REV	DATA	ZMIANY
0.1	10.01.2025	*Michał Górka (migorka@student.agh.edu.pl)*
0.2	26.01.2025	*Michał Górka (migorka@student.agh.edu.pl)*

„COSMO MINER" – GRA INSPIROWANA „ASTEROIDS"

Autor: Michał Górka
Akademia Górniczo-Hutnicza

Spis treści

Lista oznaczeń

HP	Punkty życia
CD	Czas odnowienia

Wstęp

Dokument dotyczy opracowania gry komputerowej w języku C++ inspirowanej grą „Asteroids” z 1979 roku. Gra polega na unikaniu asteroid i niszczeniu ich za pomocą lasera w celu zdobycia jak największej ilości punktów.

Projekt początkowo miał bazować na grze „Vampire Survivors” ale z powodu ograniczonego czasu początkowy koncept został zmieniony.

Głównym założeniem projektu było poznanie podstaw projektowania i programowania gier oraz poznanie biblioteki Raylib.

Funkcjonalność (*functionality*)

Funkcje gry:

- Obracanie statku
- Nadawanie statkowi przyśpieszenia w kierunku, w który jest skierowany
- Strzelanie laserami w kierunku, w który skierowany jest statek
- Generowanie asteroid w losowych miejscach na granicach ekranu z losowymi parametrami (takimi jak kierunek ruchu czy rozmiar)
- Zdobywanie punktów za niszczenie asteroid zależne od ich wielkości
- Przechodzenie obiektów które dotknęły krawędzi ekranu na przeciwną stronę
- Obsługa dźwięków i muzyki

Specyfikacja techniczna:

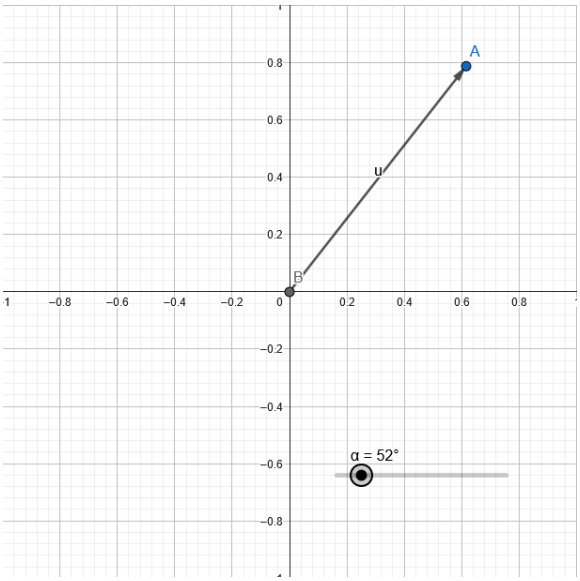
- Rozdzielczość: 1920x1080
- FPS: 60

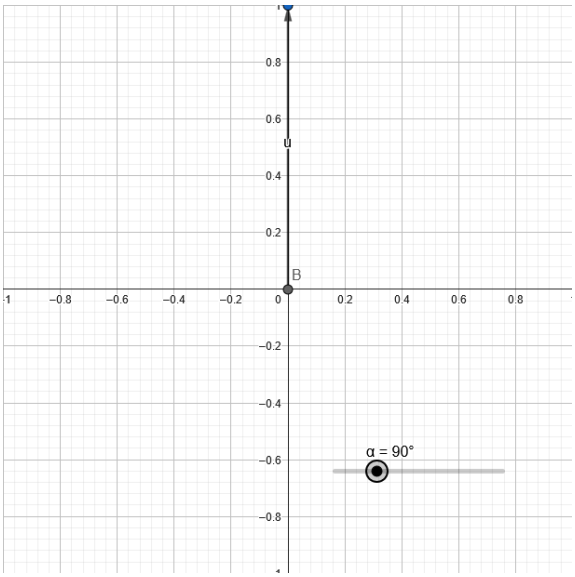
Analiza problemu (*problem analysis*)

Obliczanie wektora przyśpieszenia:

Wektor przyśpieszenia jest obliczany za pomocą dwóch składowych:

$x = \cos(\alpha) \quad y = \sin(\alpha)$





Wektor ten jest wektorem jednostkowym co możemy łatwo obliczyć ze wzoru na długość wektora i wzoru na jedynkę trygonometryczną:

$$l = \sqrt{\cos^2(\alpha) + \sin^2(\alpha)} = \sqrt{1} = 1$$

W kodzie wektor ten jest przemnażany prze konkretną wartość aby aby nadać obiektowi odpowiednią prędkość.

Podstawy działania biblioteki raylib:

Biblioteka Raylib jest biblioteką służącą do programowania prostych gier komputerowych. Daje ona użytkownikowi dostęp do wielu funkcji takich jak:

- Sprawdzanie kolizji
- Obsługa inputów (np. klawiatura, myszka)
- Rysowanie prostych kształtów i ładowanie własnych tesktur
- Rysowanie menu za pomocą tekstu
- Obsługa grafiki 3D
- Odtwarzanie muzyki i dźwięków

Wybrana została do tego projektu głównie ze względu na to w jak prosty i przejrzysty sposób umożliwia zaimplementowanie wszystkich potrzebnych funkcji do prostej gry jaką jest „Cosmo Miner”.

Projekt techniczny (*technical design*)

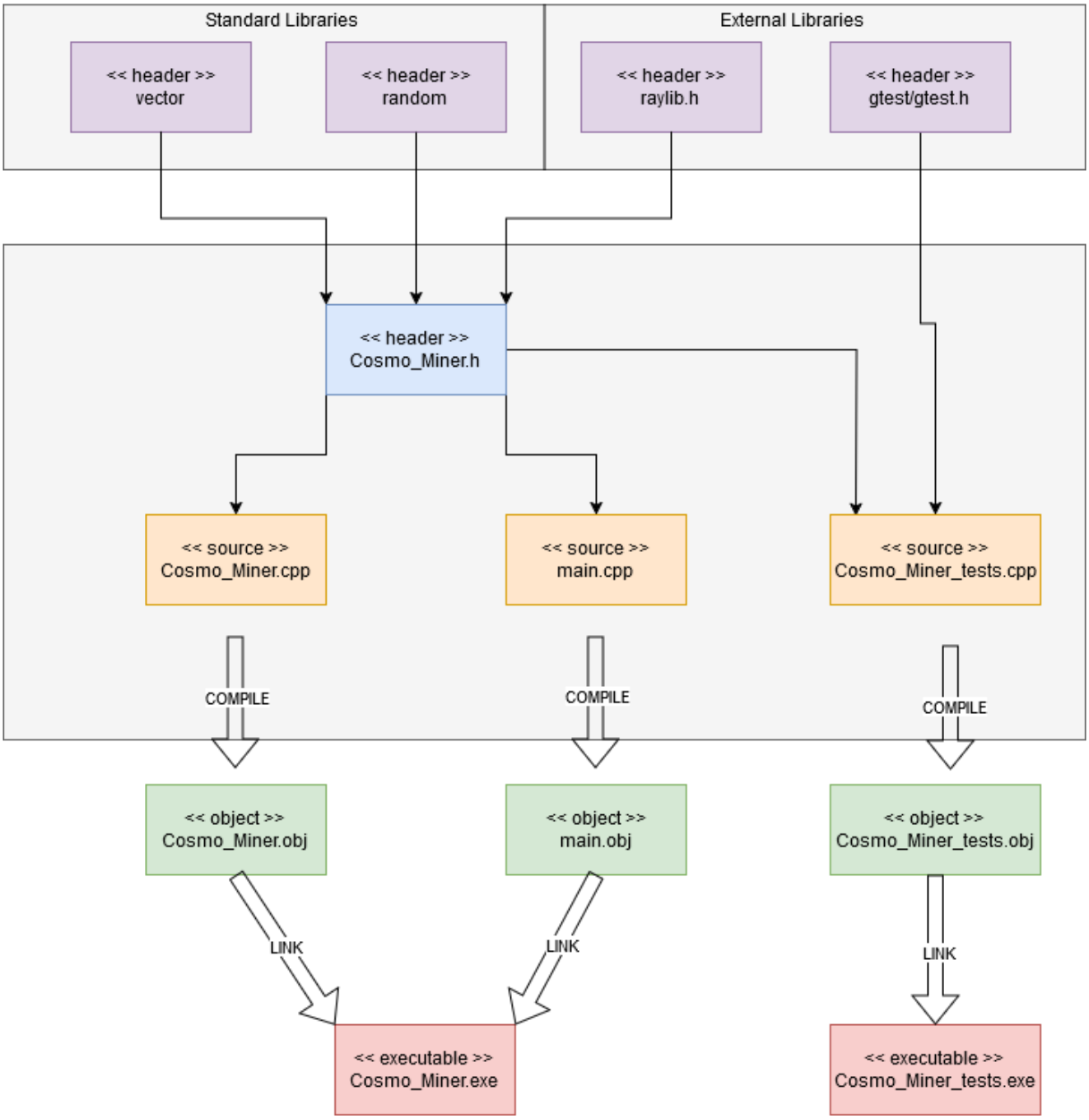


Diagram 1. Diagram UML przedstawiający relacje między plikami w projekcie. Fioletowe – biblioteki standardowe i zewnętrzne, niebieskie – pliki nagłówkowe, pomarańczowe – pliki źródłowe samej gry i pliku testowego, zielone – obiekty, czerwone – pliki wykonywalne obecne w folderze build.

Di

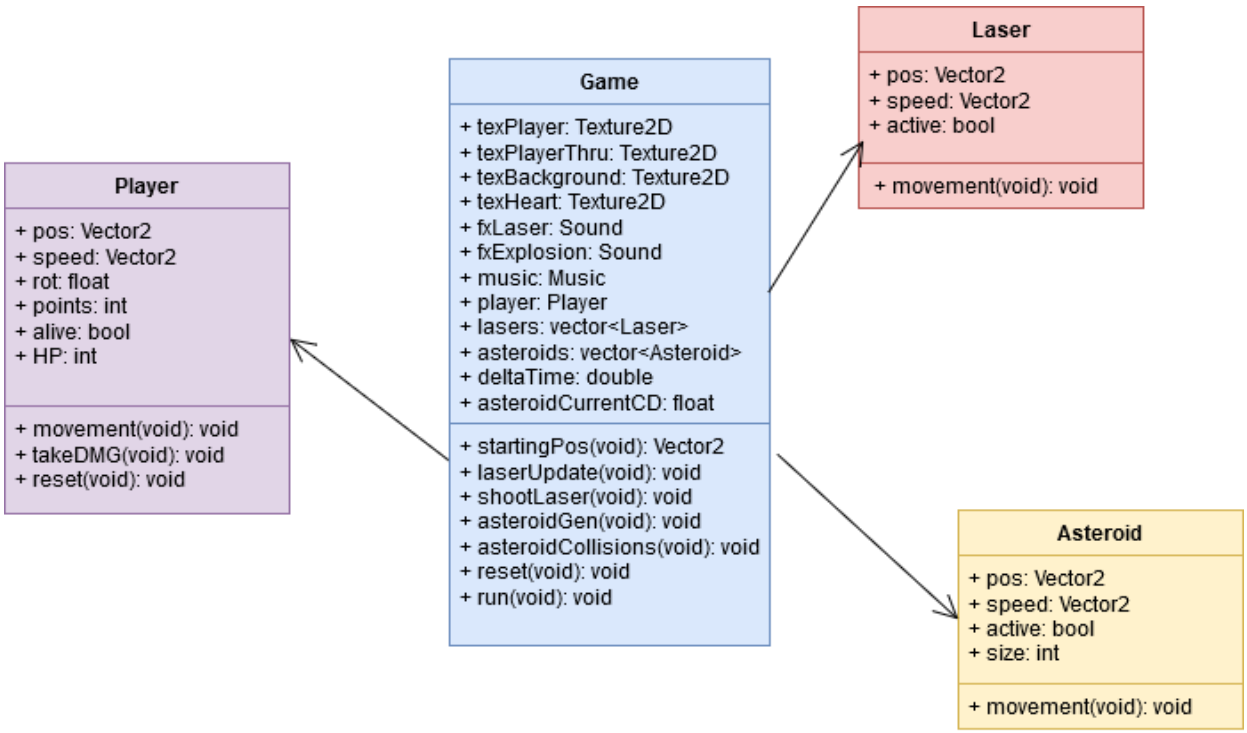


Diagram 2. Diagram UML przedstawiający zależności między klasami. Główna klasa Game pobiera informacje ze wszystkich pozostałych klas

Opis realizacji (*implementation report*)

Oprogramowanie:

- Język programowania: C++
- Biblioteka: Raylib + biblioteki standardowe
- Biblioteka do testów: GTest
- IDE: Visual Studio 2022
- System kontroli wersji: Git

Środowisko testowe:

- System: Windows 10 (64 bit)
- 16 GB RAM

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

Kod usterki	**Data**	**Autor**	**Opis**	**Stan**
ERR#1	15.01.2024	Michał Górka	Asteroida czasem nie znika ale po kontakcie i tak pojawiają się mniejsze. Dzieje się to głównie na początku działania kodu.	Nienaprawione
ERR#2	22.01.2024	Michał Górka	Błędy przy kompilowaniu programu testowego z narzędziem GTest.	Naprawione poprzez wymuszenie na CMake korzystania z nowszej wersji C++

Testy wybranych funkcjonalności projektu zostały wykonane za pomocą biblioteki GTest i są dostępne w pliku Cosmo_Miner_tests.cpp:

```
[==========] Running 5 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 5 tests from CosmoMinerTests
[ RUN      ] CosmoMinerTests.PlayerSimpleMovement
[       OK ] CosmoMinerTests.PlayerSimpleMovement (0 ms)
[ RUN      ] CosmoMinerTests.PlayerMovementBesidesBorder
[       OK ] CosmoMinerTests.PlayerMovementBesidesBorder (0 ms)
[ RUN      ] CosmoMinerTests.PlayerDMGTest
[       OK ] CosmoMinerTests.PlayerDMGTest (0 ms)
[ RUN      ] CosmoMinerTests.PlayerDeathTest
[       OK ] CosmoMinerTests.PlayerDeathTest (0 ms)
[ RUN      ] CosmoMinerTests.AsteroidGen
[       OK ] CosmoMinerTests.AsteroidGen (5 ms)
[----------] 5 tests from CosmoMinerTests (7 ms total)

[----------] Global test environment tear-down
[==========] 5 tests from 1 test suite ran. (8 ms total)
[  PASSED  ] 5 tests.
```

Wszystkie testy przebiegły pomyślnie.

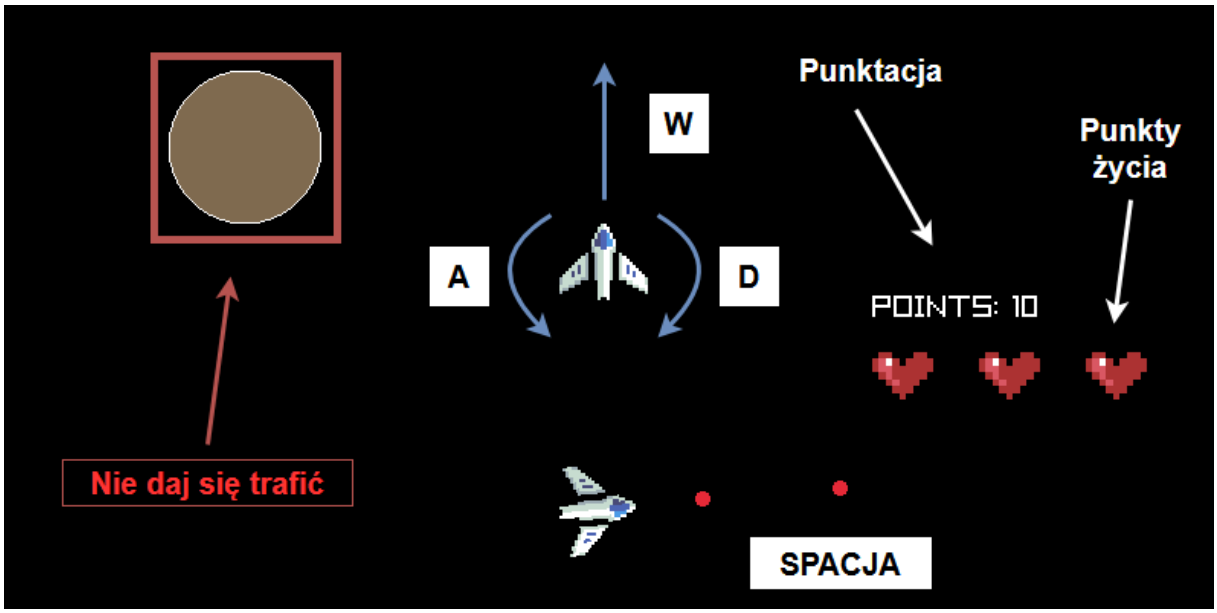
Podręcznik użytkownika (*user's manual*)

W celu kompilacji programu należy sklonować repozytorium i skompilować za pomocą narzędzia CMake.

Celem gry jest unikanie nadlatujących asteroid i niszczenie ich za pomocą lasera w celu zdobycia jak największej liczby punktów. Duże i średnie asteroidy rozbijają się na mniejsze, a ich nowe fale generowane są co losową ilość czasu. Zostanie trafionym przez asteroidę trzy razy skutkuje porażką z możliwością ponownego rozpoczęcia gry.

Punktacja:

- Mała asteroida – 10 pkt.
- Średnia asteroida – 20 pkt.
- Duża asteroida – 30 pkt.



W: nadanie przyśpieszenia w kierunku, w który zwrócony jest statek

A/D: obrót statku w lewo/prawo

SPACJA: wystrzelenie lasera w kierunku, w który zwrócony jest statek

ENTER: restart gry po przegranej

Metodologia rozwoju i utrzymania systemu (*system maintenance and deployment*)

Projekt był rozwijany w metodologii kaskadowej – po zakończeniu jednej części projektu rozpoczynana była dopiero praca nad kolejną. Wiedza potrzebna do obsługi biblioteki Raylib została czerpana z jej dokumentacji jak i kodów przykładowych umieszczonych na stronie Rayliba (zob. bibliografia).

Możliwości rozwoju na przyszłość:

- Poprawa optymalizacji kodu
- Dodanie większej ilości tekstur i dźwięków
- Tablica wyników
- Szansa na „wypadnięcie" przedmiotów z asteroid czasowo ułatwiających rozgrywkę (podwójny laser, tarcza chroniąca jednorazowo przed utratą życia)
- Lokalna rozgrywka wieloosobowa

W wolnym czasie planuje rozbudowywać projekt o dalsze funkcjonalności albo wykorzystać go jako bazę stworzenia innego projektu z wykorzystaniem biblioteki Raylib.

Bibliografia

[1] Cyganek B.: Programowanie w języku C++. Wprowadzenie dla inżynierów. PWN, 2023.
[2] Dokumentacja Raylib: https://www.raylib.com/cheatsheet/cheatsheet.html
[3] Przykłady użycia funkcji w Raylib: https://www.raylib.com/examples.html

Assety

[1] Tekstura statku: https://foozlecc.itch.io/void-main-ship
[2] Muzyka: https://starshipinfinity.itch.io/retrofuture-soundtrack-pack-vol-1
[3] Dźwięki: https://shapeforms.itch.io/shapeforms-audio-free-sfx