**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 topic is “Implementation of a 2D pixel game using the SLD2 library”.

**SPIS TREŚCI**

[1. WSTĘP 1](#_Toc136806075)

[2. ANALIZA PROBLEMU 3](#_Toc136806076)

[2.1 Wymagania projektowe 3](#_Toc136806077)

[2.2 Analiza funkcjonalności 3](#_Toc136806078)

[2.3 Potencjalne wyzwania 4](#_Toc136806079)

[3. ANALIZA ISTNIEJĄCYCH ROZWIĄZAŃ 5](#_Toc136806080)

[3.1 Pikselowe gry 2D 5](#_Toc136806081)

[3.2 Biblioteki programistyczne 5](#_Toc136806082)

[3.3 Porównanie istniejących rozwiązań 5](#_Toc136806083)

[4. KONCEPCJA WŁASNEGO ROZWIĄZANIA 6](#_Toc136806084)

[4.1 Cele projektowe 6](#_Toc136806085)

[4.2 Funkcjonalności 6](#_Toc136806086)

[4.3 Architektura i narzędzia 7](#_Toc136806087)

[5. ANALIZA WYMAGAŃ SYSTEMU I PROJEKT 8](#_Toc136806088)

[5.1 Wymagania systemowe 8](#_Toc136806089)

[5.2 Analiza funkcjonalności 8](#_Toc136806090)

[5.3 Projekt systemu 9](#_Toc136806091)

[6. OPIS WYKORZYSTANYCH TECHNOLOGII 10](#_Toc136806092)

[6.1 C++ 14 10](#_Toc136806093)

[6.2 Microsoft Visual Studio 2019 10](#_Toc136806094)

[6.3 SDL2 (Simple DirectMedia Layer) 10](#_Toc136806095)

[6.4 Rozszerzenia SDL2 Image i SDL2 TTF 11](#_Toc136806096)

[7. OPIS APLIKACJI 12](#_Toc136806097)

[7.1 Interfejs użytkownika 12](#_Toc136806098)

[7.2 Funkcjonalności gry 12](#_Toc136806099)

[7.3 Zasoby graficzne 13](#_Toc136806100)

[8. PODSUMOWANIE 14](#_Toc136806101)

[8.1 Realizacja celów pracy 14](#_Toc136806102)

[8.2 Wnioski 14](#_Toc136806103)

[8.3 Podsumowanie osiągnięć 14](#_Toc136806104)

[8.4 Perspektywy rozwoju 14](#_Toc136806105)

[8.5 Podziękowania 14](#_Toc136806106)

[8.6 Zakończenie 14](#_Toc136806107)

# 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.

# ANALIZA PROBLEMU

W tym rozdziale przeprowadzimy analizę problemu. Skoncentrujemy się na identyfikacji głównych wymagań projektowych, funkcjonalności gry oraz potencjalnych wyzwań, z jakimi mogliśmy się spotkać 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 możliwość zapisu stanu gry~~ i oceny wyników graczy.

## 2.2 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:

* Sterowanie postacią: Gracz będzie mógł poruszać swoją postacią (lewo, prawo, góra, dół) i korzystać z innych umiejętności w zależności od konkretnych wymagań gry.
* Kolizje: Zaimplementujemy system kolizji, który będzie wykrywać kolizje między postacią a innymi obiektami na planszy, takimi jak przeszkody, wrogowie lub przedmioty do zebrania.
* Interakcja: Gracz będzie mógł interakcjonować z różnymi obiektami na planszy, takimi jak portale, przełączniki, skrzynie itp., co może prowadzić do odkrywania nowych obszarów lub rozwiązywania zagadek.
* ~~Efekty dźwiękowe i muzyka: Dodamy dźwięki i muzykę, aby wzmocnić atmosferę gry i uczynić ją bardziej immersyjną dla graczy.~~
* ~~Zapis stanu gry i: Stworzymy system zapisu stanu gry, który umożliwi graczom zapisanie i wczytanie postępu w grze.~~
* Ocena wyników: zapewnimy możliwość oceny wyników graczy, tak aby mogli porównywać swoje osiągnięcia.

## 2.3 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.

# 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.

## 3.2 Biblioteki programistyczne

Następnie przeanalizujemy 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++.

## 3.3 Porównanie istniejących rozwiązań

W kolejnym 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.

# 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

Następnie przejdziemy do omówienia głównych funkcjonalności, które zamierzamy zaimplementować w naszej grze. Mogą to być funkcje związane z rozgrywką, interfejsem użytkownika, grafiką, dźwiękiem, zapisem stanu gry, czy innymi aspektami. Przykładowe funkcjonalności mogą obejmować:

* Ruch postaci: Gracz będzie mógł poruszać postacią w różnych kierunkach, skakać, atakować lub używać specjalnych umiejętności w zależności od charakterystyki postaci.
* System kolizji: Zaimplementujemy precyzyjny system kolizji, który będzie wykrywać kolizje między postacią a obiektami na planszy, takimi jak przeszkody, wrogowie czy przedmioty.
* Interaktywne obiekty: Dodamy różne interaktywne obiekty, takie jak portale czy skrzynie, które będą wpływać na rozgrywkę i odkrywanie nowych obszarów.
* ~~System dźwiękowy: Wprowadzimy efekty dźwiękowe i muzykę, które tworzą odpowiednią atmosferę i wzmocnią wrażenia graczy podczas gry.~~
* ~~Zapis stanu gry: Stworzymy system zapisu stanu gry, który pozwoli graczom zapisywać i wczytywać swoje postępy w grze, aby mogli kontynuować rozgrywkę w dowolnym momencie.~~

## 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.

# 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 Analiza funkcjonalności

Następnie przejdziemy do analizy funkcjonalności naszej gry. Spiszemy listę głównych funkcji i zachowań, które nasza gra powinna posiadać. Przykładowe funkcjonalności mogą obejmować:

* Poruszanie postacią: Gracz powinien mieć możliwość poruszania postacią w różnych kierunkach, skakania, atakowania i używania umiejętności specjalnych.
* Interakcje z otoczeniem: Postać gracza powinna mieć możliwość interakcji z różnymi obiektami na planszy, takimi jak przełączniki, drzwi, skrzynie, itp.
* System walki: Gracz powinien móc walczyć z przeciwnikami, korzystając z różnych rodzajów ataków i umiejętności.
* Zbieranie przedmiotów: W grze powinny być dostępne przedmioty, które gracz może zbierać, takie jak monety, mikstury, itp.
* System poziomów: Gra powinna mieć kilka poziomów, z różnymi układami plansz, przeciwnikami i zagadkami.

## 5.3 Projekt systemu

Po analizie wymagań i funkcjonalności przystąpimy 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.

# 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.

# 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, jak będzie wyglądać ekran gry, czyli plansza, na której rozgrywać się będą akcje. Omówimy interfejs gracza, takie jak pasek życia, wskaźniki punktów, wyświetlanie zebranych przedmiotów czy umiejętności postaci.
* Tekstowy interfejs gracza: Wyświetlać będzie posiadaną ilość zdrowia, obecne statystyki oraz zebrane punkty.

## 7.2 Funkcjonalności gry

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

* Mechanika ruchu: Opiszemy, jak gracz może poruszać postacią, w jakich kierunkach, jak skakać, atakować lub korzystać z umiejętności specjalnych.
* System kolizji: Wytłumaczymy, jak działa nasz system kolizji, jakie obiekty są uwzględniane, jakie reakcje występują po kolizji, jakie są zasady i ograniczenia kolizji.
* Interakcje z otoczeniem: Przedstawimy różne obiekty, z którymi postać gracza może się interakcjonować, takie jak portale, skrzynie czy inne przedmioty do zebrania.
* System walki: …?
* Progresja postaci: …?
* Algorytmy sztucznej inteligencji: …?

## 7.3 Zasoby graficzne

# 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

## 8.2 Wnioski

## 8.3 Podsumowanie osiągnięć

## 8.4 Perspektywy rozwoju

## 8.5 Podziękowania

## 8.6 Zakończenie