**POLITECHNIKA BIAŁOSTOCKA**

**WYDZIAŁ INFORMATYKI**

PRACA DYPLOMOWA INŻYNIERSKA

TEMAT: IMPLEMENTACJA PIKSELOWEJ GRY 2D Z WYKORZYSTANIEM BIBLIOTEKI SDL2  
IMPLEMENTATION OF A 2D PIXEL GAME   
USING THE SLD2 LIBRARY

WYKONAWCA:   
POGREBNIAK MATEUSZ

OPIEKUN PRACY DYPLOMOWEJ :   
DR INŻ. URSZULA KUŻELEWSKA

**BIAŁYSTOK 2023 ROK**

**SUMMARY**

The engineering thesis titled "Implementation of a 2D Pixel Game using the SDL2 Library" focuses on the development of an immersive and visually appealing video game using the SDL2 library in conjunction with custom graphics. The project highlights the practical application of software engineering principles, showcasing the creation of a captivating 2D pixel game with a personal touch.

The thesis begins with an introduction that outlines the motivation behind the project, emphasizing the desire to create a unique gaming experience through the integration of custom graphics. The SDL2 library, renowned for its versatility and cross-platform compatibility, serves as the foundation for the game's development. By leveraging the library's capabilities, the project aims to deliver an engaging visual experience that captivates players.

The implementation section provides an in-depth description of the technical aspects involved in creating the game. This encompasses the setup of the development environment, configuration of the SDL2 library, and the design and implementation of various game mechanics, user interface elements, and custom graphics. The thesis highlights the significance of personalized graphics, underscoring the effort and creativity invested in developing visually striking elements that enhance the overall aesthetic appeal of the game.

Throughout the development process, the thesis addresses challenges encountered and the corresponding solutions employed. It highlights the iterative nature of graphic design, discussing techniques for optimizing performance and refining graphical elements. The thesis showcases the author's creative abilities by emphasizing the contribution of personally created graphics, demonstrating the fusion of technical skills and artistic vision in the game's visual presentation.

To evaluate the implementation, the thesis employs various metrics and user feedback analysis. This assessment helps identify areas of improvement and provides insights into the effectiveness of the custom graphics in enhancing the gameplay experience. By incorporating user feedback and conducting playtesting, the thesis demonstrates a commitment to refining the game's graphics and ensuring player satisfaction.

In conclusion, the thesis successfully demonstrates the implementation of a 2D pixel game using the SDL2 library while showcasing the author's talent in creating custom graphics. By leveraging software engineering principles and investing in personalized visual elements, the project achieves its objective of providing an immersive and visually captivating gaming experience. The thesis serves as a testament to the fusion of technical expertise and artistic creativity, offering valuable insights for future game development endeavors where unique graphics play a significant role.

**SPIS TREŚCI**

[1. Wstęp 1](#_Toc137830209)

[2. Analiza problemu 3](#_Toc137830210)

[3. Analiza istniejących rozwiązań 5](#_Toc137830211)

[3.1 Pikselowe gry 2D 5](#_Toc137830212)

[3.2 Biblioteki programistyczne 8](#_Toc137830213)

[3.3 Porównanie istniejących rozwiązań 9](#_Toc137830214)

[4. Koncepcja własnego rozwiązania 10](#_Toc137830215)

[5. Analiza projektu 12](#_Toc137830216)

[5.1 Opis poszczególnych klas – część 1 13](#_Toc137830217)

[5.2 Opis poszczególnych klas – część 2 14](#_Toc137830218)

[6. Opis wykorzystanych technologii 15](#_Toc137830219)

[7. Opis aplikacji 17](#_Toc137830220)

[7.1 Interfejs użytkownika oraz funkcjonalności gry 17](#_Toc137830221)

[7.2 Zasoby graficzne 21](#_Toc137830222)

[8. Podsumowanie 27](#_Toc137830223)

# Wstęp

Gry komputerowe stanowią jedną z najpopularniejszych form rozrywki w dzisiejszych czasach, ciesząc się ogromną popularnością zarówno wśród dzieci, młodzieży, jak i dorosłych. Od początków swojego istnienia gry komputerowe przekształciły się z prostych form interaktywnych do pełnoprawnych dzieł sztuki, oferujących użytkownikom niezapomniane doświadczenia.

Dynamiczny rozwój technologii w dziedzinie gier umożliwił tworzenie coraz bardziej zaawansowanych i wizualnie imponujących produkcji. Grafika, dźwięk, animacje, fabuła i mechanika rozgrywki - to tylko niektóre elementy, które twórcy gier starają się doskonalić, aby zapewnić użytkownikom unikalne, immersyjne doświadczenia.

Jednym z kluczowych aspektów gier komputerowych jest grafika. Od prostej pikselowej estetyki retro po realistyczne trójwymiarowe środowiska, grafika odgrywa istotną rolę w tworzeniu atmosfery gry, wciąganiu graczy w wirtualne światy oraz przekazywaniu emocji. Technologie graficzne, takie jak silniki graficzne, efekty specjalne, oświetlenie dynamiczne i tekstury wysokiej jakości, umożliwiają twórcom gier osiągnięcie wizualnej finezji i realizmu.

W dziedzinie tworzenia gier istnieje wiele narzędzi, bibliotek i silników programistycznych, które wspomagają proces projektowania i implementacji. Jednym z takich silników jest SDL2 (Simple DirectMedia Layer) - biblioteka programistyczna, która dostarcza zestaw narzędzi i funkcji do tworzenia aplikacji multimedialnych, w tym gier. Silnik SDL2 cieszy się dużą popularnością ze względu na swoją prostotę, przenośność na różne platformy oraz wsparcie dla grafiki, dźwięku i wejścia od użytkownika.

Celem niniejszej pracy inżynierskiej jest zaprezentowanie procesu projektowania i implementacji pikselowej gry przy użyciu silnika SDL2, która nawiązuje do klasycznych produkcji retro. Projektowanie i implementacja gry zostały przeprowadzone w języku C++ przy użyciu środowiska Visual Studio. Wybór tego języka programowania oraz narzędzia programistycznego pozwolił na wykorzystanie pełnej mocy obiektowości oraz ułatwił proces implementacji gry na platformie SDL2.

Dodatkowo, w ramach pracy, opracowano również grafiki do gry. Projektując własne grafiki, mieliśmy możliwość stworzenia unikalnego i spójnego wizualnego świata gry, który odzwierciedla naszą kreatywność i estetykę. Wykorzystanie własnych grafik wpłynęło na oryginalność gry oraz dało nam pełną kontrolę nad wyglądem i atmosferą, jaką chcieliśmy przekazać graczom.

Praca składa się z kilku głównych etapów. Na początku przeanalizowano istniejące rozwiązania z zakresu gier pikselowych i zapoznano się z ich cechami oraz charakterystycznymi elementami. Następnie omówiono proces projektowania gry, w tym projektowanie poziomów, mechaniki rozgrywki oraz interfejsu użytkownika.

Kolejnym etapem było implementowanie gry przy użyciu silnika SDL2 w języku C++. Zaprezentowano proces tworzenia podstawowych elementów gry, takich jak postacie, przeszkody, animacje ~~oraz efekty dźwiękowe~~. Wykorzystanie pełnej funkcjonalności silnika SDL2 oraz języka C++ pozwoliło na elastyczność i efektywność w implementacji gry.

Ostatnim etapem pracy było przetestowanie i optymalizacja gry. Przeprowadzono niewielkie testy wydajnościowe, aby upewnić się, że nasza gra działa płynnie i jest przyjemna w użytkowaniu. Skoncentrowano się również na optymalizacji kodu i zasobów, aby zwiększyć wydajność gry oraz zmniejszyć jej rozmiar.

W rezultacie tej pracy inżynierskiej uzyskano gotową pikselową grę na silniku SDL2, która nie tylko stanowi efekt naszego wysiłku, ale również platformę do dalszego rozwoju i modyfikacji. Praca w języku C++ w środowisku Visual Studio oraz samodzielne projektowanie grafik pozwoliło nam na pełną kontrolę nad procesem tworzenia gry i osiągnięcie zamierzonych efektów.

Mam nadzieję, że ta praca przyczyni się do pogłębienia naszej wiedzy na temat projektowania i implementacji gier oraz dostarczy inspiracji dla przyszłych projektów.

# Analiza problemu

Gry 2D przeszły długą drogę pod względem grafiki i gatunków, czasami chcesz wrócić do piękna pikselowego stylu! Witamy w naszej kategorii gier pikselowych 2D, w której znajdziesz fantastyczne tytuły, takie jak APICO, Dead Cells i Core Keeper!

W tym rozdziale zostaną zidentyfikowane główne wymagania projektowe. Nastąpi analiza funkcjonalności gry oraz potencjalnych wyzwań, jakie napotkano w trakcie realizacji projektu. Analiza problemu pozwoli lepiej zrozumieć zakres projektu i ustalić aspekty istotne podczas implementacji pikselowej gry 2D.

**Wymagania projektowe**

~~W celu uzyskania jasnego zrozumienia problemu, zostały zidentyfikowane wymagania projektowe.~~ Ustalono, że przedmiotem pracy będzie pikselowa gra 2D, opierająca się na estetyce retro, z prostymi pikselowymi grafikami. W założeniu miała to być przyjemna dla oka gra z intuicyjną rozgrywką, która zapewni użytkownikom zarówno rozrywkę, jak i wyzwania.

Dodatkowo gra będzie obejmowała elementy takie jak ruch postaci, kolizje, interakcję z obiektami na planszy, ~~efekty dźwiękowe i muzykę,~~ jak również oceny wyników graczy.

**Potencjalne wyzwania**

Podczas analizy problemu, zidentyfikowane zostały również potencjalne wyzwania, z jakimi możemy się spotkać podczas implementacji gry. Niektóre z tych wyzwań mogą obejmować:

* Efektywność i wydajność: Pikselowa gra 2D może wymagać odpowiedniej optymalizacji, aby zapewnić płynność działania nawet na starszych komputerach.
* Zarządzanie zasobami: Względnie duże ilości grafik, ~~dźwięków i muzyki~~ mogą wymagać odpowiedniego zarządzania zasobami, tak aby nie obciążać pamięci i dysku.
* Projektowanie i implementacja poziomów: Tworzenie różnorodnych, interesujących i równoważnych poziomów gry może być wyzwaniem, które trzeba będzie skonfrontować.

**Analiza funkcjonalności**

Ważnym etapem analizy problemu było zidentyfikowanie kluczowych funkcjonalności, które gra powinna posiadać. Przeanalizowano różne gry 2D, zarówno pikselowe, jak i współczesne, aby wyciągnąć wnioski i zdefiniować podstawowe elementy gry, takie jak:

* Automatycznie i losowo generowana plansza
* Mechanika ruchu bohatera gracza
* Statystki bohatera gracza
* System kolizji i interakcji z otoczeniem
* ~~Efekty dźwiękowe i muzyka~~
* Algorytmy sztucznej inteligencji – maszyna stanów
* Ocena wyników graczy

# Analiza istniejących rozwiązań

W tym rozdziale zostanie przeprowadzona analiza istniejących rozwiązań w dziedzinie pikselowych gier 2D oraz bibliotek programistycznych do tworzenia gier. Celem analizy jest zapoznanie się z istniejącymi rozwiązaniami, ich zaletami i wadami, aby móc dokonać świadomego wyboru technologii i podejść w projekcie. Na podstawie tej analizy dokonano wyboru ostatecznych technologii i narzędzi do implementacji pikselowej gry 2D.

## 3.1 Pikselowe gry 2D

Na początek dokonano analizy istniejących pikselowych gier 2D, aby zrozumieć, jakie gatunki i style są popularne w tej kategorii. Analizie poddano zarówno starsze, klasyczne tytuły, jak i nowsze produkcje, które wykorzystują estetykę pikseli. Przeanalizowano różne aspekty gier, takie jak mechanika rozgrywki, poziomy trudności, elementy wizualne, fabuła oraz reakcje graczy i opinie społeczności. W ten sposób zostało zrobione rozeznanie odnośnie preferencji graczy w pikselowych grach 2D.

**Terraria**

Terraria to 2D gra przygodowa z otwartym światem, w której gracze eksplorują rozległy świat pełen tajemnic i niebezpieczeństw. Gra oferuje swobodę budowania, walki z potworami, zdobywania surowców i tworzenia przedmiotów. Gracze mogą również odkrywać ukryte podziemne lochy, walczyć z bossami i współpracować z innymi graczami w trybie wieloosobowym. Terraria cechuje się unikalnym stylem pikselowej grafiki, która dodaje uroku i nostalgii.



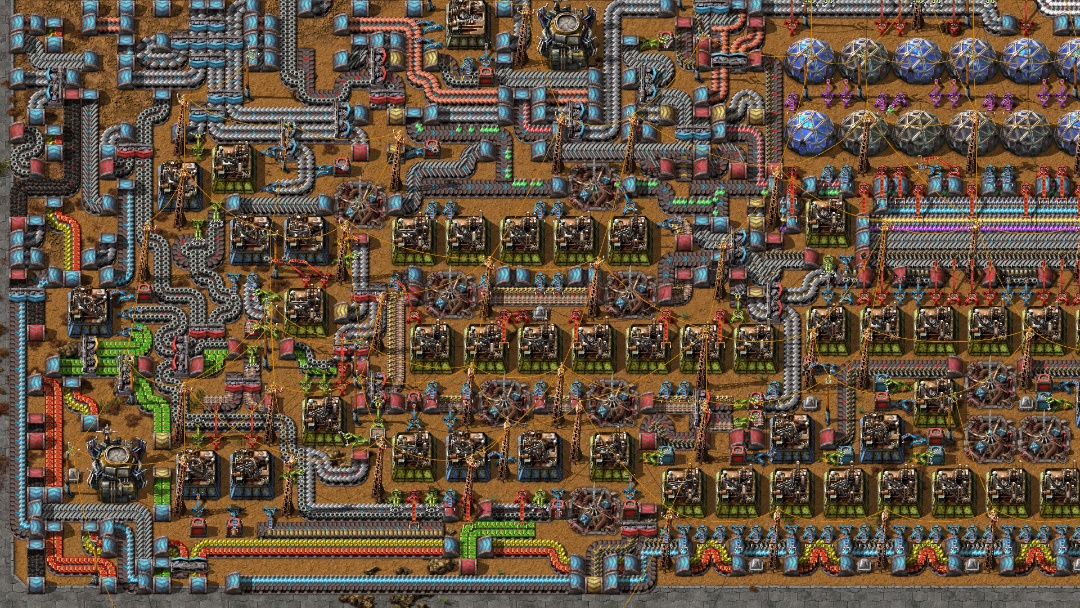
**The Binding of Isaac**

The Binding of Isaac to 2D roguelike gra akcji z elementami strzelaniny. Gracz wciela się w postać chłopca o imieniu Isaac, który ucieka przed swoją szalonym matką w otchłań piwnicy. Gra oferuje losowo generowane poziomy, potwory i przedmioty, co sprawia, że każda rozgrywka jest inna. Gracz musi pokonać różne przeciwności, zbierać power-upy i rozwijać postać, aby stawić czoła coraz trudniejszym wyzwaniom. Styl graficzny gry jest mroczny i charakterystyczny, przekazując atmosferę niepewności i grozy.



**Factorio**

Factorio to 2D symulacyjna gra ekonomiczna, w której gracz ma za zadanie budować i zarządzać fabryką. Celem gry jest automatyzacja produkcji i optymalizacja procesów, aby zaspokoić rosnące potrzeby przemysłu. Gracz musi wydobywać surowce, projektować linie produkcyjne, badania technologiczne i utrzymywać produkcję. Grafika w grze jest prostych pikseli, ale umożliwia precyzyjne planowanie i konstrukcję zaawansowanych układów fabrycznych.



**Pokemon Emerald**

Pokemon Emerald to 2D przygodowa gra RPG, będąca częścią popularnej serii Pokémon. Gracz wciela się w trenera Pokemonów, który podróżuje po świecie, łapie różnorodne stworzenia zwane Pokemonami, trenuje je i walczy z innymi trenerami. Celem gry jest zbudowanie silnego zespołu Pokemonów i pokonanie Ośmiu Mistrzów, aby stać się Mistrzem Ligi Pokemonów. Gra oferuje rozbudowany świat do eksploracji, liczne zadania i unikalny system walki turowej. Grafika w grze jest kolorowa i przyjazna dla oka, co podkreśla urok Pokemonów.



## 3.2 Biblioteki programistyczne

W kolejnym kroku przeanalizowane zostały różne biblioteki programistyczne, które są popularne w tworzeniu gier 2D, takie jak SDL (Simple DirectMedia Layer), SFML (Simple and Fast Multimedia Library), Allegro czy Love2D. Dokładnie przyjrzano się ich funkcjonalnościom, wydajności, dostępnej dokumentacji i wsparciu społeczności. Skupiono się na tych, które obsługują pikselową grafikę i są kompatybilne z językiem C++.

**Simple DirectMedia Layer (SDL)**

SDL to popularna biblioteka wieloplatformowa, która zapewnia prosty i wydajny sposób na tworzenie gier 2D w języku C++. Oferuje zestaw narzędzi do obsługi grafiki, dźwięku, wejścia od użytkownika i innych podstawowych funkcji potrzebnych do tworzenia gier. SDL jest cenione za swoją prostotę i łatwość użycia, jednocześnie dając dużą kontrolę nad interakcją z niskopoziomowymi funkcjami systemowymi. Jest szeroko stosowane w branży gier i posiada aktywną społeczność deweloperów.

**Simple and Fast Multimedia Library (SFML)**

SFML to kolejna popularna biblioteka C++, która umożliwia tworzenie gier 2D. Zapewnia interfejs programistyczny do obsługi grafiki, dźwięku, sieci, wejścia od użytkownika i innych aspektów gry. SFML jest znane z prostoty użycia, wydajności i rozbudowanego zestawu funkcji. Biblioteka posiada również wsparcie dla wielu platform, w tym Windows, macOS, Linux i inne. SFML posiada czytelną dokumentację i aktywną społeczność, co ułatwia naukę i rozwiązywanie problemów.

**Allegro**

Allegro to kolejne narzędzie, które jest często wykorzystywane do tworzenia gier 2D w języku C++. Zapewnia zestaw funkcji do obsługi grafiki, dźwięku, wejścia od użytkownika, animacji i innych aspektów gry. Biblioteka Allegro jest znana z prostego interfejsu, elastyczności i możliwości rozszerzania jej funkcjonalności poprzez dodatkowe moduły. Posiada wsparcie dla wielu platform i jest aktywnie rozwijana przez społeczność deweloperów.

**Love2D**

Love2D to otwarte źródło framework do tworzenia gier 2D w języku Lua, ale oferuje także API dla języka C++. Love2D dostarcza narzędzi do zarządzania grafiką, dźwiękiem, fizyką, wejściem od użytkownika i innych aspektów gry. Biblioteka ma prostą i intuicyjną składnię oraz wiele gotowych funkcji, które ułatwiają tworzenie gier. Love2D jest popularne wśród twórców gier indie i posiada aktywną społeczność, która udostępnia wiele przykładów i rozszerzeń.

## 3.3 Porównanie istniejących rozwiązań

Po zebraniu informacji na temat pikselowych gier 2D i bibliotek programistycznych zostanie dokonana ocena, które rozwiązania najlepiej pasują do celów i wymagań projektowych. Podczas porównania brana pod uwagę były takie kryteria jak łatwość użycia, elastyczność, dostępność narzędzi i zasobów, wydajność, przenośność na różne platformy oraz wspierane funkcje, takie jak obsługa grafiki, dźwięku, wejścia od użytkownika czy sieci.

Porównanie.

# Koncepcja własnego rozwiązania

W tym rozdziale przedstawiona zostanie koncepcja rozwiązania, które zostanie zaimplementowane w tworzonej pikselowej grze 2D. Opisane zostaną główne założenia, cele i funkcjonalności, aby stworzyć interesującą i satysfakcjonującą grę.

**Cele projektowe**

Pracę na projektem rozpoczęto od określenia celów do osiągnięcia. Brano pod uwagę rozgrywkę, estetykę, technologię, wydajność, jak również inne aspekty gry. Ostatecznie została podjęta decyzja o:

* Stworzeniu gry o ciekawej i wciągającej mechanice rozgrywki, która zapewni graczom zarówno zabawę, jak i wyzwania.
* Osiągnięciu estetyki pikselowej, która nawiązuje do retro gier 2D, z dbałością o szczegóły i unikalny styl wizualny.
* Zaimplementowaniu systemu kolizji, który będzie sprawiedliwy i precyzyjny, zapewniając płynne interakcje między postacią a obiektami na planszy.
* Dodaniu elementów interaktywnych i zagadek, które zachęcą graczy do eksploracji i odkrywania nowych obszarów gry.
* Optymalizacji gry pod kątem wydajności, tak aby działała płynnie na różnych platformach i komputerach.

**Funkcjonalności**

Następnie skupiono się na głównych funkcjonalnościach gry. Zdecydowano, że będą to funkcje związane z rozgrywką, interfejsem użytkownika, grafiką, dźwiękiem, czy innymi aspektami i powinny obejmować:

* Automatycznie i losowo generowaną planszę:  
  Gra powinna mieć kilka poziomów, z różnymi układami plansz, przeciwnikami.
* Mechanikę ruchu bohatera gracza:  
  Gracz powinien móc poruszać swoją postacią (lewo, prawo, góra, dół) i korzystać z innych umiejętności w zależności od konkretnych wymagań gry.
* Statystki bohatera gracza:  
  Gracz powinien mieć możliwość sprawdzania swoich statystyk w dowolnym momencie gry. Pomogłoby mu to w rozgrywce oraz uczyniłoby ją bardziej przejrzystą.
* System kolizji i interakcji z otoczeniem:  
  Zostanie zaimplementowany system kolizji, który będzie wykrywać kolizje między postacią a innymi obiektami na planszy, takimi jak przeszkody, wrogowie lub przedmioty do zebrania.  
  Postać gracza powinna mieć możliwość interakcji z różnymi obiektami na planszy, takimi jak portale, przełączniki, skrzynie itp., co może prowadzić do odkrywania nowych obszarów lub rozwiązywania zagadek. W grze powinny być dostępne przedmioty, które gracz może zbierać, takie jak monety, mikstury, itp.
* ~~Efekty dźwiękowe i muzyka:   
  Dodamy dźwięki i muzykę, aby wzmocnić atmosferę gry i uczynić ją bardziej immersyjną dla graczy.~~
* Algorytmy sztucznej inteligencji – maszyna stanów:  
  Zachowanie przeciwników powinno się zmieniać w zależności od akcji podjętych przez gracza.
* Ocenę wyników graczy:   
  Zapewnienie możliwości oceny wyników graczy, tak aby mogli oni porównywać swoje osiągnięcia.

**Architektura i narzędzia**

W tym podrozdziale zostanie omówiona architektura naszego rozwiązania oraz narzędzia, które zostaną wykorzystane do implementacji gry. Warto znów wspomnieć o wykorzystaniu języka C++ w wersji 14 wraz z biblioteką SDL2, Microsoft Visual Studio jako środowisko programistyczne oraz inne narzędzia pomocnicze, takie jak programy do projektowania grafiki ~~czy edytory dźwięku~~. Opisana zostanie również struktura projektu, podział na moduły i klasy. Omówione zostaną ważne decyzje projektowe, takie jak organizacja plików, zarządzanie zasobami czy obsługa zdarzeń.

Przedstawienie koncepcji własnego rozwiązania pozwoli czytelnikowi lepiej zrozumieć, w jaki sposób została zrealizowana pikselową grę 2D i jakie są główne elementy, które zostaną w niej uwzględnione.

Struktura projektu, podział na klasy

# Analiza projektu

W tym rozdziale zostanie przeanalizowany projekt tworzonej pikselowej gry 2D. Skupimy się na identyfikacji funkcjonalności, ograniczeń technicznych i innych czynników, które będą miały wpływ na projekt i implementację gry. Analiza wymagań systemowych i wykonanie projektu systemu pozwolą na lepsze zrozumienie wymagań projektowych i pomogą w późniejszych etapach implementacji gry.

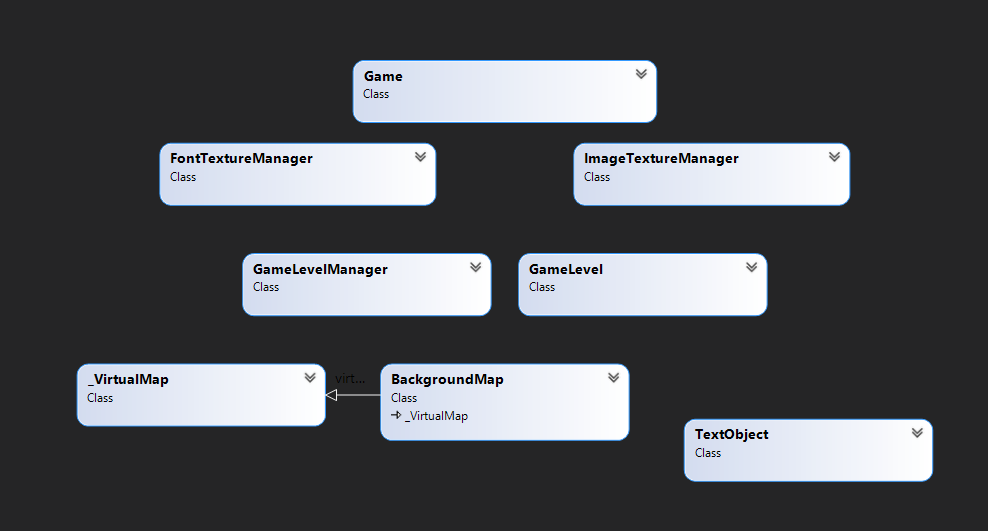
Po analizie wymagań i funkcjonalności przystępujemy teraz do wykonania projektu naszego systemu. Spiszemy opis architektury, przedstawiający główne komponenty, ich relacje i zadania. Możemy wykorzystać diagramy UML, takie jak diagram klas, diagram sekwencji czy diagram stanów, aby lepiej zobrazować strukturę i zachowanie systemu.

W projekcie systemu uwzględnimy również organizację plików, zarządzanie zasobami (grafiki, dźwięki), obsługę zdarzeń, logikę gry, interfejs użytkownika i inne istotne elementy.

Ważne jest, aby projekt systemu był czytelny, dobrze udokumentowany i zrozumiały dla innych osób, które mogą być zaangażowane w proces implementacji gry.

### 5.1 Opis poszczególnych klas – część 1

Wygenerowany przez rozszerzenie do Visual Studio 2019.



**Game**

Klasa Game jest główną klasą gry i nie jest przypisana do żadnego konkretnego folderu. Pełni kluczową rolę w funkcjonowaniu gry. Jest odpowiedzialna za dostarczanie najważniejszych bibliotek, takich jak SDL2, wszystkim innym klasom w grze. Współpracuje również z plikiem Main.cpp, które razem odpowiadają za działanie gry

Klasa Game tworzy nieskończoną pętlę gry, która kontroluje przebieg rozgrywki. Odpowiada za inicjalizację i zarządzanie GameLevelManager’em, który jest odpowiedzialny za tworzenie i zarządzanie poziomami gry.

Klasa Game posiada atrybuty, które odpowiadają za wymiary aplikacji i gry. Mogą to być na przykład szerokość i wysokość okna gry, rozdzielczość ekranu itp.

Jednak najważniejsze atrybuty klasy Game to static SDL\_Renderer\* mainGameRender oraz static SDL\_Event mainGameEvent. Są to obiekty biblioteki SDL2, które są niezbędne do renderowania grafiki i obsługi zdarzeń w grze. Atrybuty te są statyczne, co oznacza, że są dostępne dla wszystkich instancji klasy Game i mogą być używane przez inne klasy w celu renderowania grafiki i obsługi zdarzeń.

**Folder SDL2\_Managers:**

Folder SDL2\_Managers zawiera klasy, które dostarczają niezbędne narzędzia do zarządzania czcionkami, obrazami i tekstem w grze. Dzięki nim można w sposób intuicyjny i wygodny renderować napisy i obrazy na ekranie, dodając w ten sposób estetyczny i funkcjonalny aspekt do pikselowej gry 2D.  
Oto opis tych klas:

**FontTextureManager**

Klasa FontTextureManager to prawdziwy magik czcionek w grze. Jest ona odpowiedzialna za załadowanie odpowiedniej czcionki z pliku i przygotowanie tekstury, która może być wykorzystana do wyświetlania tekstu na ekranie gry. FontTextureManager umożliwia łatwe manipulowanie rozmiarem, kolorem i stylem tekstu, dając Ci kontrolę nad wyglądem napisów w grze. Ta klasa jest niezastąpiona, gdy chcesz wyświetlić informacje dla gracza, takie jak wyniki, wskazówki czy dialogi postaci.

**ImageTextureManager**

Klasa ImageTextureManager to niezawodny obrazowy pomocnik, który umożliwia łatwe ładowanie i wyświetlanie obrazów na ekranie gry. Załaduje obraz z odpowiedniej ścieżki, a następnie przygotuje go do renderowania na ekranie. Dzięki ImageTextureManagerowi możesz wyświetlać tła, postacie, przedmioty i wiele innych elementów graficznych w swojej grze. Oferuje on również funkcje skalowania, przycinania i manipulowania wyglądem obrazu, aby dostosować go do potrzeb gry.

**TextObject**

Klasa TextObject to magiczny obiekt tekstowy, który reprezentuje napis, który można wyświetlić na ekranie gry. Ten obiekt korzysta z FontTextureManager i ImageTextureManager, aby przetworzyć tekst na teksturę i wyświetlić go w odpowiednim miejscu na ekranie. Dzięki TextObject możesz tworzyć dynamiczne napisy, takie jak wyniki, punktacja czy informacje o zdrowiu postaci. Możesz kontrolować styl, kolor i rozmiar tekstu, aby dostosować go do wyglądu gry i tworzyć wspaniałe efekty wizualne.

**Folder MapUtils**

Folder SDL2\_ MapUtils zawiera klasy, które dostarczają narzędzi i funkcji potrzebnych do zarządzania mapami w grze. Dzięki nim możesz tworzyć mapy za pomocą plików tekstowych, generować losowe poziomy gry i dostosowywać położenie bohatera oraz elementów w grze. To umożliwia tworzenie różnorodnych i ekscytujących poziomów, które stanowią kluczową część rozgrywki i zapewniają graczom interesujące wyzwania do pokonania.  
Oto opis tych klas:

**\_VirtualMap**

Klasa \_VirtualMap jest interfejsem dla wszystkich map w grze. Wszystkie klasy dziedziczące po \_VirtualMap będą miały dostęp do ścieżek do niezbędnych grafik potrzebnych do zbudowania mapy. Klasa ta umożliwia również konstrukcję mapy z pliku tekstowego, zapisywanie mapy do pliku tekstowego oraz generowanie losowo wygenerowanej mapy. Dzięki temu interfejsowi klasy dziedziczące będą miały wspólne funkcjonalności i będą mogły operować na mapach w sposób spójny i elastyczny.

**BackgroundMap**

Klasa BackgroundMap zajmuje się konkretną konstrukcją poziomów gry. Korzysta ona w pełni z interfejsu \_VirtualMap. Posiada informacje dotyczące położenia środka mapy, położenia bohatera gracza, przesunięcia bohatera względem środka mapy i wiele innych. Dzięki tym informacjom BackgroundMap może dokładnie określić, jak zbudować poziom gry na podstawie dostępnych danych. Może tworzyć różnorodne elementy mapy, takie jak tła, platformy, przeszkody, przedmioty do zebrania i wiele więcej, aby zapewnić bogate i zróżnicowane środowisko gry.

**Folder LevelUtils**  
Folder LevelUtils zawiera klasy, które są odpowiedzialne za tworzenie i zarządzanie poziomami gry. Umożliwiają tworzenie dynamicznych i zróżnicowanych poziomów, które dostarczają różnorodności i wyzwań dla graczy.  
Oto opis tych klas:

**GameLevel**

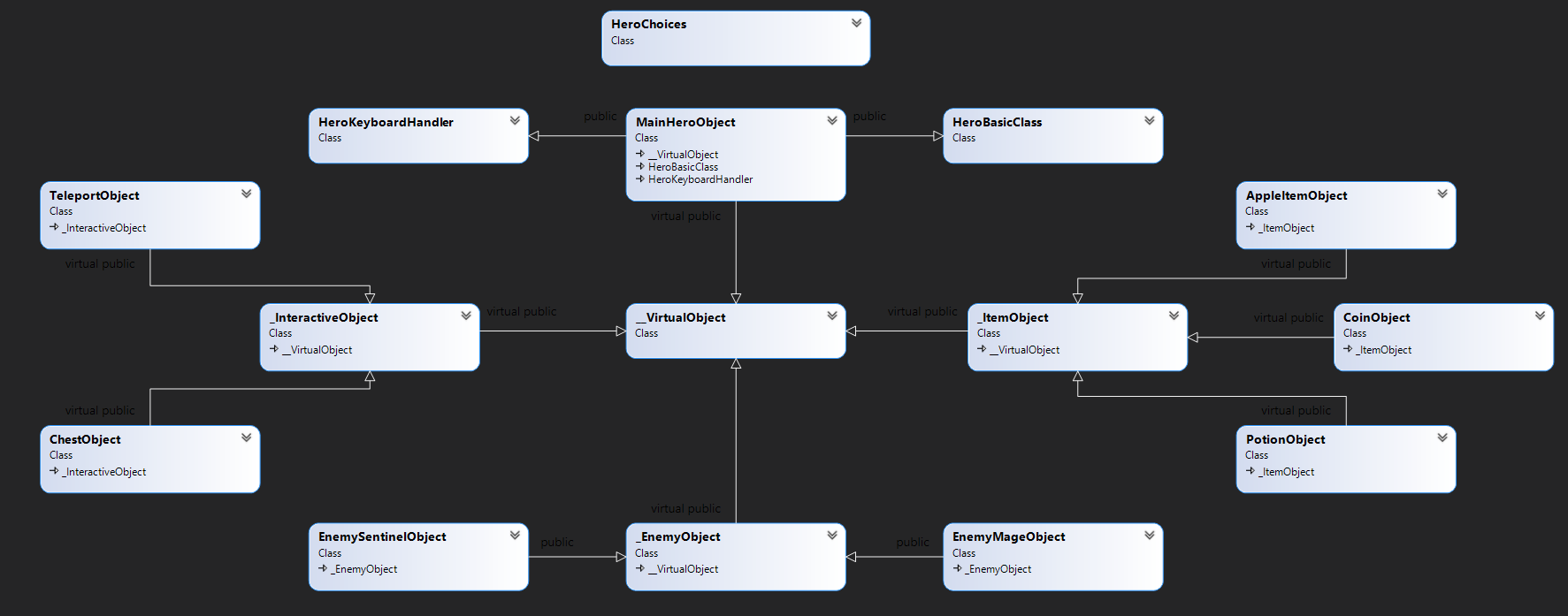
Klasa GameLevel reprezentuje konkretny poziom gry. Posiada unikalne ID, które identyfikuje dany poziom, oraz wektory zawierające wszystkie GameObjects (obiekty gry). GameObjects to obiekty takie jak przeciwnicy, przedmioty i obiekty interaktywne. Obiekty gry generowane są w losowej ilości i na losowych pozycjach, co sprawia, że każda rozgrywka jest inna i bardziej zróżnicowana.

**GameLevelManager**

Klasa GameLevelManager posiada wektor zbudowany z obiektów klasy GameLevel. Jest odpowiedzialna za zarządzanie poziomami gry. Ta klasa obsługuje zderzenia między GameObjects, co oznacza, że zajmuje się wywoływaniem odpowiednich akcji w zależności od interakcji między obiektami. Na przykład może kontrolować walkę z przeciwnikami, zbieranie monet czy przechodzenie przez portale. GameLevelManager pełni kluczową rolę w kontrolowaniu przebiegu gry i zarządzaniu poziomami.

### 5.2 Opis poszczególnych klas – część 2

Wygenerowany przez rozszerzenie do Visual Studio 2019.



**Folder GameObjects**

Folder GameObjects zawiera obiekty gry, które współdziałają z postacią gracza. Jest to kluczowy folder, który ułatwia rozwój i rozbudowę gry poprzez dostarczanie interfejsów i wartości generowanych losowo, co urozmaica każdą rozgrywkę.

Zawiera w sobie podfoldery SpecEnemyObjects, SpecInteractiveObjects i SpecItemObjects, które z kolei zawierają podklasy dziedziczące po odpowiednich interfejsach (kolejno \_EnemyObject, \_InteractiveObject, \_ItemObject). Każdy podfolder zawiera konkretne typy obiektów, takie jak różni przeciwnicy, interaktywne obiekty i przedmioty. Te klasy specjalizowane mogą dostosowywać zachowanie i właściwości obiektów do konkretnych potrzeb gry, tworząc różnorodność i unikalność wśród elementów rozgrywki.

Zawiera również w sobie podfolder HeroUtils, który zawiera klasy powiązane z MainHeroObject. HeroBasicClass przechowuje bazowe statystyki bohatera i pomaga w obsłudze kolizji. HeroChoices obsługuje wybory i akcje podejmowane przez gracza, a HeroKeyboardHandler zbiera informacje o naciśniętych przyciskach na klawiaturze, umożliwiając grze reagowanie na działania gracza. Te klasy w HeroUtils są istotne dla sterowania bohaterem gracza i umożliwiają interakcję z innymi obiektami w grze.

Poniżej znajduje się opis poszczególnych klas w tym folderze:

**\_\_VirtualObject**

Klasa \_\_VirtualObject jest interfejsem dla wszystkich obiektów gry. Posiada metody i atrybuty, które umożliwiają komunikację z ImageTextureManager, co pozwala nam na renderowanie obiektów na ekranie gry. Każdy obiekt gry musi posiadać metody Update i Render, które odpowiednio aktualizują stan obiektu i rysują go na ekranie. Dzięki temu interfejsowi, klasy dziedziczące po \_\_VirtualObject mogą definiować różne typy obiektów w grze i dostosowywać ich zachowanie w zależności od potrzeb.

**\_EnemyObject**

Klasa \_EnemyObject jest interfejsem dla wszystkich przeciwników w grze. Posiada metody i atrybuty odpowiedzialne za poruszanie się przeciwników oraz może generować różne parametry dla konkretnych obiektów. Dodatkowo, klasa ta posiada atrybuty umożliwiające rozróżnienie siły przeciwników, co pozwala na wprowadzenie różnych poziomów trudności w grze. Przeciwnicy mogą mieć różne umiejętności, takie jak atakowanie gracza, obrona przed atakami, czy poruszanie się w określonych wzorcach. Dzięki klasie \_EnemyObject, możesz tworzyć różnorodne przeciwników, które stanowią wyzwanie dla gracza i dodają dynamiki do rozgrywki.

**\_InteractiveObject**

Klasa \_InteractiveObject jest interfejsem dla wszystkich obiektów interaktywnych w grze. Posiada metody i atrybuty odpowiedzialne za interakcję z tymi obiektami. Dodatkowo, klasa ta posiada atrybuty umożliwiające rozróżnienie wymaganego wyzwania, czyli jakie statystyki gracza muszą być spełnione, aby móc skorzystać z danego obiektu. Obiekty interaktywne mogą mieć różne funkcje, takie jak otwieranie skrzyń, aktywowanie teleportów, czy wyzwalanie innych zdarzeń specjalnych. Dzięki klasie \_InteractiveObject, możesz tworzyć różnorodne obiekty, z którymi gracz może interakcjonować, co dodaje głębi i możliwości w rozgrywce.

**\_ItemObject**

Klasa \_ItemObject jest interfejsem dla wszystkich przedmiotów, które mogą być automatycznie zebrane przez gracza. Posiada metody i atrybuty odpowiedzialne za efekty zbieranych przedmiotów. Dodatkowo, klasa ta posiada atrybuty umożliwiające określenie siły efektów zebranych przedmiotów, takich jak przywracanie zdrowia, zadawanie obrażeń, czy poprawianie statystyk gracza. Przedmioty mogą mieć różne właściwości i wpływać na rozgrywkę w różnorodny sposób. Dzięki klasie \_ItemObject, możesz tworzyć różne przedmioty, które gracze mogą zbierać, co daje im nagrody i możliwość rozwijania swojej postaci.

**MainHeroObject**

Klasa MainHeroObject reprezentuje głównego bohatera gracza. Skupia w sobie inne pomocnicze klasy z podfolderu HeroUtils, które pomagają w zarządzaniu postacią gracza. Może przechowywać bazowe statystyki bohatera (np. zdrowie, punkty doświadczenia, siłę, zwinność, inteligencję) oraz obsługiwać kolizje z innymi obiektami. Dodatkowo, klasa ta może reagować na wybory i akcje podejmowane przez gracza za pomocą maszyny stanów, co pozwala na różnorodne zachowanie bohatera w zależności od kontekstu gry. MainHeroObject jest kluczową klasą reprezentującą gracza i umożliwiającą sterowanie nim w grze.

**Podfolder SpecEnemyObjects**

Podfolder SpecEnemyObjects zawiera podklasy \_EnemyObject, czyli różne typy przeciwników w grze, które mają unikalne cechy i zachowania w grze. Gracz będzie musiał podejmować odpowiednie działania, takie jak zebranie jabłek, aby poprawić relacje z przeciwnikami lub rozwijać swoje statystyki, aby pokonać ich. To wprowadza element strategii i wyzwań dla gracza, zachęcając go do rozwijania swojej postaci i podejmowania odpowiednich decyzji w rozgrywce.  
Oto opis poszczególnych klas w tym podfolderze:

**EnemyMageObject**

Klasa EnemyMageObject reprezentuje przeciwnika-maga. Na początku gry ta postać nie będzie atakować bohatera gracza. Jednak wraz z zebraniem każdej mikstury oraz otwarciem skrzyni, stosunki z tymi przeciwnikami będą się pogarszać, aż w końcu będą oni atakować gracza i zadawać podwójne obrażenia. Zbieranie jabłek poprawia relacje z nimi. Aby pokonać przeciwników-magów, gracz będzie musiał posiadać odpowiednio wysoką statystykę siły (Strength).

**EnemySentinelObject**

Klasa EnemySentinelObject reprezentuje przeciwnika-strażnika. Na początku gry ta postać nie będzie atakować bohatera gracza. Jednak za każdym razem, gdy bohater przejdzie przez teleport, stosunki z tymi przeciwnikami będą się pogarszać, aż w końcu będą oni atakować gracza i zadawać podwójne obrażenia. Podobnie jak w przypadku przeciwników-magów, zbieranie jabłek poprawia relacje z nimi. Aby pokonać przeciwników-strażników, gracz będzie musiał posiadać odpowiednio wysoką statystykę siły (Strength).

**Podfolder SpecInteractiveObjects**

Podfolder SpecInteractiveObjects zawiera podklasy \_InteractiveObject, czyli różne typy obiektów interaktywnych w grze, które wpływają na rozgrywkę i relacje z przeciwnikami. Gracz będzie musiał rozważnie korzystać z tych obiektów, biorąc pod uwagę swoje statystyki i konsekwencje wyborów, które mogą mieć wpływ na rozwój gry.  
Oto opis poszczególnych klas w tym podfolderze:

**ChestObject**

Klasa ChestObject reprezentuje interaktywny obiekt w postaci skrzyni. Gracz może otworzyć tę skrzynię, używając odpowiedniego przycisku i posiadając odpowiednio wysoką statystykę zwinności (Agility). Po otwarciu skrzyni, gracz otrzymuje wzmocnienie losowej statystyki, jednak relacje z przeciwnikami-magami się pogarszają. Dodatkowo, gracz otrzymuje określoną ilość punktów wyniku (ScorePoints).

**TeleportObject**

Klasa TeleportObject reprezentuje interaktywny obiekt w postaci teleportu. Gracz może podróżować za pomocą tego teleportu, używając odpowiedniego przycisku i posiadając odpowiednio wysoką statystykę inteligencji (Intelligence). Korzystając z teleportu, gracz może przenieść się do kolejnych poziomów gry. Jednak korzystanie z teleportu pogarsza relacje z przeciwnikami-strażnikami.

**Podfolder SpecItemObjects**

Podfolder SpecItemObjects zawiera podklasy \_ItemObject, czyli różne typy przedmiotów do automatycznego zbierania w grze, które można zbierać w grze. Każdy z tych przedmiotów ma unikalne właściwości i wpływ na rozgrywkę, zarówno pod względem zdrowia, relacji z przeciwnikami, jak i punktów wyniku. Gracz będzie musiał decydować, jakie przedmioty zbierać, biorąc pod uwagę ich efekty i konsekwencje w grze.  
Oto opis poszczególnych klas w tym podfolderze:

**AppleItemObject**

Klasa AppleItemObject reprezentuje zbieralny przedmiot w postaci jabłka. Zbieranie jabłek może mieć różne efekty, takie jak przywracanie zdrowia lub zadawanie obrażeń. Jednak najważniejsze jest to, że zbieranie jabłek poprawia relacje z każdym typem przeciwnika. To oznacza, że zebranie jabłek może wpłynąć na postawę przeciwników wobec gracza.

**CoinItemObject**

Klasa CoinItemObject reprezentuje zbieralny przedmiot w postaci monety. Zbieranie monet przynosi określoną ilość punktów wyniku (ScorePoints). Monety są wartościowe i mogą być wykorzystane do zwiększenia wyniku gracza.

**PotionItemObject**

Klasa PotionItemObject reprezentuje zbieralny przedmiot w postaci mikstury. Zbieranie mikstur zawsze przywraca zdrowie gracza. Jednak warto zauważyć, że zbieranie mikstur pogarsza relacje z przeciwnikami-magami. Zbierając mikstury, gracz może odnieść korzyści zdrowotne, ale może również narazić się na nieprzychylność przeciwników.

**Podfolder HeroUtils**

Podfolder HeroUtils zawiera klasy powiązane z klasą MainHeroObject (która reprezentuje głównego gracza) i pomagają w zarządzaniu statystykami, wyborami oraz obsługą klawiatury związanej z głównym bohaterem gracza. Te klasy są istotne dla funkcjonowania bohatera gracza i umożliwiają interakcję z grą w sposób, który odzwierciedla decyzje i działania gracza.  
Oto opis poszczególnych klas w tym podfolderze:

**HeroBasicClass**

Klasa HeroBasicClass reprezentuje bazowe statystyki bohatera gracza. Może zawierać informacje takie jak HeroHealthPoints (punkty zdrowia bohatera), ScorePoints (punkty wyniku), Strength (siła), Agility (zwinność), Intelligence (inteligencja) itp. Ta klasa pomaga w zarządzaniu statystykami bohatera oraz rozpatrywaniu kolizji z innymi obiektami w grze.

**HeroChoices**

Klasa HeroChoices reprezentuje wybory i akcje podejmowane przez gracza jako maszynę stanów. Może zawierać informacje o relacjach gracza z przeciwnikami, takich jak zmiany w tych relacjach w zależności od podjętych akcji. Ta klasa pomaga śledzić i zarządzać interakcjami bohatera gracza z otoczeniem i przeciwnikami.

**HeroKeyboardHandler**

Klasa HeroKeyboardHandler zbiera informacje o naciśniętych przyciskach na klawiaturze przez gracza i przekazuje je dalej w celu odpowiedniego zareagowania w grze. Dzięki tej klasie gra może odczytywać intencje gracza związane z poruszaniem się i wykonywaniem działań przez bohatera. HeroKeyboardHandler pomaga w obsłudze sterowania bohaterem gracza na podstawie akcji gracza na klawiaturze.

# Opis wykorzystanych technologii

W tym rozdziale przedstawiamy technologie, które zostały wykorzystane podczas projektowania i implementacji naszej pikselowej gry. W ramach pracy skorzystaliśmy z następujących technologii: C++ 14, Microsoft Visual Studio 2019 oraz SDL2 wraz z rozszerzeniami SDL2 Image i SDL2 TTF. Poniżej znajduje się bardziej szczegółowy opis każdej z tych technologii.

Dzięki wykorzystaniu poniższych technologii byliśmy w stanie skonstruować solidne fundamenty dla naszej pikselowej gry, zapewniając nie tylko efektywność implementacji, ale także wygodę programowania i elastyczność w realizacji naszych założeń.

**Język programowania - C++ wersja 14**

Język programowania C++ stanowił podstawę naszego projektu. Zdecydowaliśmy się na użycie wersji C++ 14 ze względu na jej rozwiniętą składnię, możliwość programowania obiektowego oraz efektywne zarządzanie pamięcią. Język ten zapewnia również wsparcie dla wielu bibliotek i narzędzi, co umożliwiło nam wygodne korzystanie z innych technologii w naszej grze.

**Środowisko programistyczne - Microsoft Visual Studio 2019**

Do implementacji naszej gry wybraliśmy środowisko programistyczne Microsoft Visual Studio 2019. Jest to zaawansowane narzędzie, które zapewnia bogate funkcjonalności, takie jak edytor kodu, debugger, profiler i narzędzia do zarządzania projektem. Wybraliśmy Visual Studio ze względu na jego szerokie wsparcie dla języka C++, bogatą dokumentację i rozbudowaną społeczność, co ułatwiło nam pracę nad projektem i rozwiązywanie ewentualnych problemów.

**Biblioteka programistyczna - Simple DirectMedia Layer (SDL2)**

SDL2 (Simple DirectMedia Layer) jest biblioteką programistyczną, która dostarcza zestaw narzędzi i funkcji do tworzenia aplikacji multimedialnych, w tym również gier. Wybraliśmy SDL2 jako nasz główny silnik graficzny ~~i dźwiękowy~~ ze względu na jego prostotę, wydajność i przenośność na różne platformy. Dzięki SDL2 mogliśmy łatwo zarządzać oknem gry, obsługiwać wejście od użytkownika (klawiatura), ~~odtwarzać dźwięki i muzykę~~ oraz renderować grafikę w czasie rzeczywistym.

**Rozszerzenia biblioteki programistycznej - Simple DirectMedia Layer Image (SDL Image) oraz Simple DirectMedia Layer True Type Fonts (SDL TTF)**

W naszej grze wykorzystaliśmy także rozszerzenia SDL2 Image i SDL2 TTF. SDL2 Image umożliwiło nam łatwe wczytywanie różnych formatów plików graficznych, takich jak PNG czy JPEG, co pozwoliło nam na wykorzystanie różnorodnych grafik w naszej grze. Natomiast SDL2 TTF umożliwiło nam renderowanie tekstu w grze, zapewniając szeroki wybór czcionek i możliwość personalizacji interfejsu użytkownika.

# Opis aplikacji

W tym rozdziale przedstawimy szczegółowy opis naszej pikselowej gry 2D. Przedstawimy główne elementy i funkcjonalności gry, interfejs użytkownika, zasoby graficzne i dźwiękowe, oraz inne istotne aspekty aplikacji.

Opisując aplikację, ważne jest, aby zapewnić czytelnikowi pełne zrozumienie jej funkcji, wyglądu i działań. Możemy wspierać opisy przykładami graficznymi, kodem źródłowym lub innymi elementami wizualnymi, które pomogą wizualizować opisywane aspekty gry.

## ****7.1 Interfejs użytkownika oraz funkcjonalności gry****

Opiszemy interfejs użytkownika naszej gry, czyli to, co gracz będzie widział na ekranie i jak będzie na nią oddziaływał.

* Ekran gry:  
  Opiszemy teraz, jak będzie wyglądać ekran gry, czyli plansza, na której rozgrywać się będą akcje.
  + Bazowe rozmiary ekranu gry to 1312x928 pikseli.
  + Plansza gry jest większa od widocznego na ekranie gry. Gracz może poruszać się swoją postacią aż do granic planszy gry.
* Tekstowy interfejs gracza:   
  Używając odpowiedniego klawisza, gracz może wyświetlić na ekranie swoje obecne statystyki, w skład których wchodzą między innymi:
  + „HeroHealthPoints” – Punkty Zdrowia Gracza. Określają poziom życia postaci gracza. Jeśli spadną do 0, gra zostanie zakończona i zresetowana. Gracz może tracić punkty życia. Wartość startowa to 25.
  + „ScorePoints” – Punkty Wynikowe. Są to punkty zdobyte przez gracza, na ich podstawie gracze mogą określać, który z graczy lepiej sobie poradził. Wartość startowa to 0.
  + „Strength” – Siła. Ta statystyka odpowiada za pokonywanie przeciwników, im wyższa, tym silniejszych wrogów postać gracza może pokonanć. Wartość startowa to 10.
  + „Agility” – Zwinność. Ta statystyka odpowiada odpowiada za otwieranie skrzyń, im wyższa inteligencja, tym lepsze łupy można zbierać. Wartość startowa to 10.
  + „Intelligence” – Inteligencja. Ta statystyka odpowiada za przechodzenie między poziomami, wystarczająco wysoka inteligencja pozwala przechodzić pomiędzy większą ilością portali. Wartość startowa to 10.
  + „RelationshipWithMages” – Stosunki względem Czarodziei. Statystyka ta określa jak pozytywnie bądź negatywnie są nastawieni wobec niego Czarodzieje.
  + „RelationshipWithSentinels” – Stosunki względem Strażników. Statystyka ta określa jak pozytywnie bądź negatywnie są nastawieni wobec niego Strażnicy.

Przedstawimy teraz szczegółowy opis funkcjonalności, które zostały zaimplementowane w naszej grze.

* Automatycznie i losowo generowana plansza:  
  Przy każdym uruchomieniu gry, plansza gry jest losowo generowana. Oznacza to, że rozmieszczenie i ilość obiektów, zawsze będzie się różnić.
* Mechanika ruchu bohatera gracza:  
  Postać gracza, może poruszać się po całej planszy – góra, dół, lewo, prawo - używając odpowiednich klawiszy.
* Statystki postaci/ gracza:  
  Gracz może ulepszać swoje statystyki otwierając skrzynie, zbierając monety, lub pokonując wrogów. Rozróżniamy dwa typy statystyk:  
  Statystyki gracza:
  + „ScorePoints”

Statystyki gracza:

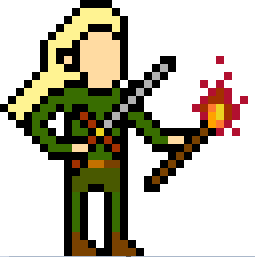
* + „HeroHealthPoints”
  + „Strength”
  + „Agility”
  + „Intelligence”
  + „RelationshipWithMages”
  + „RelationshipWithSentinels”
* System kolizji i interakcje z otoczeniem:  
  Cześć kolizji następuje automatycznie, jednak niektóre obiekty wymagają od gracza ręcznej interkacji, w innym wypadku stanowią po prostu tło. Gracz może prowadzić interakcje z otaczającymi go elementami naciskając odpowiedni klawisz.
  + Przykłady automatycznej interkacji:
    - Walka.
    - Zbieranie monet.
    - Zbieranie mikstur.
  + Przykład ręcznej interakcji:
    - Otworzenie skrzyni.
    - Przejście przez portal.
* Algorytmy sztucznej inteligencji – maszyna stanów:  
  Początkowo przeciwnicy będą wobec gracz całkowicie neutralni, pomimo faktu, że poruszać się będą po planszy nie wywołują graczowi żadnej krzywdy. Jednak:
  + Gdy gracz użyje portalu, Strażnicy zaczną zadawać mu obrażenia. Za każdym kolejnym razem gdy gracz użyje portalu, Strażnicy będą bardziej negatywni w stosunku do gracza. Ostatecznie dojdzie do momentu w którym będą zadawać podwójne obrażenia.
  + Gdy gracz zbierze miksturę, Czarodzieje staną się bardziej negatywni wobec gracza. Początkowo zaczną go atakować, a jeśli gracz zbierze za dużo mikstur, zaczną zadawać mu podwójne obrażenia.
  + Gdy gracz zbierze Złote Jabłko, losowa relacja (z Czarodziejami lub Strażnikami ulega poprawie, ostatecznie wrogowie mogą nawet znów nie atakować gracza)/

Poniżej znajduje się instrukcja sterowania grą:

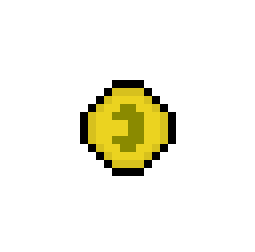
|  |
| --- |
| Witaj w naszej pikselowej grze 2D! Poniżej przedstawiamy instrukcję, która pomoże Ci rozpocząć przygodę i poruszać się po świecie gry.  Poruszanie się:   * Użyj klawiszy "W", "A", "S" i "D" do poruszania się swoją postacią. Klawisz "W" pozwoli Ci poruszać się do przodu, "A" - w lewo, "S" - do tyłu, a "D" - w prawo.   Interakcja z przedmiotami:   * Aby zinterakcjonować z przedmiotami w grze, naciśnij klawisz "F". To pozwoli Ci na zbieranie przedmiotów, otwieranie drzwi, czy aktywowanie innych elementów interaktywnych.   Przyśpieszenie i skradanie się:   * Klawisz "Shift" pozwala na przyśpieszenie postaci, dając Ci większą szybkość poruszania się. Możesz go używać, gdy chcesz szybko przemieszczać się po świecie gry. * Jeśli chcesz przemieszczać się ostrożnie i niezauważenie, naciśnij klawisz "Ctrl". To spowolni Twoją postać i pozwoli na skradanie się w cichy sposób.   Wyświetlanie statystyk postaci:   * Aby zobaczyć obecne statystyki swojej postaci, naciśnij klawisz "Tab". Wyświetli się ekran z informacjami na temat zdrowia, poziomu, doświadczenia, czy innych ważnych parametrów Twojej postaci.   Wyjście z gry:   * Jeśli chcesz zakończyć grę, naciśnij klawisz "Esc". To spowoduje wyłączenie gry i powrót do menu głównego.   Teraz, gdy znasz instrukcję, możesz cieszyć się grą i odkrywać świat, który przed Tobą leży! Powodzenia i miłej zabawy! |

## 7.2 Zasoby graficzne

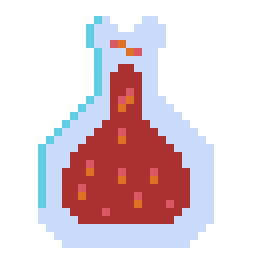
Poniżej znajduje się lista rysowanych przeze mnie grafik.



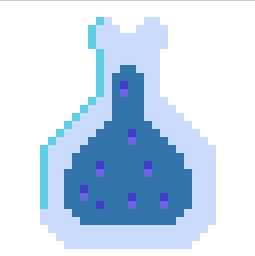
Rysunek 1 Bohater gracza



Rysunek 2 Moneta



Rysunek 3 Mikstura wzmacniająca nr 1



Rysunek 4 Mikstura wzmacniająca nr 2



Rysunek 5 Złote jabłko



Rysunek 6 Skrzynia nr 1 (drewniana)



Rysunek 7 Skrzynia nr 2 (żelazna)



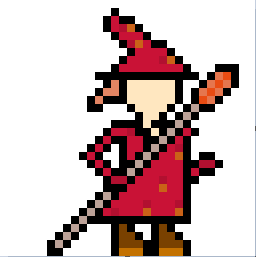
Rysunek 8 Skrzynia nr 3 (otworzona)



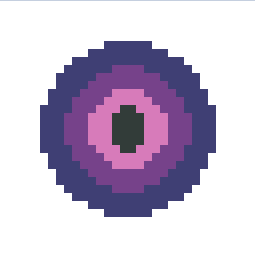
Rysunek 9 Wróg czarodziej nr 1 (łatwy)



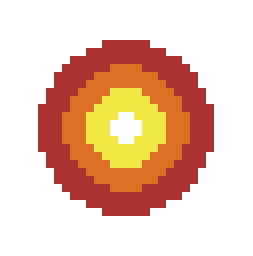
Rysunek 10 Wróg czarodziej nr 2 (średni)



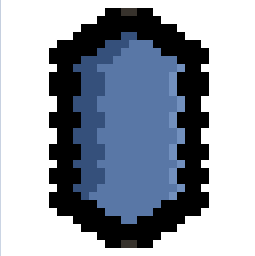
Rysunek 11 Wróg czarodziej nr 3 (trudny)



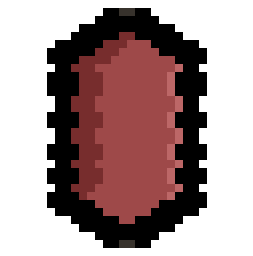
Rysunek 12 Wróg strażnik nr 1 (najsilniejszy)



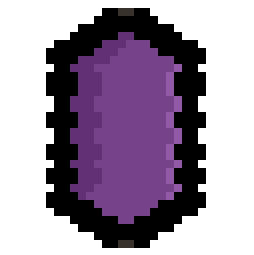
Rysunek 13 Wróg strażnik nr 2 (najsilniejszy)



Rysunek 14 Portal nr 1



Rysunek 15 Portal nr 2



Rysunek 16 Portal nr 3

# Podsumowanie

Podsumowanie stanowi zakończenie pracy inżynierskiej, podsumowując jej główne cele, wyniki oraz wnioski płynące z przeprowadzonych badań i implementacji. Praca miała na celu stworzenie gry komputerowej, której głównym elementem jest interakcja gracza z różnymi obiektami wirtualnego świata. Głównym celem było zaprojektowanie i implementacja hierarchii klas obiektów gry oraz zarządzanie nimi w kontekście tworzenia poziomów i interakcji z graczem.

W ramach pracy przeprowadzono szczegółową analizę wymagań oraz ustalono strukturę projektu. Przyjęto hierarchię klas, która obejmowała główne elementy gry, takie jak obiekty gry, przeciwnicy, obiekty interaktywne oraz przedmioty do zbierania. Każda klasa posiadała odpowiednie metody i atrybuty, które umożliwiały ich interakcję z graczem oraz generowanie różnorodnych efektów w trakcie rozgrywki. Wszystkie te klasy zostały zorganizowane w odpowiednie foldery, aby zachować porządek i ułatwić zarządzanie projektem.

W ramach folderu "GameObjects" zdefiniowano klasę VirtualObject, która stanowiła podstawowy interfejs dla wszystkich obiektów gry. Klasa ta zawierała niezbędne metody i atrybuty, umożliwiające komunikację z menadżerem tekstur i renderowanie obiektów w grze. Klasy \_EnemyObject, \_InteractiveObject i \_ItemObject były interfejsami dla poszczególnych typów przeciwników, obiektów interaktywnych i przedmiotów do zbierania. Każda z tych klas posiadała specyficzne metody i atrybuty, które określały ich zachowanie i właściwości.

Dodatkowo, w folderze "HeroUtils" znajdowały się klasy powiązane z głównym bohaterem gracza, takie jak HeroBasicClass, HeroChoices i HeroKeyboardHandler. HeroBasicClass definiowała bazowe statystyki bohatera, takie jak zdrowie, punkty doświadczenia i umiejętności. HeroChoices była odpowiedzialna za reprezentowanie wyborów i akcji podejmowanych przez gracza. HeroKeyboardHandler natomiast obsługiwał naciśnięcia przycisków na klawiaturze i przekazywał te informacje do gry.

Folder "LevelUtils" skupiał klasy związane z zarządzaniem poziomami gry. Klasa GameLevel reprezentowała konkretny poziom gry i zawierała identyfikator oraz wektory wszystkich obiektów gry. Obiekty gry były generowane w sposób losowy, aby zapewnić zróżnicowanie rozgrywki w każdej sesji. Klasa GameLevelManager natomiast odpowiadała za zarządzanie kolizjami między obiektami gry, takimi jak walka z przeciwnikami czy zebranie przedmiotów.

Główna klasa gry, Game, stanowiła kluczowy punkt zarządzający i dostarczała niezbędne biblioteki, takie jak SDL2, wszystkim innym klasom. Odpowiadała również za uruchomienie nieskończonej pętli gry oraz inicjalizację GameLevelManagera. Istotnymi atrybutami były również obiekty statyczne SDL\_Renderer\* mainGameRender i SDL\_Event mainGameEvent, które umożliwiały interakcję z systemem operacyjnym i obsługę zdarzeń.

Podsumowując, praca inżynierska przyniosła sukces w zakresie stworzenia gry komputerowej, która umożliwia interakcję gracza z różnymi obiektami wirtualnego świata. Hierarchia klas obiektów gry oraz zarządzanie nimi w kontekście tworzenia poziomów i interakcji z graczem zostały z powodzeniem zaimplementowane. Efektem tego jest grywalna i zróżnicowana rozgrywka, która oferuje interaktywność i wyzwanie dla gracza.

Praca inżynierska przyczyniła się do rozwinięcia umiejętności programistycznych autora oraz zdobycia praktycznej wiedzy z zakresu tworzenia gier komputerowych. Zastosowane podejście hierarchiczne do projektowania obiektów gry okazało się efektywne i umożliwiło łatwą rozbudowę oraz modyfikację gry w przyszłości.

Wnioski płynące z pracy wskazują na sukces osiągnięcia zamierzonych celów projektu. Praca inżynierska stanowi solidną podstawę do dalszego rozwoju gry, rozszerzania funkcjonalności oraz optymalizacji. Możliwości rozwoju obejmują dodanie nowych klas obiektów, tworzenie bardziej zaawansowanych poziomów oraz implementację różnorodnych mechanik gry.

Autor pracy zyskał również cenne doświadczenie w zakresie zarządzania projektem, implementacji hierarchii klas oraz interakcji obiektów w grze. Praca stanowi udane zakończenie cyklu inżynierskiego i stanowi punkt wyjścia do dalszych badań i rozwoju w dziedzinie tworzenia gier komputerowych.

Wartościowym elementem pracy inżynierskiej było również korzystanie z modelu językowego ChatGPT, który dostarczał wsparcia w generowaniu kodu oraz udzielaniu informacji na temat struktury i implementacji projektu. Model ten okazał się użytecznym narzędziem, ułatwiającym pracę i przyspieszającym proces tworzenia gry.

Praca inżynierska stanowi ważny krok w rozwoju autora w dziedzinie tworzenia gier komputerowych. Otwiera drogę do dalszych projektów i poszerzenia wiedzy w tej dziedzinie. Jest to wartościowy wkład w dziedzinę produkcji gier komputerowych, który może być wykorzystany zarówno w celach edukacyjnych, jak i rozrywkowych.