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

My engineering thesis titled "Implementation of a 2D Pixel Game using the SDL2 Library" focuses on the development of a video game using the SDL2 library and incorporating basic AI algorithms in C++. The project aims to explore the practical application of software engineering principles and provide an engaging and interactive gaming experience for users.

The thesis begins with an introduction that outlines the motivation behind the project and highlights the significance of incorporating both the SDL2 library and basic AI algorithms. The SDL2 library, renowned for its versatility and cross-platform compatibility, offers a robust framework for game development. By leveraging the library's capabilities, the project seeks to create a visually appealing and functional 2D pixel game.

The implementation section describes the technical details and steps involved in developing the game. This includes setting up the necessary environment, configuring the SDL2 library, and designing the game mechanics, graphics, and user interface. The state machine model, chosen as the fundamental AI algorithm, is discussed in detail, highlighting its relevance and applicability to the game's logic and decision-making processes.

The thesis also addresses challenges encountered during the development process and the corresponding solutions employed. Issues such as optimization, performance enhancements, and bug fixing are tackled to ensure a smooth and seamless gaming experience. Furthermore, considerations for user feedback and playtesting are emphasized to refine and improve the game's overall quality.

To evaluate the implementation, the thesis employs various metrics and user feedback analysis. This assessment helps in identifying areas of improvement, providing insights into the effectiveness of the implemented state machine algorithm, and determining user satisfaction levels. The results and observations obtained from the evaluation phase contribute to the refinement and finalization of the game.

In conclusion, the thesis successfully demonstrates the implementation of a 2D pixel game using the SDL2 library and basic AI algorithms. Through the application of software engineering principles, the project achieves its objective of creating an engaging and interactive gaming experience. The thesis provides valuable insights and serves as a

foundation for future advancements in game development, particularly in integrating AI algorithms into video games.

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 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 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 przeprowadzimy analizę problemu. Skoncentrujemy się na identyfikacji głównych wymagań projektowych, funkcjonalności gry oraz potencjalnych wyzwań, z jakimi spotkaliśmy się w trakcie realizacji projektu. Analiza problemu pozwoli nam lepiej zrozumieć zakres naszego projektu i ustalić, jakie są główne aspekty, na które powinniśmy się skoncentrować podczas implementacji naszej pikselowej gry 2D.

2.1 Wymagania projektowe

W celu uzyskania jasnego zrozumienia problemu, rozpoczęliśmy od zidentyfikowania wymagań projektowych. Określiliśmy, że nasza gra będzie pikselową grą 2D, co oznacza, że będzie się opierać na estetyce retro, z prostymi pikselowymi grafikami. Chcieliśmy stworzyć przyjemną dla oka grę z intuicyjną rozgrywką, która zapewni użytkownikom zarówno rozrywkę, jak i wyzwania.

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

2.2 Potencjalne wyzwania

Podczas analizy problemu, zidentyfikowaliśmy również potencjalne wyzwania, z jakimi możemy się spotkać podczas implementacji naszej 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 będziemy musieli skonfrontować.

2.3 Analiza funkcjonalności

Ważnym etapem analizy problemu było zidentyfikowanie kluczowych funkcjonalności, które nasza gra powinna posiadać. Przeanalizowaliśmy 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

3. ANALIZA ISTNIEJĄCYCH ROZWIĄZAŃ

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

3.1 Pikselowe gry 2D

Rozpoczniemy od zbadania pikselowych gier 2D, aby zrozumieć, jakie gatunki i style są popularne w tej kategorii. Przyjrzymy się zarówno starszym, klasycznym tytułom, jak i nowszym produkcjom, które wykorzystują estetykę pikseli. Przeanalizujemy różne aspekty gier, takie jak mechanika rozgrywki, poziomy trudności, elementy wizualne, fabuła oraz reakcje graczy i opinie społeczności. W ten sposób zdobędziemy wgląd w trendy i preferencje graczy w pikselowych grach 2D.

3.1.1 Terraria

Terraria to 2D sandboxowa gra przygodowa, 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.

3.1.2 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.

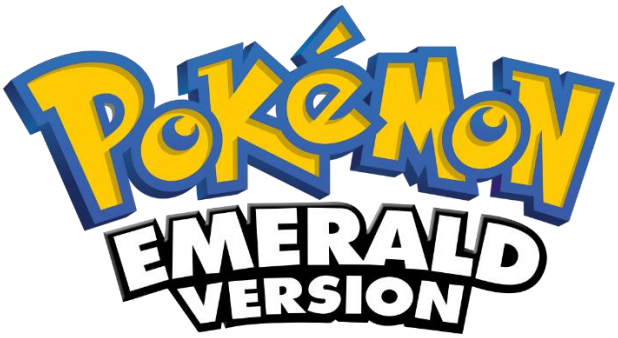
3.1.3 Factorio

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.

3.1.4 Pokémon Emerald

Pokémon Emerald to 2D przygodowa gra RPG, będąca częścią popularnej serii Pokémon. Gracz wciela się w trenera Pokémonów, który podróżuje po świecie, łapie różnorodne stworzenia zwane Pokémonami, trenuje je i walczy z innymi trenerami. Celem gry jest zbudowanie silnego zespołu Pokémonów i pokonanie Ośmiu Mistrzów, aby stać się Mistrzem Ligi Pokémonów. 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 Pokémonów.



3.2 Biblioteki programistyczne

Przeanalizujemy teraz 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 przyjrzymy się ich funkcjonalnościom, wydajności, dostępnej dokumentacji i wsparciu społeczności. Skupimy się na tych, które obsługują pikselową grafikę i są kompatybilne z językiem C++.

3.2.1 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.

3.2.2 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.

3.2.3 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.

3.2.4 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

W tym kroku porównamy zebrane informacje na temat pikselowych gier 2D i bibliotek programistycznych. Dokonamy oceny, które rozwiązania najlepiej pasują do naszych celów i wymagań projektowych. Podczas porównania będziemy brać pod uwagę 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.

4. KONCEPCJA WŁASNEGO ROZWIĄZANIA

W tym rozdziale przedstawimy koncepcję naszego własnego rozwiązania, które zaimplementujemy w naszej pikselowej grze 2D. Opiszemy główne założenia, cele i funkcjonalności, które planujemy zrealizować, aby stworzyć interesującą i satysfakcjonującą grę.

4.1 Cele projektowe

Rozpoczniemy od określenia celów, jakie chcemy osiągnąć w naszym projekcie. Mogą to być cele związane z rozgrywką, estetyką, technologią, wydajnością, czy innymi aspektami gry. Cele projektowe będą obejmować:

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

4.2 Funkcjonalności

Przejdziemy teraz do omówienia głównych funkcjonalności, które zamierzamy zaimplementować w naszej grze. Będą to być funkcje związane z rozgrywką, interfejsem użytkownika, grafiką, dźwiękiem, czy innymi aspektami. Nasze funkcjonalności powinny obejmować:

- Automatycznie i losowo generowana plansza:
  Gra powinna mieć kilka poziomów, z różnymi układami plansz, przeciwnikami.
- Mechanika 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:
  Zaimplementujemy 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.
- Ocena wyników graczy:
  Powinniśmy zapewnić możliwość oceny wyników graczy, tak aby mogli porównywać swoje osiągnięcia.

4.3 Architektura i narzędzia

W tym podrozdziale omówimy architekturę naszego rozwiązania oraz narzędzia, które wykorzystamy do implementacji gry. Warto znów wspomnieć o wykorzystaniu języka C++ wraz z biblioteką SDL2, Microsoft Visual Studio jako środowisko programistyczne oraz inne narzędzia pomocnicze, takie jak programy do projektowania grafiki ~~czy edytory dźwięku~~. Opiszemy również strukturę projektu, podział na moduły i klasy, oraz omówimy ważne decyzje projektowe, takie jak organizacja plików, zarządzanie zasobami czy obsługa zdarzeń.

Przedstawienie koncepcji własnego rozwiązania pozwoli czytelnikowi lepiej zrozumieć, jak zamierzamy zrealizować naszą pikselową grę 2D i jakie są główne elementy, które zostaną w niej uwzględnione.

Struktura projektu, podział na klasy

5. ANALIZA WYMAGAŃ SYSTEMU I PROJEKT

W tym rozdziale przeprowadzimy analizę wymagań systemowych oraz wykonamy projekt naszej 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 nam na lepsze zrozumienie wymagań projektowych i pomoże w późniejszych etapach implementacji gry.

5.1 Wymagania systemowe

Rozpoczniemy od określenia wymagań systemowych naszej gry. Będziemy musieli zdefiniować minimalne i zalecane parametry sprzętowe, na których gra powinna działać w sposób płynny i wydajny. Pomimo, iż gra nie będzie mocno skomplikowana warto spojrzeć i upewnić się w sprawie następujących wymagań systemowych:

- System operacyjny: Określimy, na jakich systemach operacyjnych nasza gra powinna być kompatybilna, na przykład Windows, macOS, Linux.
- Procesor: Określimy minimalny wymagany typ procesora oraz częstotliwość taktowania, która zapewni odpowiednią wydajność.
- Pamięć RAM: Określimy minimalną ilość pamięci RAM, która będzie wymagana do uruchomienia gry.
- Karta graficzna: Określimy minimalne wymagania dotyczące karty graficznej, aby zapewnić odpowiednią jakość grafiki i płynność animacji.
- Przestrzeń dyskowa: Określimy ilość wolnego miejsca na dysku, które będzie wymagane do zainstalowania gry oraz przechowywania zasobów.
- Inne: Jeśli nasza gra wymaga dodatkowych zewnętrznych bibliotek lub narzędzi, określimy również ich wymagania systemowe.

5.2 Projekt systemu

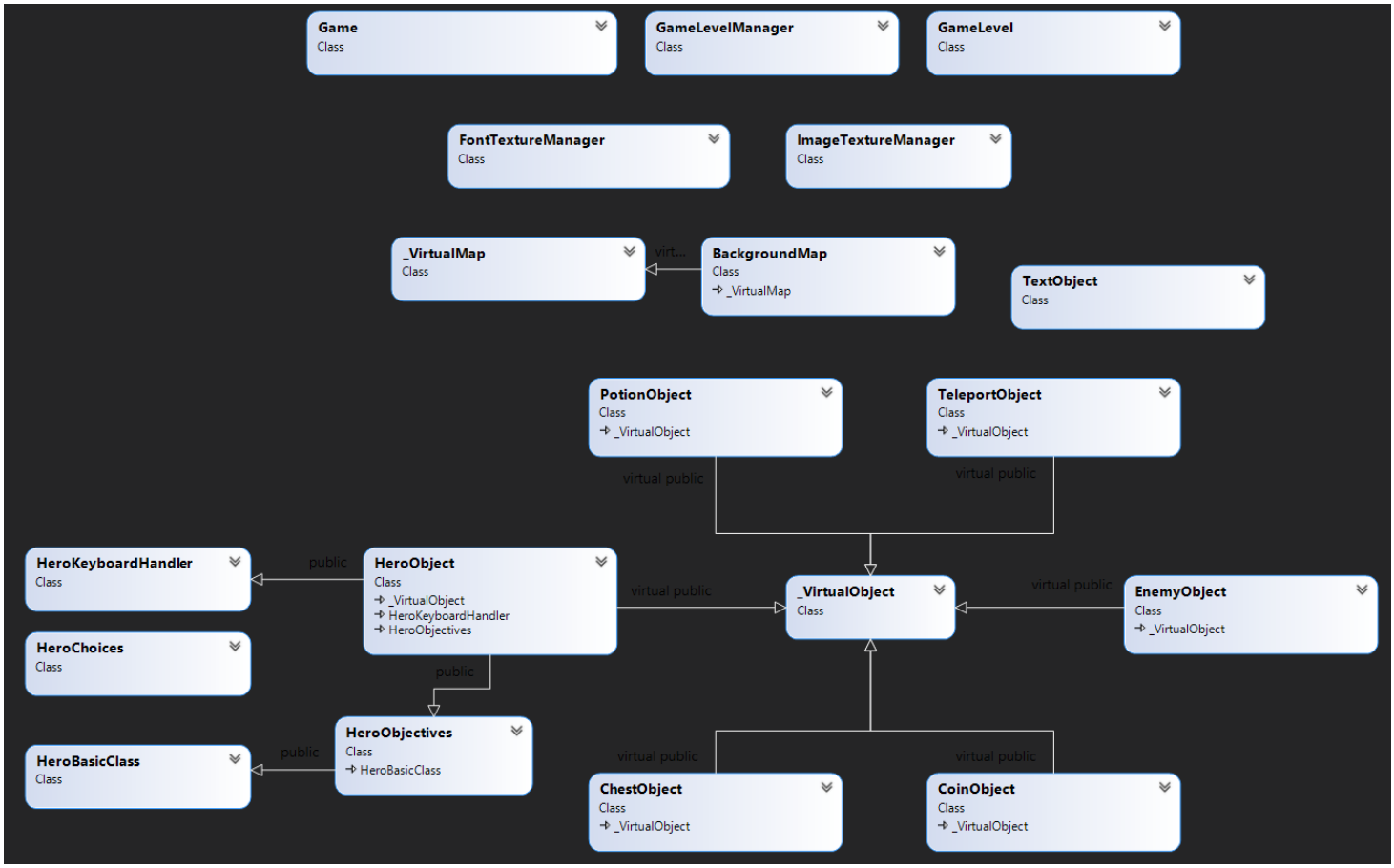
Po analizie wymagań i funkcjonalności przystępujemy teraz do wykonania projektu naszego systemu. Spiszemy opis architektury, przedstawiający główne komponenty, ich relacje i zadania. Możemy wykorzystać diagramy UML, takie jak diagram klas, diagram sekwencji czy diagram stanów, aby lepiej zobrazować strukturę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

Diagram klas

Wygenerowany przez rozszerzenie do Visual Studio 2019.



MainGame

LevelUtils

SDL2_Managers

MapUtils

GameObjects

HeroUtils

6. 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.

6.1 C++ 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.

6.2 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.

6.3 SDL2 (Simple DirectMedia Layer)

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.

6.4 Rozszerzenia SDL2 Image i SDL2 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.

7. 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

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.
- ~~Stosunki względem Czarodziei. Statystyka ta określa jak pozytywnie bądź negatywnie są nastawieni wobec niego Czarodzieje.~~
- ~~Stosunki względem Strażników. Statystyka ta określa jak pozytywnie bądź negatywnie są nastawieni wobec niego Strażnicy.~~

7.2 Funkcjonalności gry

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.

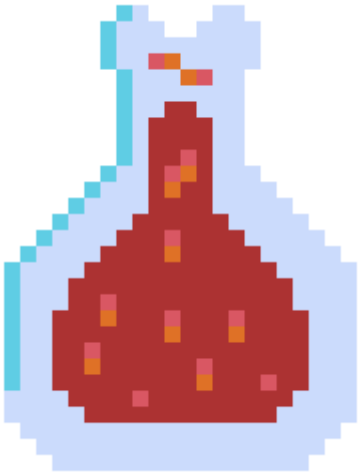
Teraz, gdy znasz instrukcję, możesz cieszyć się grą i odkrywać świat, który przed Tobą leży! Powodzenia i miłej zabawy!

7.3 Zasoby graficzne

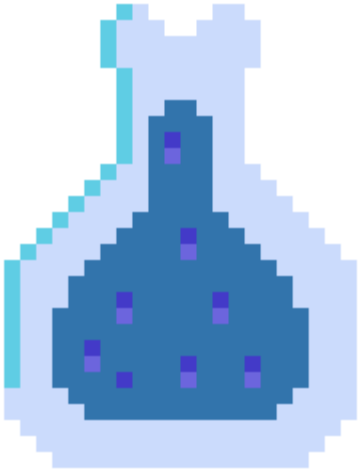
Poniżej znajduje się lista rysowanych przeze mnie grafik.

Rysunek 1 Bohater gracza

Rysunek 2 Moneta

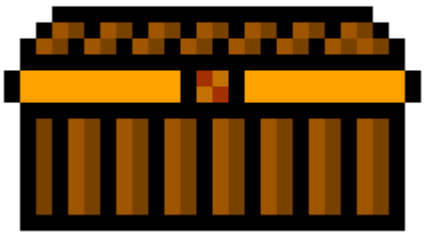


Rysunek 3 Mikstura wzmacniająca nr 1



Rysunek 4 Mikstura wzmacniająca nr 2

Rysunek 5 Złote jabłko



Rysunek 6 Skrzynia nr 1 (drewniana)

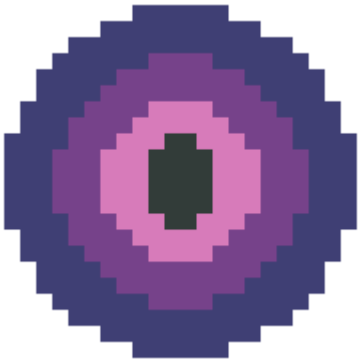
Rysunek 7 Skrzynia nr 2 (żelazna)

Rysunek 8 Skrzynia nr 3 (otworzona)

Rysunek 9 Wróg czarodziej nr 1 (łatwy)

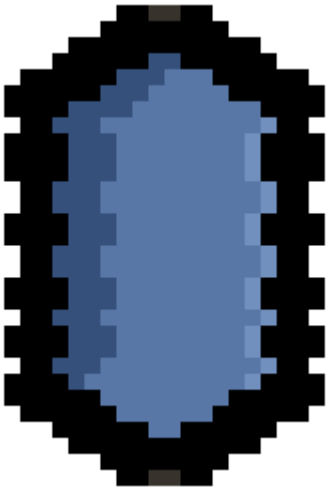
Rysunek 10 Wróg czarodziej nr 2 (średni)

Rysunek 11 Wróg czarodziej nr 3 (trudny)

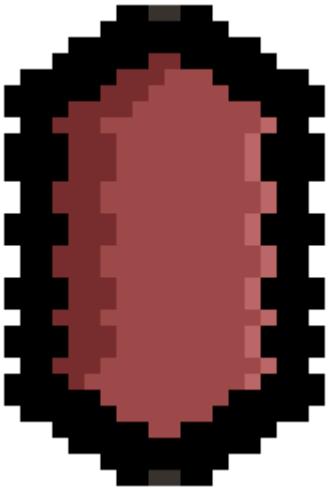


Rysunek 12 Wróg strażnik nr 1 (najsilniejszy)

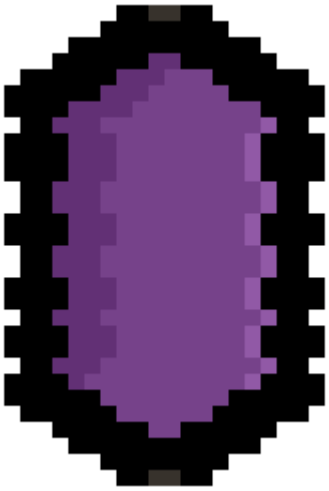
Rysunek 13 Wróg strażnik nr 2 (najsilniejszy)



Rysunek 14 Portal nr 1



Rysunek 15 Portal nr 2



Rysunek 16 Portal nr 3

8. PODSUMOWANIE

W ostatnim rozdziale naszej pracy inżynierskiej skupimy się na podsumowaniu wszystkich wcześniejszych rozdziałów oraz przedstawimy wnioski i refleksje dotyczące naszego projektu pikselowej gry 2D z wykorzystaniem biblioteki SDL2.

8.1 Realizacja celów pracy

Podsumujemy, w jaki sposób udało nam się zrealizować cele postawione przed projektem. Przypomnimy, jakie były główne cele naszej pracy, takie jak implementacja pikselowej gry, wykorzystanie biblioteki SDL2, stworzenie interfejsu użytkownika, wprowadzenie funkcjonalności gry, itp. Porównamy to, co udało nam się osiągnąć, z pierwotnymi założeniami i ocenimy, czy udało się spełnić nasze oczekiwania.

8.2 Wnioski

Przedstawimy wnioski, które wyciągnęliśmy podczas pracy nad projektem. Możemy podzielić się swoimi obserwacjami dotyczącymi implementacji gry, wykorzystanych technologii, napotkanych trudności i sposobu ich rozwiązania. Warto również wspomnieć o ewentualnych możliwościach rozwoju i ulepszenia gry w przyszłości.

8.3 Podsumowanie osiągnięć

Przedstawimy podsumowanie osiągnięć naszego projektu. Możemy wspomnieć o tym, jakie nowe umiejętności zdobyliśmy podczas realizacji pracy, jakie wyzwania przyniosło nam tworzenie gry, jakie sukcesy udało nam się osiągnąć. Ważne jest, aby podkreślić nasze indywidualne wkłady i wysiłek w procesie tworzenia gry.

8.4 Perspektywy rozwoju

Jeśli uważamy, że nasz projekt ma potencjał do dalszego rozwoju, możemy przedstawić krótko perspektywy rozwoju gry. Możemy wspomnieć o możliwościach dodania nowych poziomów, funkcji, postaci czy trybów rozgrywki. Możemy również zastanowić się nad możliwościami rozszerzenia gry na inne platformy czy dodania trybu wieloosobowego.

8.5 Podziękowania

Na zakończenie pracy możemy umieścić podziękowania dla osób, które w jakikolwiek sposób przyczyniły się do powodzenia naszego projektu. Może to być podziękowanie dla opiekuna pracy, nauczyciela, przyjaciół, czy współpracowników, którzy nam pomogli i motywowali nas w trakcie tworzenia gry.

8.6 Zakończenie

Na koniec pracy zakończymy nasze podsumowanie, podkreślając znaczenie projektu, który udało nam się zrealizować. Możemy podsumować nasze doświadczenia i satysfakcję z wykonanej pracy oraz wyrazić nadzieję, że nasza gra będzie cieszyła graczy i przyniesie wiele radości.