POLITECHNIKA BIAŁOSTOCKA

WYDZIAŁ INFORMATYKI

PRACA DYPLOMOWA INŻYNIERSKA

TEMAT: IMPLEMENTACJA PIKSELOWEJ GRY 2D Z WYKORZYSTANIEM BIBLIOTEKI SDL2

WYKONAWCA:

POGREBNIAK MATEUSZ

OPIEKUN PRACY DYPLOMOWEJ :

DR INŻ. URSZULA KUŻELEWSKA

BIAŁYSTOK 2023 ROK

SUBJECT OF DIPLOMA THESIS

Implementation of 2D pixel game using the SDL2 library

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

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 postać bohatera, przeciwnicy, przedmioty, interaktywne obiekty. 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.

2. Analiza problemu

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, zidentyfikowano wymagania projektowe. Ustalono, że przedmiotem pracy będzie pikselowa gra 2D, opierająca się na estetyce retro, z prostymi pikselowymi grafikami. W założeniu ma to być przyjemna dla oka gra z intuicyjną rozgrywką, która zapewni użytkownikom zarówno rozrywkę, jak i wyzwania.

Dodatkowo gra ma obejmować takie elementy jak ruch postaci, kolizje, interakcję z obiektami na planszy, jak również oceny wyników graczy.

Potencjalne wyzwania

Podczas analizy problemu, zidentyfikowane zostały również potencjalne wyzwania, z jakimi możemy się spotkać podczas implementacji gry. Wyzwania te będą 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
- Algorytmy sztucznej inteligencji – maszyna stanów
- Ocena wyników graczy

3. 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.

Rysunek 3.1 Gra Terraria

Źródło: https://pliki.ppe.pl/storage/51efd6ca39ad27262be3/51efd6ca39ad27262be3.jpg

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.

Rysunek 3.2 Gra Binding of Isaac

Źródło: https://pliki.ppe.pl/storage/80cf1a4295b5871fe6a4/80cf1a4295b5871fe6a4.jpg

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.

Rysunek 3.3 Gra Factorio

Źródło: https://pliki.ppe.pl/storage/0c2697d91847272ed53a/0c2697d91847272ed53a.jpg

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.

Rysunek 3.4 Gra Pokemon Emerald

Źródło: https://pliki.ppe.pl/storage/b11fd870d5ac0930edf4/b11fd870d5ac0930edf4.png

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

Analiza istniejących gier oraz porównanie bibliotek programistycznych pozwoliło dokonać świadomego wyboru technologii do implementacji gry. Po dokładnym zbadaniu bibliotek i porównaniu ich cech, ostatecznie autor zdecydował się skorzystać z biblioteki SDL. Istnieje kilka powodów, dla których wybrał akurat SDL jako technologię do implementacji gry.

1. Autor posiadał już doświadczenie w korzystaniu z biblioteki SDL z poprzednich projektów. Był zaznajomiony z jej API i potrafił się nią sprawnie posługiwać. To dało pewność, że będzie w stanie efektywnie korzystać z SDL podczas tworzenia gry.
2. SDL jest szeroko stosowaną i dobrze udokumentowaną biblioteką, która cieszy się dużą popularnością w społeczności deweloperów gier. Istnieje wiele

przykładów, poradników i forów, które mogą pomóc w rozwiązywaniu ewentualnych problemów podczas tworzenia gry.

3. SDL oferuje wiele funkcji, które są niezbędne do implementacji gier 2D. Biblioteka obsługuje renderowanie grafiki, obsługę dźwięku, zarządzanie oknami oraz obsługę zdarzeń, co pozwala na płynne działanie i interakcję z graczem.

Dodatkowo, w analizie gier zwrócono uwagę na elementy, takie jak rzut izometryczny, który został zaprezentowany w Pokemon Emerald. To jeden z aspektów, który autor chciał uwzględnić w swojej grze. Dzięki możliwościom renderowania grafiki oferowanym przez SDL, istniała pewność, że będzie można zaimplementować rzut izometryczny i uzyskać podobny efekt do tego obecnego w Pokemon Emerald.

4. 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.
- 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.

5. Analiza projektu

W rozdziale tym zostanie przedstawiona analiza projektu gry komputerowej. Zostaną przedstawione diagramy klas oraz opisane stworzone klasy, które stanowią fundamenty projektu.

Analiza projektu ma na celu głębsze zrozumienie struktury i organizacji kodu oraz przedstawienie relacji między poszczególnymi komponentami gry. Diagramy klas są wykorzystane do wizualizacji tych relacji oraz do identyfikacji kluczowych klas i interfejsów.

Projekt gry składa się z różnych komponentów, zorganizowanych w folderach, które odzwierciedlają ich funkcjonalność. Każdy folder zawiera klasy i interfejsy, które współdziałają ze sobą, tworząc pełne doświadczenie gry.

W folderze "GameObjects" znajdują się klasy reprezentujące obiekty gry oraz interfejsy ułatwiające rozwijanie gry. Istnieją interfejsy dla obiektów ogólnych, przeciwników, obiektów interaktywnych i przedmiotów. Kluczową klasą w tym folderze jest "MainHeroObject", reprezentująca głównego gracza gry.

Podfoldery takie jak "SpecEnemyObjects", "SpecInteractiveObjects" i "SpecItemObjects" zawierają klasy, które definiują specjalne typy przeciwników, obiektów interaktywnych i przedmiotów, dodając różnorodność i interakcję w grze.

Folder "HeroUtils" skupia się na klasach powiązanych z głównym bohaterem, takich jak "HeroBasicClass", "HeroChoices" i "HeroKeyboardHandler", które obsługują statystyki bohatera, interakcje z przeciwnikami i obsługę klawiatury.

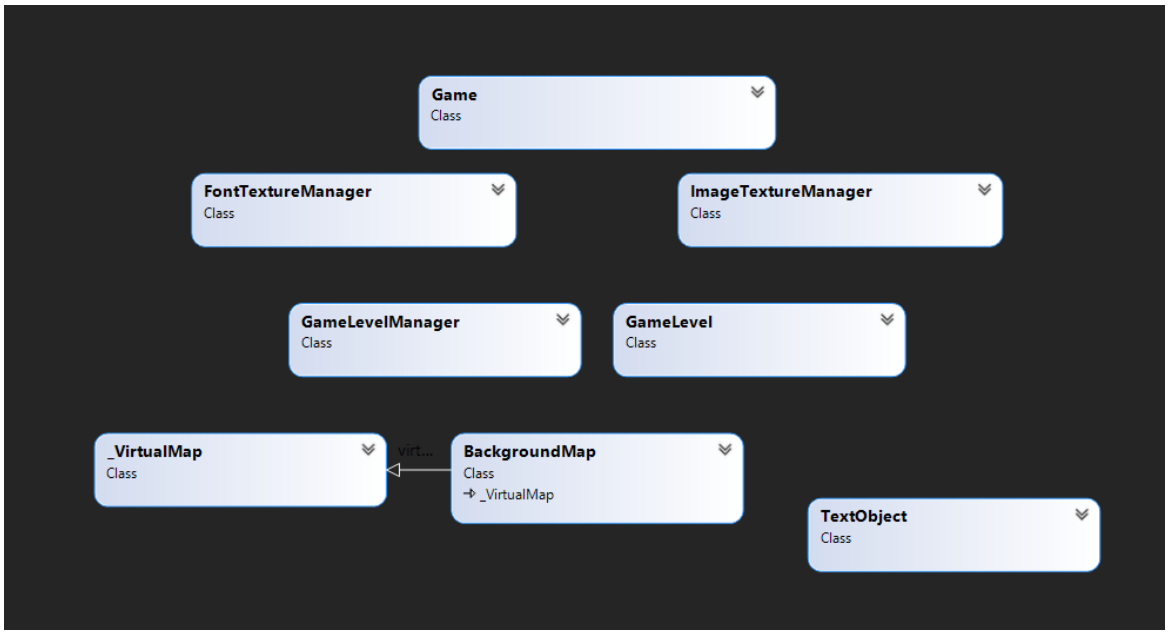
Ostatecznie, folder "LevelUtils" gromadzi klasy, które tworzą i zarządzają poziomami gry, w tym "GameLevel" i "GameLevelManager". Klasy te odpowiadają za generowanie obiektów gry w losowej ilości i miejscach, a także za zarządzanie kolizjami między nimi.

Analiza projektu dostarcza kompleksowego spojrzenia na strukturę gry oraz interakcje między jej różnymi komponentami. Diagramy klas i opisy kluczowych klas pomagają w zrozumieniu architektury projektu i stanowią solidne podstawy dla dalszego rozwoju gry.

Przejdźmy teraz do szczegółowego opisu diagramów klas oraz napisanych klas w kolejnych sekcjach tego rozdziału.

5.1 Opis poszczególnych klas – część 1

Wygenerowany przez rozszerzenie do Visual Studio 2019.



Rysunek 5.1 Diagram klas – część 1

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft 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. Posiada ona 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.

Rysunek 5.2 Klasa Game

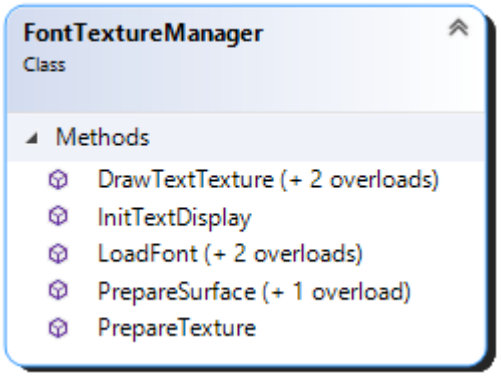
Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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 kontrolę nad wyglądem napisów w grze. Ta klasa jest niezastąpiona, gdy chcesz wyświetlić informacje dla gracza, takie jak wyniki, wskazówki czy dialogi postaci.



Rysunek 5.3 Klasa FontTextureManager

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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 ImageTextureManager można wyświetlać tła, postacie, przedmioty i wiele innych elementów graficznych w grze. Oferuje on również funkcje skalowania, przycinania i manipulowania wyglądem obrazu, aby dostosować go do potrzeb gry.

Rysunek 5.4 Klasa ImageTextureManager

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

TextObject

Klasa TextObject to magiczny obiekt tekstowy, który reprezentuje napis, który można wyświetlić na ekranie gry. Ten obiekt korzysta z klas FontTextureManager i ImageTextureManager, aby przetworzyć tekst na teksturę i wyświetlić go w odpowiednim miejscu na ekranie. Dzięki klasie TextObject można tworzyć dynamiczne napisy, takie jak

wyniki, punktacja czy informacje o zdrowiu postaci. Można również kontrolować styl, kolor i rozmiar tekstu, aby dostosować go do wyglądu gry i tworzyć wspaniałe efekty wizualne.

Rysunek 5.5 Klasa TextObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

Folder MapUtils

Folder SDL2_ MapUtils zawiera klasy, które dostarczają narzędzi i funkcji potrzebnych do zarządzania mapami w grze. Dzięki nim można 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.

Rysunek 5.6 Klasa _VirtualMap

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.7 Klasa BackgroundMap

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.8 Klasa GameLevel

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

GameLevelManager

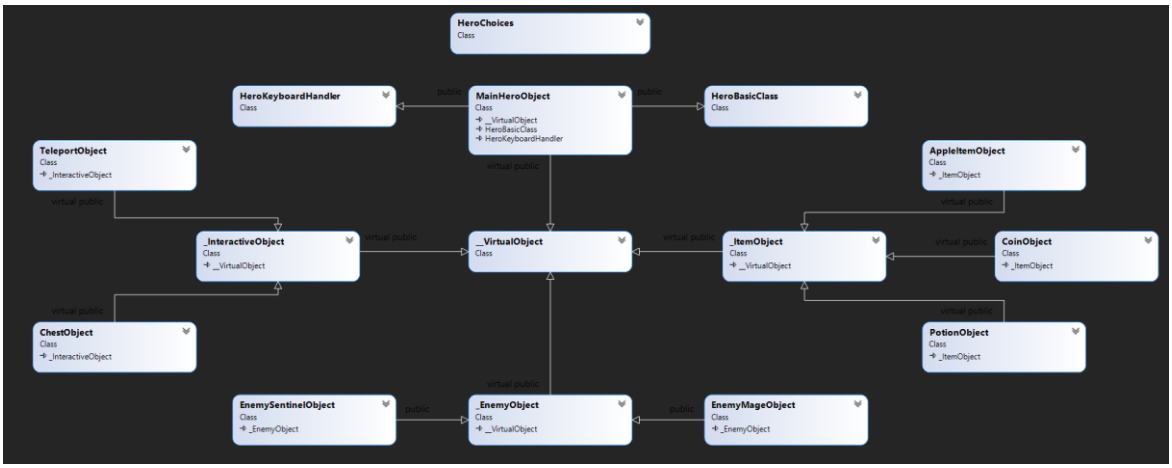
Klasa GameLevelManager posiada wektor zbudowany z obiektów klasy GameLevel. Jest odpowiedzialna za zarządzanie poziomami gry. Ta klasa obsługuje zderzenia między GameObjects, co oznacza, że zajmuje się wywoływaniem odpowiednich akcji w zależności od interakcji między obiektami. Na przykład może kontrolować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

Rysunek 5.9 Klasa GameLevelManager

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

5.2 Opis poszczególnych klas – część 2

Wygenerowany przez rozszerzenie do Visual Studio 2019.



Rysunek 5.10 Diagram klas – część 2

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft 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.

Rysunek 5.11 Klasa _VirtualObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

_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, można tworzyć różnorodne przeciwników, które stanowią wyzwanie dla gracza i dodają dynamiki do rozgrywki.

Rysunek 5.12 Klasa _EnemyObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

_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.

Rysunek 5.13 Klasa _InteractiveObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

_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.

Rysunek 5.14 Klasa _ItemObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.15 Klasa MainHeroObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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).

Rysunek 5.16 Klasa EnemyMageObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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).

Rysunek 5.17 Klasa EnemySentinelObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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).

Rysunek 5.18 Klasa TeleportObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.19 Klasa ChestObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.20 Klasa AppleItemObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.21 Klasa CoinObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

PotionItemObject

Klasa PotionItemObject reprezentuje zbieralny przedmiot w postaci mikstury. Zbieranie mikstur zawsze przywraca zdrowie gracza oraz w zależności od koloru ulepsza wybraną statystykę. 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.

Rysunek 5.22 Klasa PotionItemObject

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.23 Klasa HeroBasicClass

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.24 Klasa HeroChoices

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

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.

Rysunek 5.25 Klasa HeroKeyboardHandler

Źródło: Opracowanie własne – wygenerowane przy pomocy Microsoft Visual Studio 2019

6. Opis wykorzystanych technologii

W tym rozdziale zostaną przedstawione technologie, które zostały wykorzystane podczas projektowania i implementacji gry. W ramach pracy skorzystano z następujących technologii: C++ wersja 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 skonstruowano solidne fundamenty dla gry, zapewniając nie tylko efektywność implementacji, ale także wygodę programowania i elastyczność w realizacji założeń.

Język programowania - C++ wersja 14

Język programowania C++ stanowi podstawę projektu ze względu na rozwiniętą składnię, możliwość programowania obiektowego oraz efektywne zarządzanie pamięcią. Język ten zapewnia również wsparcie dla wielu bibliotek i narzędzi, co umożliwiło wygodne korzystanie z innych technologii w grze.

C++14 jest rozwinięciem języka C++, które wprowadza wiele nowych funkcji i udogodnień. Dzięki tym nowym możliwościom programiści mogą pisać bardziej czytelny, zwięzły i wydajny kod.

Biblioteka programistyczna - Simple DirectMedia Layer (SDL2)

SDL2 (Simple DirectMedia Layer 2) jest wieloplatformową biblioteką programistyczną, która zapewnia interfejs programistyczny do niskopoziomowych operacji wejścia/wyjścia, takich jak renderowanie grafiki, obsługa dźwięku, zarządzanie oknami oraz obsługa zdarzeń w grach i aplikacjach multimedialnych. SDL2 został stworzony w celu ułatwienia tworzenia aplikacji o wysokim wydajnościowym interfejsie użytkownika na różnych platformach.

Dzięki SDL2 programiści mogą tworzyć wydajne i platformowo niezależne aplikacje multimedialne, zwłaszcza gry, z bogatą grafiką i dźwiękiem. Biblioteka ta jest szeroko stosowana w branży gier komputerowych oraz w tworzeniu aplikacji multimedialnych na różnych platformach.

Ważne cechy biblioteki SDL2:

- Grafika: SDL2 umożliwia renderowanie grafiki w pełnym ekranie lub w oknach. Biblioteka obsługuje rysowanie 2D i 3D, zapewniając funkcje do manipulacji pikselami, renderowania tekstur, tworzenia animacji oraz wykorzystania sprajtów. Dodatkowo, SDL2 obsługuje akceleracje sprzętową, co przyczynia się do wydajnego renderowania grafiki.
- Wejście/wyjście: SDL2 zapewnia obsługę różnych urządzeń wejściowych, takich jak klawiatura, mysz, gamepady i joysticki. Biblioteka umożliwia rejestrowanie zdarzeń z tych urządzeń oraz precyzyjne odczytywanie stanu przycisków, ruchu myszy i osi joysticka. To pozwala na płynną interakcję użytkownika z aplikacją.
- Obsługa zdarzeń: SDL2 oferuje mechanizmy do obsługi zdarzeń, takich jak kliknięcia myszy, wciśnięcia klawiszy, zamykanie okna, zmiana rozmiaru okna itp. Dzięki temu programista może reagować na akcje użytkownika i dostosowywać działanie aplikacji.
- SDL2 jest biblioteką o otwartym kodzie źródłowym, co oznacza, że można ją swobodnie wykorzystywać i modyfikować. Posiada także obszerną dokumentację oraz szerokie wsparcie społeczności deweloperów, co ułatwia naukę i rozwiązywanie problemów związanych z jej użyciem.

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

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

Środowisko programistyczne - Microsoft Visual Studio 2019

Visual Studio 2019 to rozbudowane środowisko programistyczne opracowane przez firmę Microsoft. Jest to popularne narzędzie, które oferuje wiele funkcji i ułatwień dla programistów w procesie tworzenia oprogramowania.

Oto kilka cech i funkcji, które wyróżniają Visual Studio 2019:

- Edytor kodu: Posiada zaawansowany edytor kodu z funkcjami takimi jak podświetlanie składni, automatyczne uzupełnianie, refaktoryzacja kodu i podpowiedzi. Zapewnia też wsparcie dla wielu języków programowania, takich jak C++, C#, Java, Python, JavaScript, HTML, CSS i wiele innych.
- Debugowanie: Visual Studio 2019 oferuje potężne narzędzia do debugowania, które pomagają programistom w znajdowaniu i naprawianiu błędów w kodzie. Pozwala na krokowe wykonywanie kodu, obserwowanie wartości zmiennych, śledzenie stosu wywołań i analizę zdarzeń.
- Intellisense: To funkcja, która wspomaga programistów poprzez automatyczne uzupełnianie kodu, podpowiedzi dotyczące metod i właściwości, oraz wykrywanie błędów składniowych. Ułatwia to pisanie kodu i zwiększa produktywność.
- Narzędzia projektowe: Visual Studio 2019 posiada wiele narzędzi projektowych, takich jak projektowanie formularzy, edytory stylów, zarządzanie zależnościami, narzędzia do tworzenia interfejsu użytkownika i wiele innych. Pozwala to na szybkie tworzenie aplikacji graficznych i webowych.
- Integracja z platformami i usługami: Środowisko Visual Studio 2019 integruje się z wieloma platformami i usługami, takimi jak Azure, Git, Docker, Team Foundation Server (TFS) i wiele innych. Umożliwia to łatwe zarządzanie projektem, kontrolę wersji, wdrażanie aplikacji i współpracę zespołową.
- Rozszerzenia i społeczność: Visual Studio 2019 obsługuje rozszerzenia, które pozwalają programistom dostosować środowisko do swoich potrzeb. Istnieje również rozbudowana społeczność, która tworzy i udostępnia różne rozszerzenia oraz dzieli się wiedzą i doświadczeniem.

Oprogramowanie do rysowania - Aseprite

Aseprite to popularne oprogramowanie do rysowania i animacji grafiki 2D, które jest szczególnie popularne wśród artystów, twórców gier wideo i animatorów. Program ten oferuje wiele funkcji i narzędzi, które ułatwiają proces tworzenia pixel artu oraz animacji.

Oto kilka cech i funkcji, które wyróżniają program Aseprite:

- Pixel Art: Aseprite jest specjalnie zaprojektowane dla artystów tworzących pixel art. Zapewnia narzędzia do precyzyjnego rysowania pikseli, co pozwala twórcom na stworzenie szczegółowych, retro-stylowych obrazów.
- Warstwy: Program umożliwia korzystanie z warstw, co jest niezwykle przydatne przy tworzeniu skomplikowanych grafik. Można tworzyć, modyfikować i organizować warstwy, co ułatwia edycję i animację elementów graficznych.
- Animacja: Aseprite posiada narzędzia do tworzenia animacji klatka po klatce. Można ustawić kluczowe klatki, definiować animowane sekwencje i dostosowywać parametry animacji, takie jak prędkość, powtarzanie i przejścia.
- Edycja kolorów: Program oferuje funkcje edycji kolorów, które pozwalają na zmianę palety kolorów i dostosowywanie ich nasycenia, jasności i kontrastu. To ułatwia eksperymentowanie z różnymi stylami grafiki.
- Narzędzia rysowania: Aseprite zapewnia różnorodne narzędzia do rysowania, takie jak pióro, pędzel, wypełnianie wiadrem, gumka do mazania i wiele innych. Można dostosowywać rozmiar pędzla, twardość krawędzi i inne parametry, aby uzyskać pożądane efekty.
- Import i eksport: Program umożliwia importowanie różnych formatów plików graficznych, takich jak PNG, GIF czy JPEG. Można również eksportować swoje prace w różnych formatach, co ułatwia udostępnianie i wykorzystywanie ich w innych projektach.
- Skrypty i rozszerzenia: Aseprite obsługuje skrypty i rozszerzenia, które pozwalają na automatyzację powtarzalnych zadań lub dodanie dodatkowych funkcji do programu. Istnieje także rozbudowana społeczność, która tworzy i udostępnia różne rozszerzenia dla Aseprite.

7. Opis aplikacji

W tym rozdziale znajduje się szczegółowy opis pikselowej gry 2D. Przedstawiono główne elementy takie jak: interfejs użytkownika, funkcjonalności gry, instrukcja obsługi gry, zasoby graficzne oraz inne istotne aspekty aplikacji.

7.1 Interfejs użytkownika oraz funkcjonalności gry

Poniżej znajduje się interfejs użytkownika gry, czyli to, co gracz będzie widział na ekranie i jak będzie na nią oddziaływał.

- Ekran gry:
  - 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 pokonać. Wartość startowa to 10.
  - „Agility” – Zwinność. Ta statystyka 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.

Poniżej przedstawiono szczegółowy opis funkcjonalności, które zostały zaimplementowane w 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. Niektóre plansze będą generowane z całkowicie losowych grafik otoczenia.
- Mechanika ruchu bohatera gracza:
  Postać gracza, może poruszać się po całej planszy – góra, dół, lewo, prawo - używając odpowiednich klawiszy. Jeśli chce, może także używać sprintu, który będzie zwiększał szybkość ruchu lub skradania się, co będzie obniżało jego prędkość.
- Statystki postaci/ gracza:
  Gracz może ulepszać swoje statystyki otwierając skrzynie, zbierając monety, lub pokonując wrogów. Niektóre ze statystyk będą zapisywane do pliku tekstowego, po tym jak bohater zostanie pokonany – zostały one podkreślone poniżej. Rozróżniamy następujące statystyki:
  - „HeroName”
  - „HeroHealthPoints”
  - „Strength”
  - „Agility”
  - „Intelligence”
  - „RelationshipWithMages”
  - „RelationshipWithSentinels”
  - „ScorePoints”
  - "CollectedChests"
  - "CollectedCoins"

- "SlayedEnemies"

❖ System kolizji i interakcje z otoczeniem:

Cześć kolizji następuje automatycznie, jednak niektóre obiekty wymagają od gracza ręcznej interakcji, w innym wypadku stanowią po prostu tło. Gracz może prowadzić interakcje z otaczającymi go elementami naciskając odpowiedni klawisz.

- Przykłady automatycznej interakcji:
  - Walka z przeciwnikami
    - Magowie
    - Strażnicy
  - Zbieranie przedmiotów
    - Jabłka
    - Monety
    - Mikstury
- Przykłady ręcznej interakcji:
  - Obiekty interaktywne
    - Skrzynie
    - Teleporty

❖ Algorytmy sztucznej inteligencji – maszyna stanów:

Początkowo przeciwnicy będą wobec gracza 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 lub skrzyń, zaczną zadawać mu podwójne obrażenia.
- Gdy gracz zbierze Jabłko, relacje z Czarodziejami i Strażnikami ulegną poprawie, ostatecznie wrogowie mogą nawet znów przestać 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 wchodzić w interakcję 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.

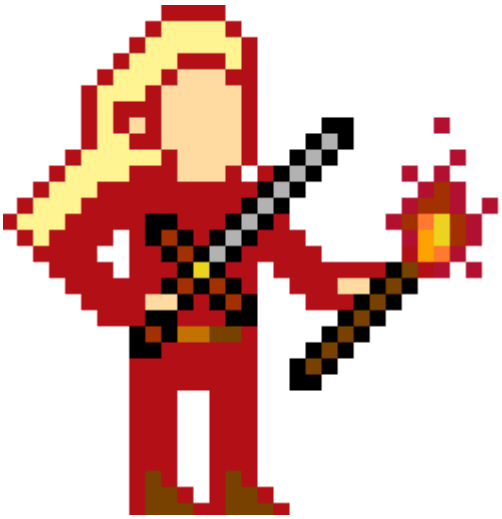
Teraz, gdy znasz instrukcję, możesz cieszyć się grą i odkrywać świat, który przed Tobą leży! Powodzenia i miłej zabawy!

7.2 Zasoby graficzne

Oto lista grafik narysowanych przez autora przy pomocy programu Aseprite.

Bohater gracza

Rysunek 7.1 Bohater



Rysunek 7.2 Bohater zraniony

Rysunek 7.3 Bohater uleczony

Źródło: Opracowanie własne

Przeciwnik Mag

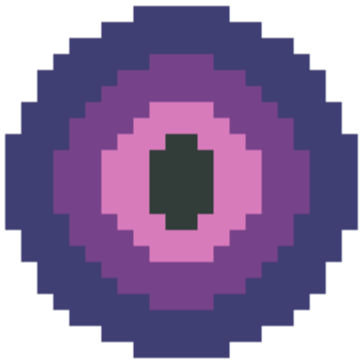
Rysunek 7.5 Przeciwnik Mag - łatwy

Rysunek 7.4 Przeciwnik Mag - średni

Rysunek 7.6 Przeciwnik Mag - trudny

Źródło: Opracowanie własne

Przeciwnik Strażnik



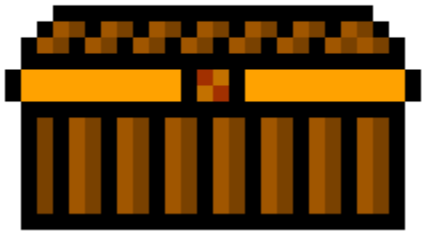
Rysunek 7.7 Przeciwnik Strażnik - trudny

Rysunek 7.8 Przeciwnik Strażnik - łatwy

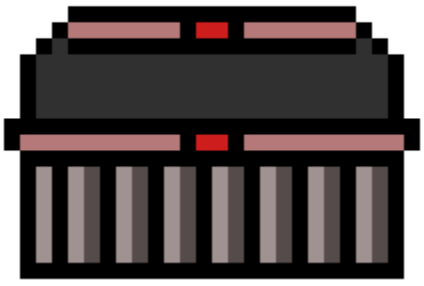
Źródło: Opracowanie własne

Interaktywne Skrzynie

Rysunek 7.9 Skrzynia - trudna



Rysunek 7.10 Skrzynia - łatwa



Rysunek 7.11 Skrzynia otwarta

Źródło: Opracowanie własne

Interaktywne Teleporty

Rysunek 7.12 Interaktywne Teleporty

Źródło: Opracowanie własne

Przedmiot Jabłko

Rysunek 7.13 Przedmiot Jabłko

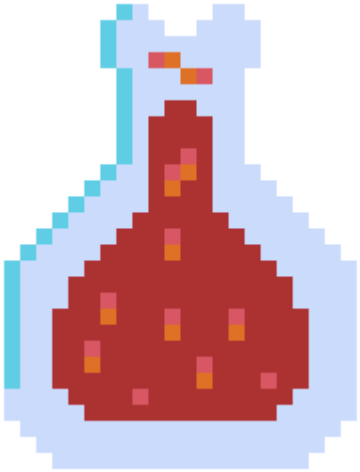
Źródło: Opracowanie własne

Przedmiot Moneta

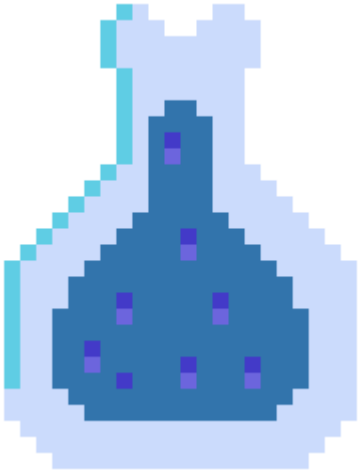
Rysunek 7.14 Przedmiot Moneta

Źródło: Opracowanie własne

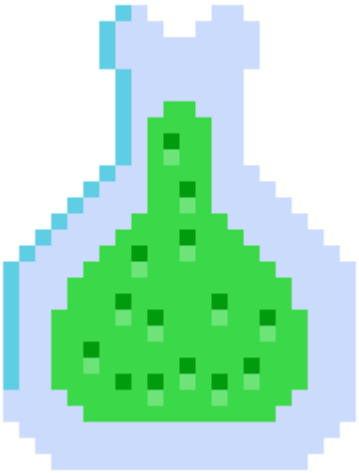
Przedmioty Mikstura



Rysunek 7.15 Mikstura siły



Rysunek 7.16 Mikstura inteligencji



Rysunek 7.17 Mikstura zwinności

Źródło: Opracowanie własne

8. Podsumowanie

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.

Realizacja celów pracy

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.

Podsumowanie osiągnięć

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 zakończenie cyklu inżynierskiego i stanowi punkt wyjścia do dalszych badań i rozwoju w dziedzinie tworzenia gier komputerowych.

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.

Spis ilustracji

Bibliografia

1. "Effective C++: 55 Specific Ways to Improve Your Programs and Designs" Scott Meyers
2. "Design Patterns: Elements of Reusable Object-Oriented Software" Gang of Four (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides)
3. "Beginning C++ Through Game Programming" Michael Dawson
4. "SDL Game Development" Shaun Mitchell
5. "Beginning Game Programming: A GameDev.net Collection" John H. Thompson
6. https://learn.microsoft.com/en-us/visualstudio/?view=vs-2022
7. https://www.libsdl.org/
8. https://en.cppreference.com/w/
9. https://cpp0x.pl/
10. https://www.ppe.pl/