GRA KOMPUTEROWA „DIVER”

Autor: Mateusz Kosek

Prowadzący zajęcia: Dr hab. inż. Bogusław Cyganek

Akademia Górniczo-Hutnicza

Spis treści

1. Wstęp

Gra "Diver" jest dwuwymiarową grą zręcznościową, w której gracz wciela się w nurka eksplorującego głębiny oceanu. Celem gry jest zbieranie butli z tlenem, unikanie rekinów i zarządzanie zasobami tlenu, aby osiągnąć jak najwyższy wynik. Projekt powstał w celu zaprezentowania umiejętności programistycznych oraz wykorzystania technologii takich jak raylib, C++, i GTest.

Głównym celem projektu było stworzenie gry zręcznościowej opartej na prostych mechanikach, które będą skalowalne oraz możliwe do przetestowania pod kątem funkcjonalności i poprawności działania. Ważnym elementem było także wdrożenie elementów animacji, interakcji gracza z obiektami w świecie gry oraz obsługi wyjątków w grze.

2. Funkcjonalności

a) Sterowanie nurkiem
b) Tworzenie rekinów płynących w stronę nurka
c) Tworzenie Butli z tlenem dla nurka, które może zebrać, aby uzupełnić zapas tlenu
d) Zarządzanie paskiem zdrowia (sercami) oraz paskiem tlenu
e) Przyśpieszanie animacji, muzyki, oraz częstotliwości pojawiania się rekinów oraz butli z tlenem, w celu zwiększenia trudności z czasem rozgrywki.
f) Przechodzenie między ekranami, Tytułowym, Rozgrywki oraz Końcowym.
g) Ponowne podejścia bez konieczności wyłączenia gry.
h) Chwilowa „niewrażliwość” nurka, potrzebna aby uciec przed atakującym rekinem
i) Wiadomości, „Dive!”, „Beware of Sharks!”, „Breathe!”, w zależności od sytuacji

3. Mechaniki Gry

a) Mechanika Tlenu

Nurek musi zbierać butle z tlenem pojawiające się na mapie, zanim znikną, aby uniknąć wyczerpania zasobów tlenu. Każda butla dodaje określoną ilość tlenu do paska tlenu nurka.

b) Mechanika kolizji i życia

Gra implementuje detekcję kolizji między nurkiem a rekinami. Kolizja z rekinem skutkuje utratą zdrowia oraz aktywowaniem trybu "nietykalności" na kilka sekund.

c) Skalowanie Trudności

Z biegiem czasu gra przyspiesza, zwiększając prędkość poruszania się rekinów oraz częstotliwość ich pojawiania się.

4. Projekt techniczny

4.1 Struktura Projektu - Hierarchia Klas

a) **Shark** - Klasa odpowiedzialna za rekiny. Zarządza ich ruchem, animacją oraz kolizjami z nurkiem.
b) **OxygenBottle** - Reprezentuje butle z tlenem, które gracz może zbierać.

4.2 Struktura blokowa projektu

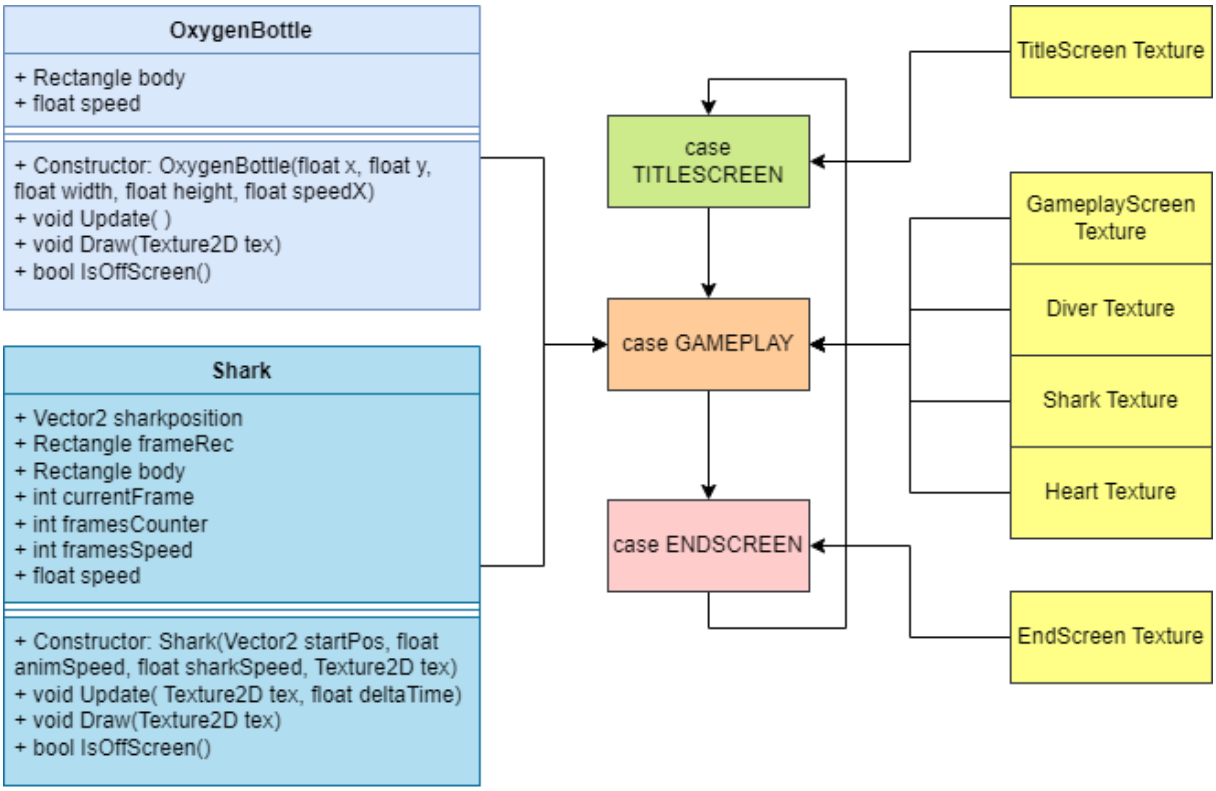


Figura 1

Obie klasy OxygenBottle oraz Shark są używane tylko w case GAMEPLAY. Tekstury używane zależnie od case. Rysowanie wszystkich tekstur oraz kształtów odbywa się w funkcji Draw() jak w kodzie poniżej:

```
BeginDrawing(); //rozpoczęcie rysowania
ClearBackground(RAYWHITE);
switch (currentScreen) {
   case TITLE:{    } break;
   case GAMEPLAY:{    } break;
   case ENDING:{    } break;
   default: break; }
EndDrawing();
```

4.3 Diagram UML

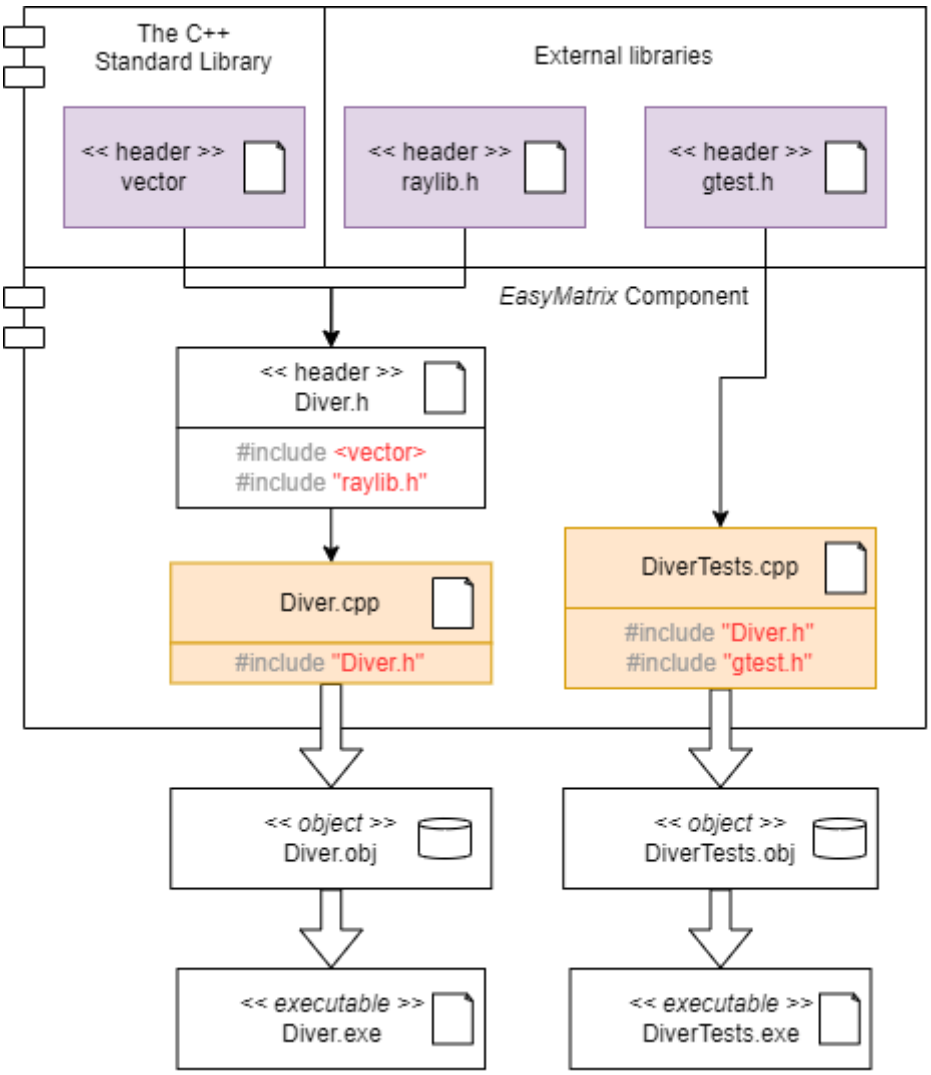


Figura 2

Diagram UML (Figura 2) pokazuję strukturę plików używanych w projekcie „Diver”. Jest to wizualizacja organizacji projektu, dla ułatwienia pokazania w schemacie blokowym.

5. Implementacja Techniczna

5.1 Język i Biblioteki

Projekt został zaimplementowany w języku C++ z wykorzystaniem biblioteki raylib do renderowania grafiki 2D oraz Google Test (GTest) do testowania jednostkowego.

5.2 Komputer do przeprowadzenia testów

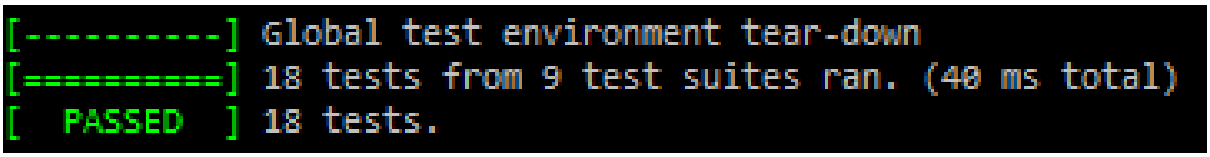
a) Procesor: Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz 3.00 GHz
b) Pamięć RAM: 32,0 GB
c) Typ systemu: 64-bitowy system operacyjny Windows 10 Pro, procesor x64
d) Karta graficzna: NIVIDIA GeForce GTX 1060 3GB
e) Rozdzielczość ekranu: 1920x1080

5.3 GTest

Do projektu zaimplementowano testy jednostkowe, które sprawdzają poprawność kluczowych funkcji w grze. Przykłady testowanych funkcji:

a) Sprawdzanie poprawności detekcji kolizji (metoda CheckCollisionCircleRec).
b) Testy poprawności aktualizacji pozycji obiektów (np. Shark::Update()).

Testy	Wynik
Test kolizji prostokątów	OK
Testy dla obliczania odległości między punktami	OK
Testy dla obliczania odległości między punktami	OK
Testy dla zarządzania rekinami	OK
Testy dla tlenu	OK
Testy dla nurka	OK
Testy dla zarządzania wektorami rekinów	OK
Testy dla mechaniki gry	OK
Testy końcowe (Sprawdzanie czy rozgrywka powinna się skończyć)	OK
Testy dla zarządzenia czasem gry	OK



6. Opis wykonanych testów - lista buggów i uzupełnień

Kod usterki	Data	Autor	Opis	Stan
Zgłoszono wyjątek w lokalizacji 0x0000000000000000 w Diver.exe: 0xC0000005: Naruszenie zasad dostępu podczas wykonywania w lokalizacji 0x0000000000000000.	24.01.2025	Mateusz Kosek	Program próbuję odwołać się do miejsca w pamięci które jest puste	Naprawione: Najpierw trzeba zainicjalizować okno InitWindow, dopiero potem ładować tekstury
Nie można otworzyć pliku dołącz: 'raylib.h': No such file or directory	25.01.2025	Mateusz Kosek	Po próbie dodania GTest, biblioteka raylib przestaje działać	Naprawione: ChatGPT, pomógł wygenerować poprawną wersję CMake z implementacją GTest i używania GTest

7. Podręcznik użytkownika

Po włączeniu gry, wita nas ekran startowy. Po wciśnięciu klawisza ENTER, zaczyna się rozgrywka. Naszym celem jest przetrwanie jak najdłużej, unikanie rekinów i dbanie aby nasz zapas tlenu się nie skończył. Musimy unikać rekinów, których z czasem jest coraz więcej i są coraz szybsze. W przypadku wyczerpania się tlenu, stracenia wszystkich punktów życia lub chęci wcześniejszego skończenia rozgrywki (Należy użyć klawisza ENTER), zastaniemy ekran końcowy, który informuje nas o naszym wyniku i końcu rozgrywki. Po przegranej możemy znów wcisnąć klawisz ENTER, aby wrócić do ekranu startowego, aby zagrać jeszcze raz.

Ekran startowy

Ekran Końcowy

8. Metodologia rozwoju i utrzymania systemu

W celu utrzymania i rozwoju gry "Diver" można rozszerzyć o niżej podane możliwości:

a) **Rozszerzenie mechanik gry**:
- Dodanie mechaniki zapobiegającej wypływaniu nurka poza granice mapy.
- Wprowadzenie nowych przeciwników, takich jak meduzy lub ośmiornice, z unikalnym zachowaniem (np. strzelaniem do nurka).
- Implementacja mechaniki zbierania serc, które pozwalają na odzyskanie zdrowia nurka.

b) **Dodanie nowych poziomów**:
- Zaprojektowanie kilku poziomów o różnym stopniu trudności.
- Dodanie nowych grafik dla innych poziomów

c) **Nowe sposoby zdobywania punktów**:
- Zbieranie skarbów, które pojawiają się losowo na mapie.
- Możliwość pokonywania przeciwników np. harpunem za co można otrzymać punkty

d) **Integracja dodatkowych funkcji**:
- Dodanie wsparcia dla trybu wieloosobowego, w którym gracze mogą rywalizować o najlepszy wynik, co za tym idzie, opcja zapisu wyniku
- Rozbudowa opcji personalizacji postaci (wybór wyglądu nurka).

e) **Rozwój testów jednostkowych i automatyzacji**:
- Rozszerzenie testów GTest o nowe funkcjonalności w grze.
- Implementacja systemu CI/CD, który automatycznie uruchamia testy przy każdej aktualizacji kodu.

Bibliografia

Kod:

[1] https://www.raylib.com/cheatsheet/cheatsheet.html

[2] https://www.raylib.com/examples.html

Darmowe tekstury:

[3] https://free-game-assets.itch.io/free-underwater-world-pixel-art-backgrounds

[4] https://havran.itch.io/pixel-heart?download

[5] https://sergeant-slash.itch.io/32x32-animated-shark-sprite

[6] https://dkproductions.itch.io/pixel-art-diver

Muzyka:

[7] https://github.com/raysan5/raylib/tree/master/examples/audio/resources

Osoby wspierające:

[8] Michał Górka – pomoc z wyborem silnika graficznego

[9] Wojciech Midura – pomoc z poprawnym konfigurowaniem GitHub'a

[10] chatGPT – stworzenie poprawnej wersji CMake implementującego GTesta

Lista oznaczeń

GTest	Testy jednostkowe GoogleTest
RayLib	Prosta i łatwa w użyciu biblioteka do programowania gier wideo.

REV	DATA	ZMIANY
0.1	11.01.2025	Mateusz Kosek (mateuszkosek@student.agh.edu.pl)
0.2	22.01.2025	Mateusz Kosek (mateuszkosek@student.agh.edu.pl)
0.3	24.01.2025	Mateusz Kosek (mateuszkosek@student.agh.edu.pl)
0.4	25.01.2025	Mateusz Kosek (mateuszkosek@student.agh.edu.pl)
0.5	26.01.2025	Mateusz Kosek (mateuszkosek@student.agh.edu.pl)