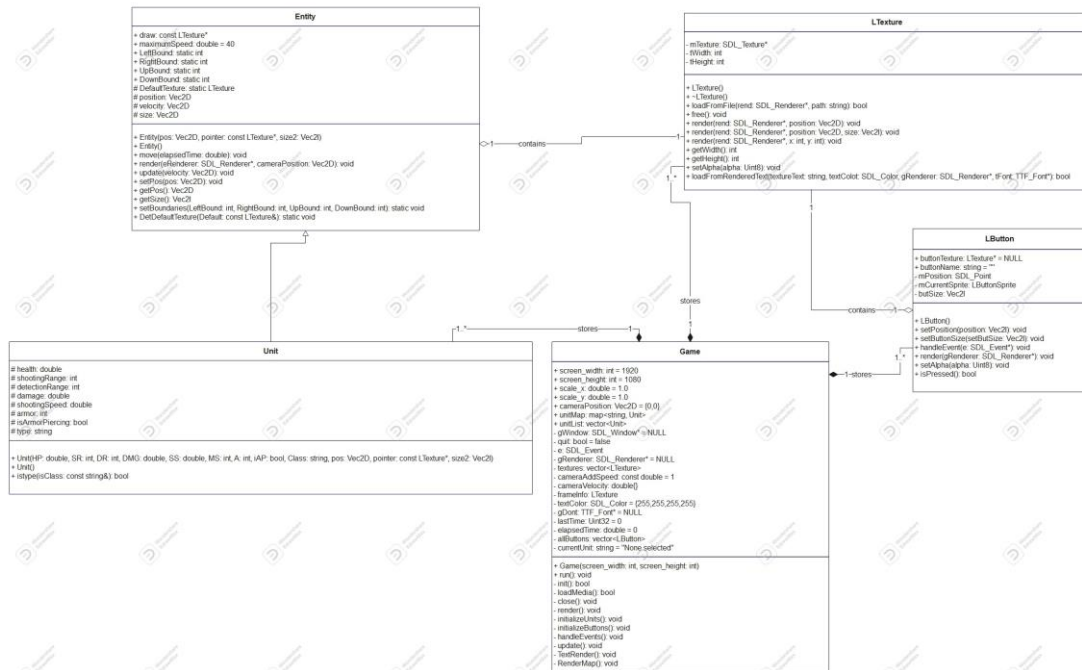
Ostatnia funkcja, `SDL_RenderFillRect`, wołana jest do renderowania określonego przez poprzednie funkcje prostokąta.

Do wyrenderowania prostokąta w oknie potrzebna jest jeszcze wbudowana funkcja `SDL_RenderPresent`, która odświeża okno wyświetlając tam stworzony kształt.

3.1.2 Renderowanie tekstur

Biblioteka SDL posiada szereg udogodnień przy pracy z teksturami. Tekstury mają własny typ nazywany `SDL_Texture`, jednak trzeba pamiętać o `SDL_Renderer`, który pozwala na wczytanie tekstur i wyświetlenie ich na ekranie.

4. Projekt techniczny



Rysunek 4.1 - Diagram klas w programie.

4.1 Źródła i wykorzystanie fragmentów kodu

Do napisania tego programu wykorzystano biblioteki SDL, SDL_image oraz SDL_ttf.

Głównym źródłem wiedzy na temat tworzenia gry była strona internetowa „Lazy Foo’ Productions” z tutorialiem „Beginning game programming v2.0” (link w bibliografii).

Podczas procesu uczenia się biblioteki były wykorzystane krótkie, uniwersalne fragmenty kodu (fragmenty funkcji, fragmenty klas, nazwy zmiennych lub opisy ich działania), lecz zdecydowana większość kodu, jeśli była skopiowana, była mocno modyfikowana lub całkowicie zmieniana na potrzeby programu. Wyszczególnione tutoriali:

- Lesson 01 „[Hello SDL](#)”
- Lesson 02 „[Getting an Image on the Screen](#)”
- Lesson 03 „[Event Driven Programming](#)”
- Lesson 04 „[Key presses](#)”
- Lesson 05 „[Optimized Surface Loading and Soft Stretching](#)”
- Lesson 07 „[Texture Loading and Rendering](#)”
- Lesson 08 „[Geometry Rendering](#)”
- Lesson 10 „[Color Keying](#)”

- Lesson 17 „[Mouse Events](#)”
- Lesson 22 „[Timing](#)”
- Lesson 26 „[Motion](#)”

4.2 ##Jeszcze nieużywane##

4.3 ##Jeszcze nieużywane##

5. Opis realizacji

Program został stworzony w środowisku Visual Studio Code C++ na 64-bitowej platformie Microsoft Windows 10. Program kompilowano przy użyciu listy Cmake a wszelkie zmiany i postępy były zapisywane przy pomocy systemu Git na platformie GitHub.

Projekt został zrealizowany przy użyciu standardowych bibliotek dostarczonych wraz z kompilatorem (np. `stdio`, `string`, `vector`). Do stworzenia interfejsu graficznego została wykorzystana biblioteka: `SDL`¹, `SDL_image`² a także `SDL_ttf`³. Biblioteki te są wykorzystywane we wszystkich częściach projektu, które obsługują okno lub jakąkolwiek grafikę.

Skompilowanie programu dzieje się automatycznie przy pomocy listy Cmake na urządzeniach posiadających system Windows, MacOS lub Linux (niektóre dystrybucje mogą nie działać). Do uruchomienia wymagane są biblioteki `SDL`, `SDL_image` oraz `SDL_ttf` zawarte w odpowiednim folderze o nazwie `Libraries`. Wymagane są także grafiki zawarte w folderze `Images`.

Oprogramowanie jest tworzone modułowo, w kolejności od najbardziej fundamentalnych funkcjonalności. W pierwszej kolejności powstał system obsługi okna. Następnie dodana została możliwość tworzenia bytów oraz możliwość reagowania na zdarzenia. Później została dodana opcja dodawania tekstur oraz system badający pozycję elementów na ekranie. W kolejnym kroku dodano możliwość przesuwania ekranu, system zarządzający czasem oraz funkcjonalność wyświetlania tekstu w wybranym foncie (wybrany został font `Karmatic Arcade`⁴). Jako następna rzecz została dodana możliwość tworzenia i używania guzików a także stawianie jednostek na mapie przy ich pomocy.

Program został napisany wykorzystując paradygmaty programowania obiektowego oraz proceduralnego. To połączenie zostało wybrane, ponieważ było najbliższe mojemu doświadczeniu z programowaniem w języku C++.

Diagramy UML przedstawiające zależności w klasach programu zostały pokazane w sekcji 4.

¹ <https://www.libsdl.org/>

² https://wiki.libsdl.org/SDL_image/FrontPage

³ https://wiki.libsdl.org/SDL_ttf/FrontPage

⁴ <https://www.1001freefonts.com/karmatic-arcade.font>

6. Opis wykonanych testów - lista buggów, uzupełnień, itd.

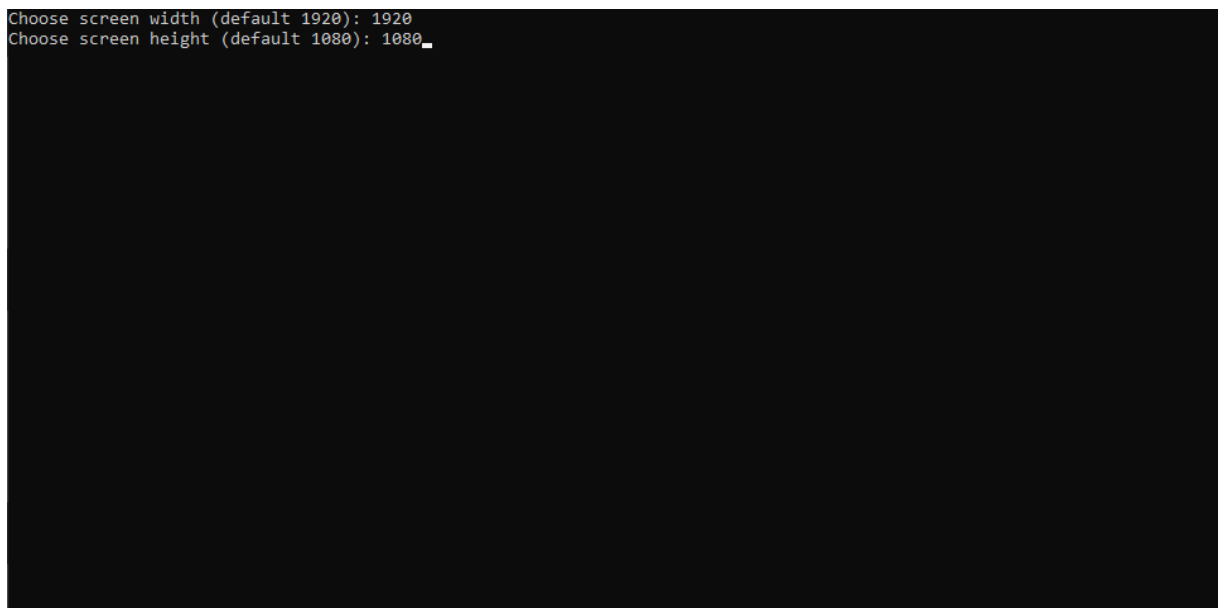
| Kod usterki | Data | Autor | Opis | Stan |
|-------------|------------|----------------|--|------------|
| #BT01 | 21.01.2023 | Tomasz Szydłak | Naciśnięcie przycisku nie pozwalało postawić jednostki | Rozwiązane |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

7. Podręcznik użytkownika

7.1 Uruchomienie programu

Oprogramowanie można uruchomić na platformie Microsoft Windows 10, MacOS albo na dystrybucjach Linuxa (wszystkie największe powinny działać). Do uruchomienia potrzebne są biblioteki SDL, SDL_image oraz SDL_ttf, które trzeba umieścić w miejscu pliku wyjściowego (.../out/build/x64-Debug). Znajdują się one w folderze Libraries.

Program zaczyna się od otwarcia okna konsoli, gdzie należy podać docelowe wymiary generowanego okna. Domyślne wartości to 1920x1080, sposób podania wartości pokazany na rysunku 7.1.

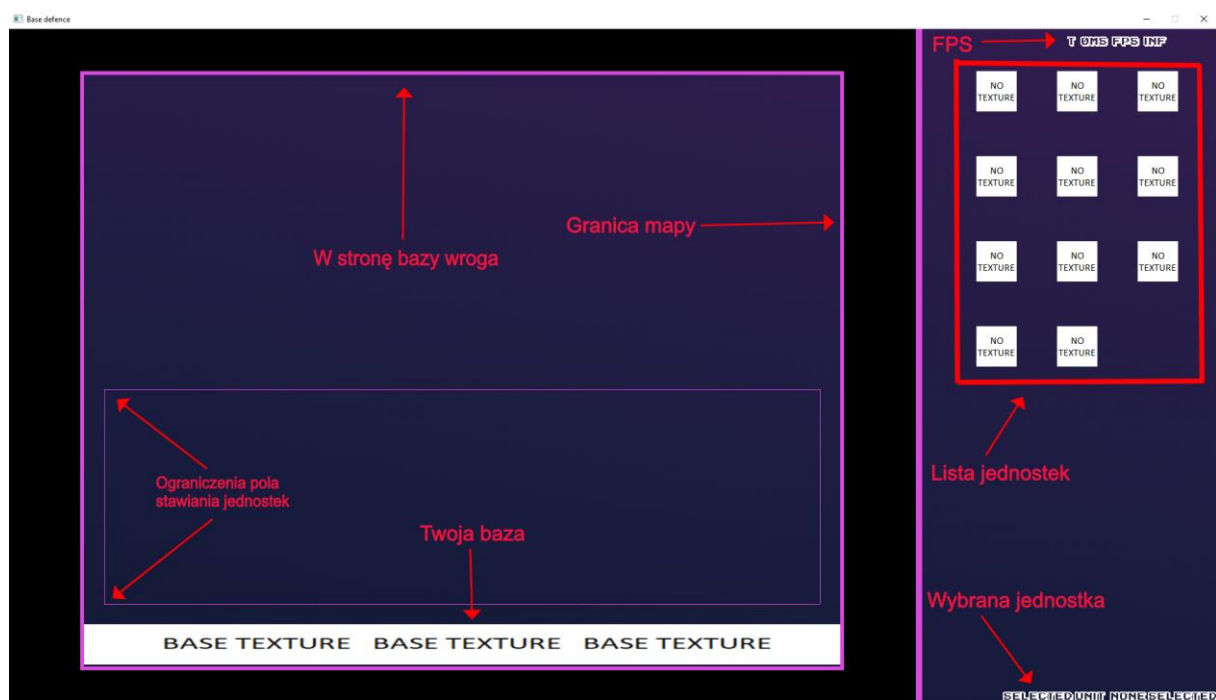


```
Choose screen width (default 1920): 1920
Choose screen height (default 1080): 1080_
```

Rysunek 7.1 – Okno określenia wymiarów okna programu

7.2 Okno gry

Po podaniu wymiarów pojawi się okno aplikacji, gdzie znajdują się wszystkie elementy gry. Istniejące elementy są wytłumaczone na rysunku 7.2.



8. Metodologia rozwoju i utrzymania systemu

Projekt został stworzony z myślą o wydajności i łatwości rozwoju. Stworzenie nowej jednostki wymaga jedynie sklasyfikowania jej cech (szybkość, życie, itp.) a także dodania odpowiedniej grafiki i dodania przycisku. Grę można także rozbudowywać o nowe mechaniki (np. ulubiony przeciwnik, statyczne jednostki, przeszkody) i nie będzie to skomplikowane dzięki rozbudowanym podstawowym możliwościom silnika gry.

Bibliografia

- [1] Lazy Foo' SDL tutorials - <http://lazyfoo.net/tutorials/SDL/index.php#Hello%20SDL>
- [2] Cyganek B.: Introduction to Programming with C++ for Engineers, Wiley-IEEE Press, 2020.