	Technical Report	
	AGH University of Science and Technology	Jun 2025

Gra komputerowa - Ufo attack

Autor: Wojciech Minior
Akademia Górniczo-Hutnicza

	Technical Report	
	AGH University of Science and Technology	Jun 2025

Spis treści

1. Wstęp

1.1. Cel projektu

Głównym celem projektu jest nauczenie się prawidłowej metodologii projektowania i implementacji oprogramowania poprzez realizację praktycznego zadania w postaci stworzenia gry komputerowej. Projekt ten ma umożliwić zastosowanie poznanych technik projektowania, programowania obiektowego oraz wzorców projektowych w praktyce.

1.2. Opis gry i jej założeń

Projekt polega na stworzeniu dwuwymiarowej gry zręcznościowej typu „space shooter”, w której gracz steruje statkiem kosmicznym, walcząc z wrogimi pociskami. Gra składa się z dwóch poziomów, a celem jest przechodzenie przez kolejne etapy, unikając przeszkód i niszcząc wrogów.

1.3. Wymagania systemowe

Minimalne Wymagania Sprzętowe

- **Procesor (CPU)**:
  - Dowolny nowoczesny procesor x86.
- **Pamięć RAM**:
  - Minimum 2 GB .
  - Zalecane 4 GB.
- **Karta Graficzna (GPU)**:
  - Karta graficzna z obsługą OpenGL w wersji co najmniej 2.0.
- **Dysk twardy**:
  - Minimum 20 MB wolnego miejsca na pliki aplikacji i zasoby.
  - Zalecane SSD dla szybszego ładowania zasobów.
- **System operacyjny**:
  - Windows 10/11 (64-bitowy).
- **Rozdzielczość ekranu**:
  - Minimum 1920x1080.
  - Zalecane 1920x1080 dla lepszych wrażeń wizualnych.

2. Funkcjonalność

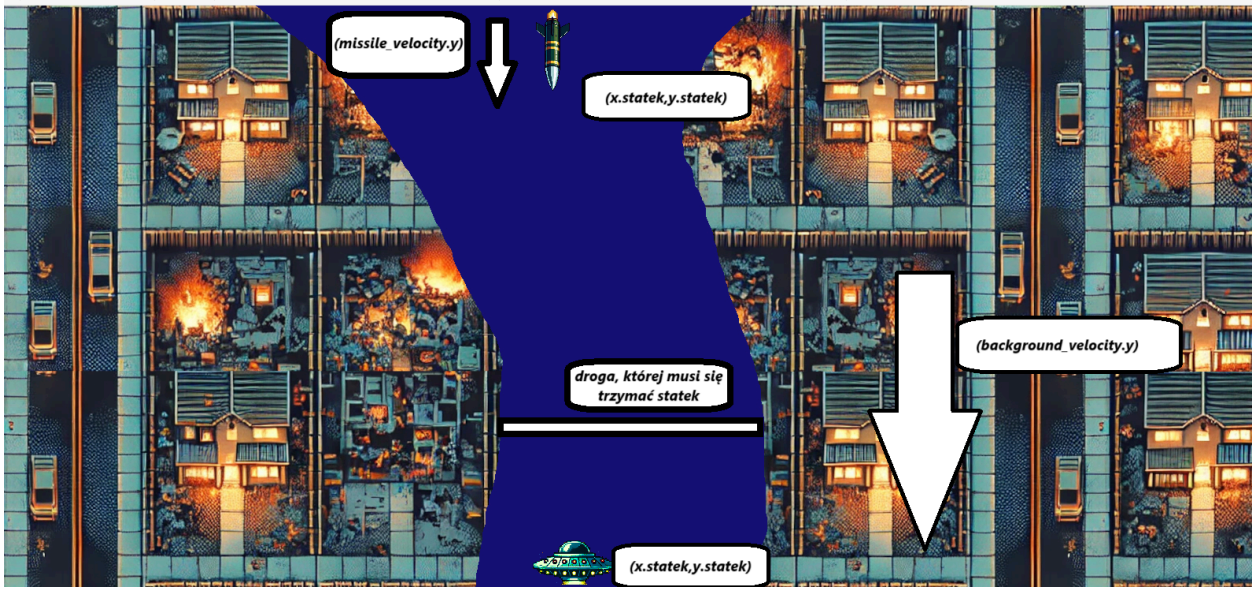
Główne funkcje gry obejmują:

- **Generowanie pocisków**: Wrogie pociski generowane są w określonych odstępach czasu i poruszają się w górę ekranu.
- **Wykrywanie kolizji**: Kolizje pomiędzy pociskami, statkiem kosmicznym i innymi elementami są wykrywane w czasie rzeczywistym.
- **Poziomy gry**: Gra posiada dwa poziomy, które różnią się tłem oraz ścieżkami, drugi poziom jest nieco trudniejszy, ponieważ niektóre drogi to pułapki.
- **Sterowanie**: Gracz może sterować statkiem za pomocą klawiszy kierunkowych oraz wystrzeliwać pociski przy użyciu klawisza górnej strzałki.
- **Ekrany zakończenia gry**: Specjalne ekrany informują o przegranej,wygranej lub przejściu do następnego poziomu.

3. Analiza Problemu

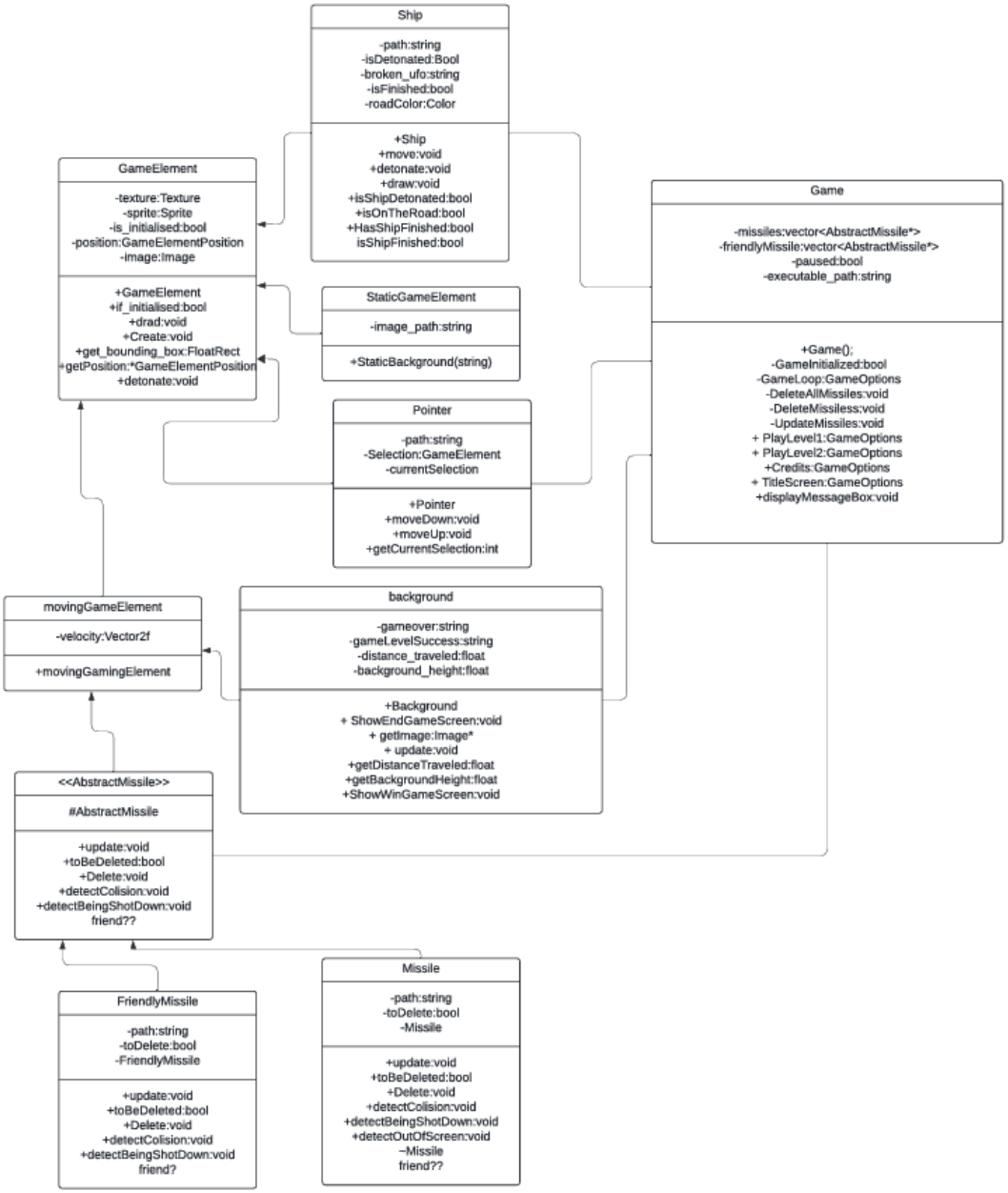
Głównym celem projektu jest stworzenie gry, w której:

- Dokładanie funkcjonalności i właściwości jest proste i intuicyjne
- Kod nie powtarza się, wspólne elementy są metodami klasy
- Brak “protez”, które przy rozbudowie gry utrudniałyby czytelność kodu
- Wszystkie nazwy jasno określają swoją funkcję
- Pociski są dynamicznie generowane w odpowiedzi na stan gry, tj pozycję statku.
- Wykrywane są kolizje między elementami gry.
- Statek kosmiczny porusza się na dynamicznym tle, reagując na interakcje z innymi elementami.
- Gra jest wizualnie atrakcyjna - w granicach możliwości



4. Projekt Techniczny

4.1 Diagram klas



4.2 Dokumentacja klas i metod

Autor podczas tworzenia aplikacji zawarł szczegółowy opis klas i metod w kodzie aplikacji. Na podstawie tych komentarzy została wygenerowana dokumentacja techniczna z wykorzystaniem programu Doxygen

UFO

Aby uruchomić dokumentację i zapoznać się z poszczególnymi elementami należy uruchomić index html, który znajduje się w katalogu dokumentacja.

5. Opis Realizacji

5.1 Środowisko programistyczne

- **Kompilator**: Visual Studio 17 2022 z obsługą C++ w standardzie C++17.
- **Biblioteka graficzna**: SFML (Simple and Fast Multimedia Library) do obsługi grafiki, okien i wejścia.
- **System operacyjny**: Windows 10/11
- **Narzędzia budowy**: CMake

5.2 Opis Implementacji

Game

- **Opis:** Najważniejsza klasa gry, zarządzająca pętlą gry, logiką poziomów oraz interakcjami między obiektami.

  Najważniejsze metody:

  - GameLoop– Główna pętla gry.
  - PlayLevel1 – Obsługuje poziom 1.
  - PlayLevel2 – Obsługuje poziom 2.

GameElement

- **Opis:** Klasa bazowa z której dziedziczą wszystkie elementy widoczne na ekranie. Klasa służąca do obsługi spirte'ów, przy jej pomocy są tworzone, ustawiane są im tekstury oraz obsługiwane są inne wspólne akcja wszystkich spritów, jeżeli jakiś element porusza się samoistnie to zamiast z GameElementu dziedziczy z MovingGameElement dzięki czemu dostaje members'a velocity.

- **Najważniejsze metody:**
  - GameElement – konstruktor służący wyłuskaniu ścieżki do pliku wykonywalnego.
  - draw – służy do rysowania sprite'ów na ekranie.
  - Create - ładujemy tekstury do sprite'a oraz kopiujemy je do metody image, która później posłuży do kopiowania koloru i porównywania go do koloru ścieżki.
  - GameInitialized - sprawdza czy statek i background poprawnie się zainicjowały

Background

- **Opis:** Klasa odpowiedzialna za elementy statyczne gry
- **Najważniejsze metody:**
  - update() – Aktualizuje pozycję tła.
  - getDistanceTraveled() – Zwraca dystans pokonany przez tło.

Ship

- **Opis:** Klasa reprezentująca statek gracza.
- **Najważniejsze metody:**
    - move – Przesuwa statek w lewo/prawo, działa na zasadzie offsetu - do aktualnej pozycji dodawane jest odpowiednia wartość w zależności od długości przytrzymania klawisza.
    - detonate() – Obsługuje zniszczenie statku.
    - isOnTheRoad() – Sprawdza, czy statek pozostaje na trasie używając koloru pod statkiem oraz koloru drogi.

AbstractMissile

- **Opis:** Abstrakcyjna klasa reprezentująca pociski (wrogie i gracza).

MissileFactory:

- **Opis:** fabryka tworząca pociski w zależności od argumentów: Missile, FriendlyMissile

Missile

- **Opis:** klasa służąca tworzeniu i obsługiwaniu pocisków lecących na gracza
- **Najważniejsze metody:**
    - Missile - tworzy pociski na na górze ekranu o tej samej współrzędnej x co statek gracza
    - update() - aktualizuje pozycję pocisku na ekranie
    - detectColision - wykrywa kolizję ze statkiem
    - detectBeingShotDown - wykrywa kolizję z pociskiem gracza

FriendlyMissile

- **Opis:** klasa służąca tworzeniu i obsługiwaniu pocisków gracza
- **Najważniejsze metody:**
    - FriendlyMissile - tworzy pociski przed graczem, lecące do góry ekranu
    - update() - aktualizuje pozycję pocisku na ekranie
    - Delete - służy do usuwania obiektu

6. Opis wykonanych testów (*testing report*) - lista buggów, uzupełnień, it

6.1 Testy jednostkowe:

- Utworzenie skryptu CMake do zbudowania pakietu testowego
- Napisanie testów jednostkowych dla klas GameElement, Missile oraz Ship.
- Wykonanie testów jednostkowych i poprawa błędów

6.2 Testy funkcjonalne:

Sprawdzenie czy tworzą się obiekty na ekranie zgodnie z założeniami.

- Sprawdzenie czy statek reaguje na naciśnięte klawisze.
- Sprawdzenie czy system wykrywa kolizję z rakietą
- Sprawdzenie czy system niszczy statek po opuszczeniu drogi
- Sprawdzenie czy wystrzelone pociski niszczą rakiety po kolizji.
- Sprawdzenie czy na ekranie może znajdować się więcej niż jeden przyjazny pocisk

Znalezione błędy i problemy

Kod usterki	Data	Autor	Opis	Stan
ERROR_01	07.01.2025	Wojciech Minior	Klasa interactions nie zwraca prawidło momentu gdy statek i pocisk się zderzają	naprawione
ERROR_02	11.01.2025	Wojciech Minior	Interactions oraz ustalanie pozycji jest nieintuicyjne	Interactions zostało usunięte a obliczanie pozycji i offsetu zostały poprawione
ERROR_03	11.01.2025	Wojciech Minior	Funkcja sprawdzająca czy statek znajduje się na wyznaczonej drodze nie działa poprawnie	Naprawione, statek może teraz poruszać się jedynie po wyznaczonej drodze
ERROR_04	19.01.2025	Wojciech Minior	Problemy ze sterowaniem, przycisk reaguje wielokrotnie na pojedyncze wciśnięcie	Naprawione, Określone przyciski wyzwalane są przy puszczaniu klawisza a nie naciskaniu

7. Podręcznik użytkownika

7.1 Sterowanie

Gra obsługuje sterowanie za pomocą klawiatury. Poniżej znajduje się lista dostępnych funkcji i ich przypisanie do klawiszy:

Ruch Statku

- Strzałka w lewo: Przesuń statek w lewo.
- Strzałka w prawo: Przesuń statek w prawo.

Strzelanie

- Strzałka w górę : Wystrzel pocisk gracza.
    - Pocisk porusza się w górę ekranu i niszczy wrogie pociski.
    - Na ekranie może być maksymalnie jeden pocisk

Pauza i Menu

- P: Włącz/wyłącz pauzę gry.
- Escape: Powróć do menu głównego lub zakończ obecny poziom.
- Spacja: Przechodzenie na nowy poziom po zakończeniu

Rozgrywka

Menu Główne

Po uruchomieniu gry zobaczysz menu główne z trzema opcjami:

1. Start Game (Rozpocznij grę) – rozpoczyna pierwszy poziom.
2. Credits (uwagi autora) – lista z paroma słowami od autora.
3. Exit (Wyjdź) – zamyka grę.

Nawigacja po menu odbywa się za pomocą strzałek w górę i w dół, a wybór opcji potwierdza klawisz Enter.

Cel Gry

Celem gry jest ukończenie wszystkich poziomów poprzez:

1. Unikanie kolizji z wrogimi pociskami.
2. Niszczenie wrogich pocisków przy pomocy pocisków gracza.
3. Utrzymanie statku na trasie i dotarcie do końca poziomu.

Jeśli statek zderzył się z wrogim pociskiem lub opuści trasę, gracz przegrywa poziom i zostaje wyświetlony ekran przegranej.

Poziomy Gry

Gra składa się z dwóch poziomów:

1. Poziom 1: Tło przedstawia trasę na pustyni. Uliczki są wąskie ale niezbyt trudne.
2. Poziom 2: Tło staje się bardziej skomplikowane pojawiają się ślepe zaułki oznaczone znakami. Wrogie pociski pojawiają się szybciej, a ich prędkość wzrasta.

Po ukończeniu poziomu 2 wyświetlany jest ekran zwycięstwa.

Przegrana

Jeśli statek gracza:

- Zostanie trafiony wrogim pociskiem.
- Opuszcza trasę (wykryte jako kolor różny od dopuszczalnego).

Gra automatycznie wyświetli ekran przegranej. Gracz może wtedy powrócić do menu głównego.

Wygrana

Gracz wygrywa poziom, gdy statek dotrze do końca trasy. Ekran wygranej wyświetli się automatycznie, a gracz zostanie przeniesiony na kolejny poziom lub do ekranu zwycięstwa.

8. Metodologia Rozwoju i Utrzymania Systemu

Struktura gry pozwala na łatwe wprowadzanie zmian:

- Dodawanie nowych poziomów i przeciwników.
- Regulacja poziomu trudności, np. prędkości pocisków.
- Aktualizowanie grafik i dźwięków.
- Prostą obsługę błędów, które są wyświetlane

9. Bibliografia

1. Cyganek B.: Programowanie w języku C++. Wprowadzenie dla inżynierów. PWN, 2023.
2. Tim Buchalka's Learn Programming Academy,Dr. Frank Mitropoulos.:Beginning C++ Programming - From Beginner to Beyond

	Technical Report	
	AGH University of Science and Technology	Jun 2025