REV	DATA	ZMIANY
0.1	23.01.2018	*Michał Berdzik (michalberdzik97@gmail.com)*
0.1.1	01.02.2018	*Michał Berdzik (michalberdzik97@gmail.com)*
0.1.1	03.04.2018	*Michał Berdzik (michalberdzik97@gmail.com)*

GRA "RIVER RAID"

Autor: Michał Berdzik

Akademia Górniczo-Hutnicza

Spis Treści

Lista oznaczeń

SDL	Simple DirectMedia Layer
SFML	Simple and Fast Multimedia Library
LMP	Lewy Przycisk Myszy
VBO	Vertex Buffer Array
NPC	Non Player Character
OpenGL	Open Graphics Library

Wstęp

Dokument dotyczy opracowania gry na wzór gry z lat 80 pod tytułem “River Raid” . Gra polega na sterowaniu samolotem, którym musimy niszczyć wrogie statki oraz zdobywać beczki z paliwem aby uzupełnić paliwo.

Wymagania systemowe (*requirements*)

- Windows 7 lub nowszy
- Karta graficzna z obsługą OpenGl powyżej wersji 2.0
- Minimum 1GB pamięci RAM
- 200MB wolnej przestrzeni na dysku

Funkcjonalność (*functionality*)

Do funkcjonalności gry możemy zaliczyć:

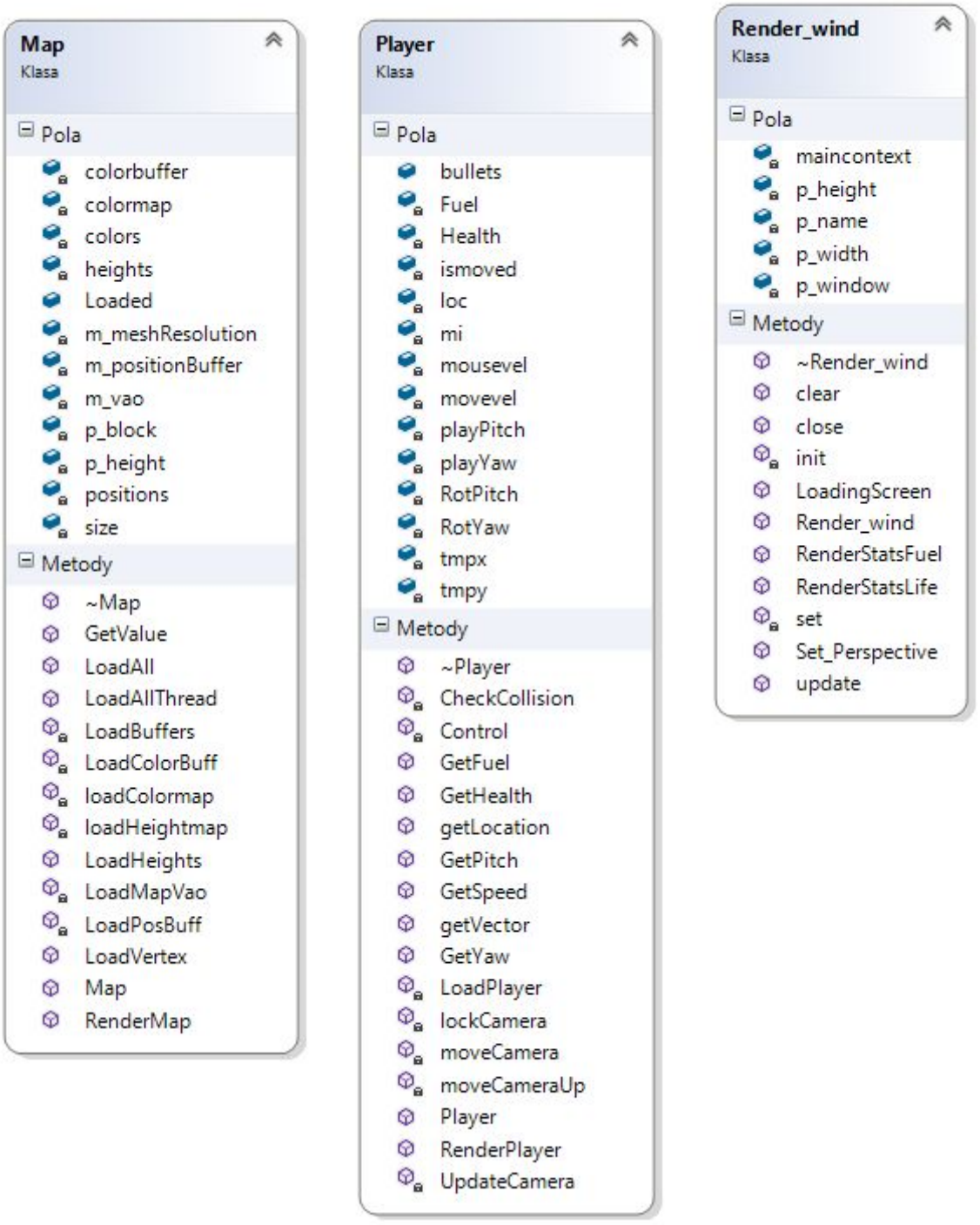
- Możliwość sterowania samolotem
- Możliwość strzelania
- Możliwość rozbicia się o elementy terenu, przez co tracimy punkty życia
- Ciągły pobór paliwa

Analiza problemu (*problem analysis*)

- **Tworzenie terenu:** Należy zaimplementować klasę reprezentującą mapę terenu, która będzie tworzyła otwarty świat za pomocą pliku opisującego teren.
- **Stworzenie samolotu:** Samolot powinien móc latać, strzelać oraz powinien być podatny na uszkodzenia związane z możliwymi zderzeniami z terenem świata
- **Sterowanie samolotem:** Sterowanie odbywa się za pomocą myszki oraz przycisku W i S, które odpowiednio zwiększają oraz zmniejszają prędkość samolotu. Ruch myszki natomiast determinuje w którą stronę samolot będzie leciał. Przyciśnięcie LPM powoduje wystrzał pocisku z samolotu.
- **Utrata punktów życia:** W wyniku zderzenia się z powierzchnią terenu samolot traci punkty życia, które się nie regenerują wraz z trwaniem gry

Projekt techniczny (*technical design*)

Opisy klas



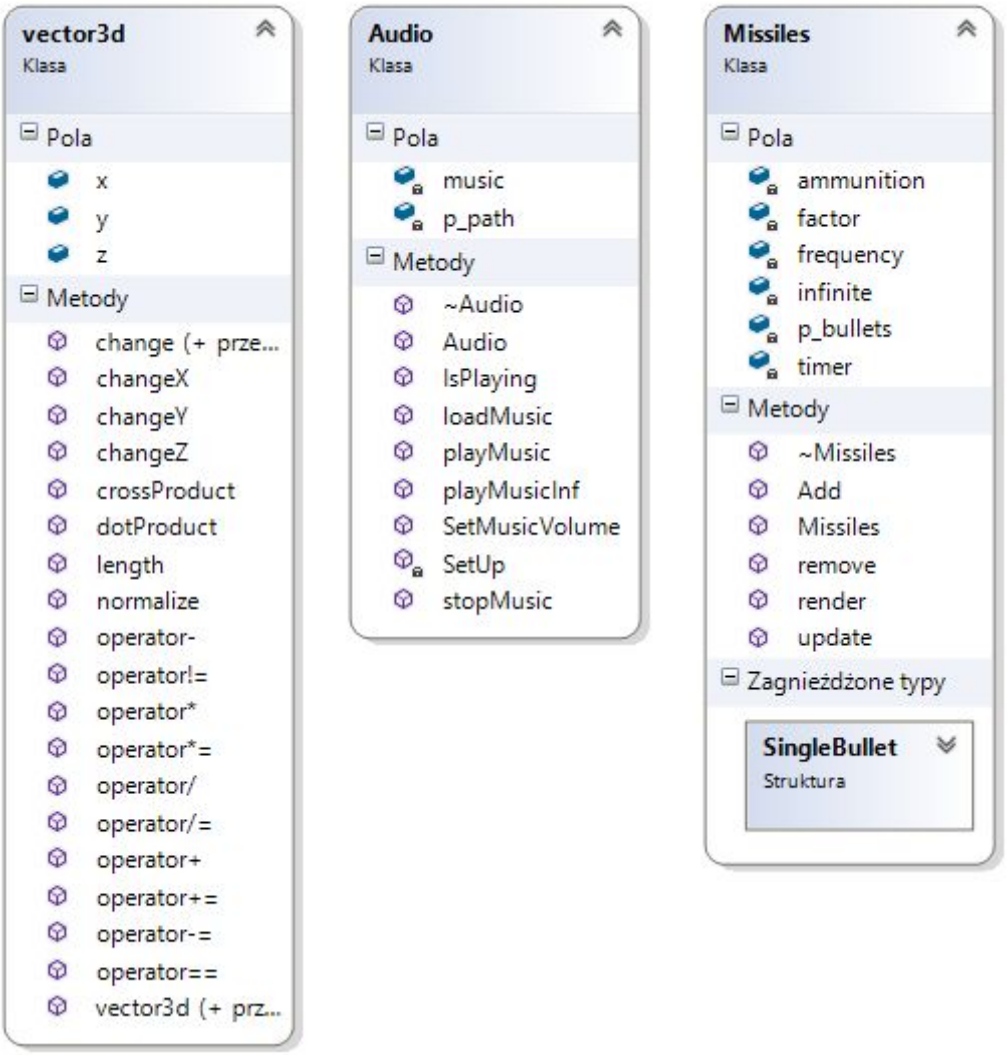
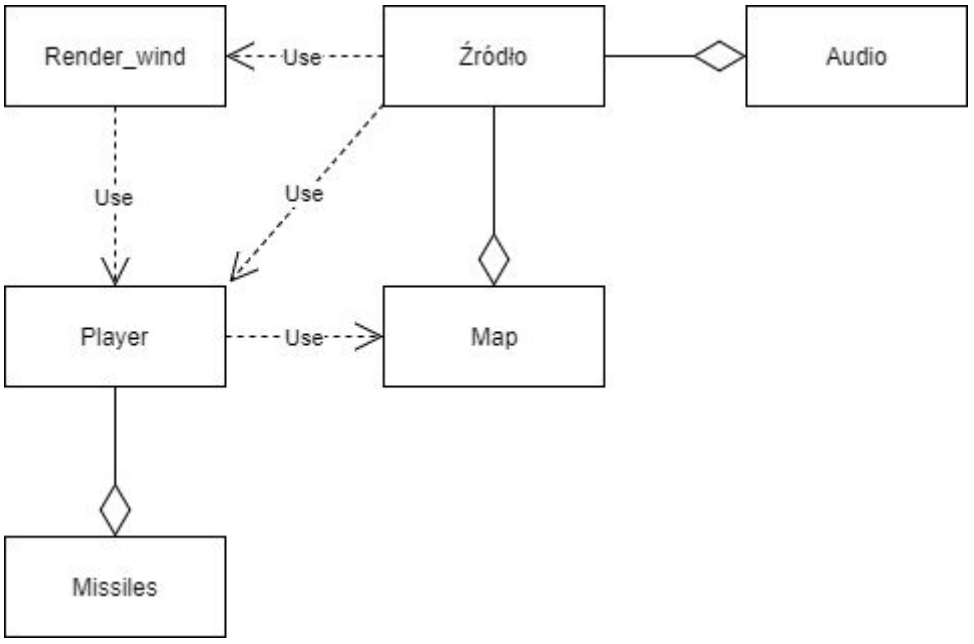

vector3d
Klasa
Pola
x
y
z
Metody
change (+ prze...
changeX
changeY
changeZ
crossProduct
dotProduct
length
normalize
operator-
operator!=
operator*
operator*=
operator/
operator/=
operator+
operator+=
operator-=
operator==
vector3d (+ prz...
Audio
Klasa
Pola
music
p_path
Metody
~Audio
Audio
IsPlaying
loadMusic
playMusic
playMusicInf
SetMusicVolume
SetUp
stopMusic
Missiles
Klasa
Pola
ammunition
factor
frequency
infinite
p_bullets
timer
Metody
~Missiles
Add
Missiles
remove
render
update
Zagnieżdżone typy
SingleBullet
Struktura

Diagram klas



Opis realizacji (*implementation report*)

Program został napisany w Visual Studio 2015 wykorzystując dodatkowe biblioteki takie jak SFML 2.1 oraz SDL2. Do zrealizowania warstwy graficznej użyłem interfejsu OpenGl. Do bardziej skomplikowanych rozwiązań jakie użyłem w programie można zaliczyć:

- **Wczytywanie terenu:** Operacja wytworzenia terenu wewnątrz programu polega na wczytaniu dwóch plików, jednego odpowiadającego za rozkład wysokości oraz drugiego odpowiadającego za kolory w danych punktach. Obie mapy bitowe mają rozdzielczość 1081 x 1081 px. Po załadowaniu plików zostają one użyte do stworzenia VBO, które przechowuje dane potrzebne do wyrenderowania terenu w pamięci RAM. Metoda ta jest o wiele szybsza niż poprzednia, która polegała na ciągłym tworzeniu i renderowaniu terenu, przez co można było poszerzyć obszar tworzonego świata do 3600 km^2 (świat to kwadrat 60km x 60km).

- **Tworzenie i usuwanie pocisków:** Pociski są tworzone w klasie Missiles, po wcześniejszym dodaniu pocisku (w klasie Player) do przechowującego je wektora *p_bullets*. Pocisk jest tworzony po naciśnięciu LMP w lokalizacji samolotu, wraz z jego kątem położenia w przestrzeni. Dzięki temu zabiegowi można z łatwością wystrzelić pocisk w kierunku w którym w danej chwili znajdował się samolot. Usuwanie pocisku następuje automatycznie w momencie gdy przeleci on zadaną mu odległość w jednej z 3 płaszczyzn (x,y lub z).

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

Kod usterki	Data	Autor	Opis	Stan

Podręcznik użytkownika (*user's manual*)

Opis gry

Gra odbywa się na kwadratowej mapie 3D, reprezentującej wycinek realnego obszaru Norwegii wokół miasta Alesund o wymiarach 60km x 60km. W trakcie gry mamy nieograniczoną możliwość eksploracji terenu lecącym samolotem. Samolot może się rozbijać o elementy terenu, tracąc przy tym punkty życia. Ponadto wraz z poruszaniem się samolotu, jego poziom paliwa stale maleje. Oprócz tego mamy jeszcze możliwość strzelania nieograniczoną liczbą pocisków.

Na ekranie wyświetlają nam się informacje na ekranie odnośnie pozostałych punktów życia (po prawej stronie) oraz pozostałego paliwa (po lewej stronie).

Sterowanie

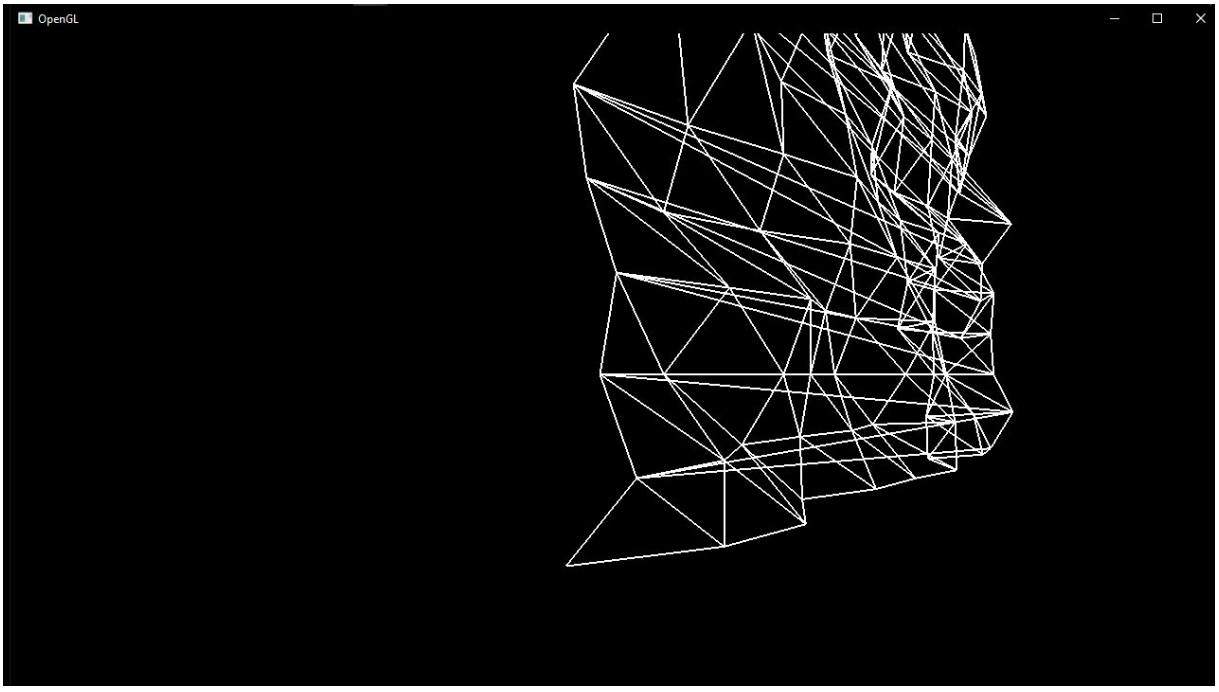
- W - zwiększenie prędkości samolotu
- S - zmniejszenie prędkości samolotu
- Ruch myszki - sterowanie samolotem
- LPM - wystrzał pocisku

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

- Dodanie przeciwników oraz beczek paliwa uzupełniających paliwo samolotu
- Dodanie menu głównego oraz menu pauzy podczas gry
- Dodanie kolejnych poziomów
- Dodanie poziomów trudności (np. NPC którzy po wykryciu samolotu zaczęli by go gonić i ostrzeliwać)
- Dodanie efektów graficznych takich jak dynamiczna kamera lub efekty wybuchu pocisków podczas zderzenia

Opis zmian generowania terenu

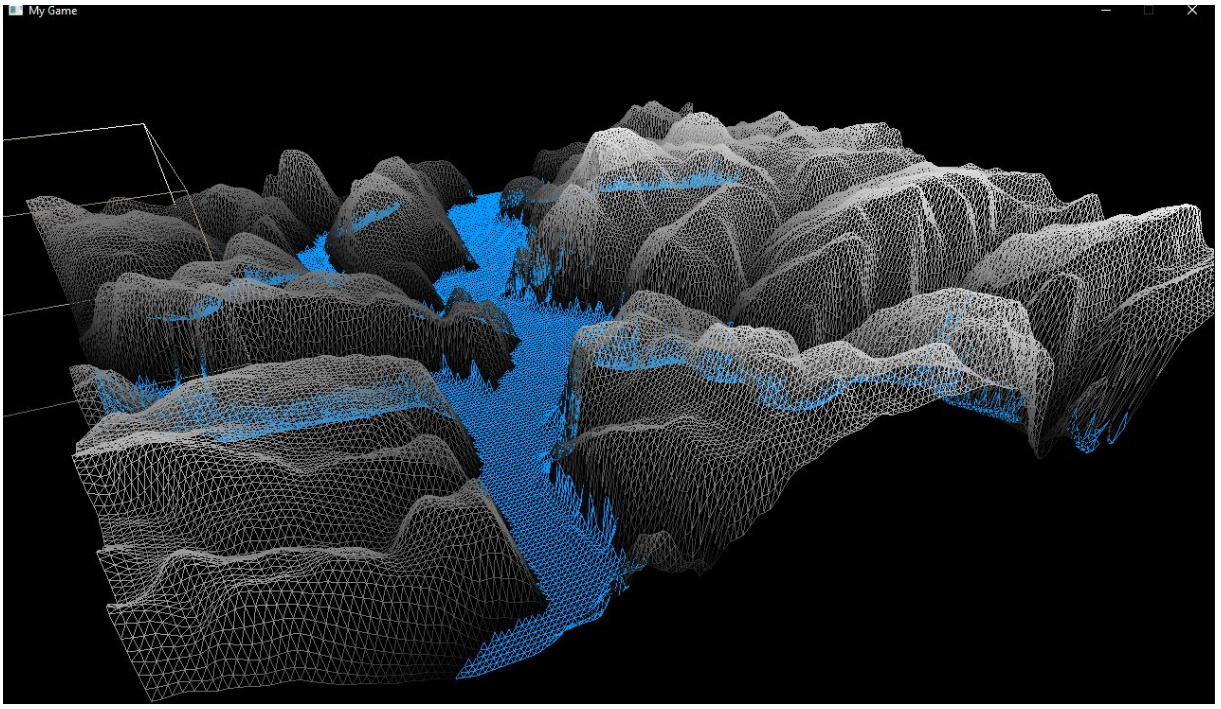
Na początku trzeba było zacząć od rzeczy najprostszych, czyli zrobienia podstawowej siatki, która następnie została “wyrzeźbiona” przez zastosowanie na niej szumu Perlina, który nadał terenowi kształt. Niestety metoda ta była mało wydajna i obszar ograniczał się do rozmiaru 100x1000 kratek, co nie było dla mnie zadowalające, ponieważ taką mapę, samolot przelatywał w ciągu minuty.



Metoda ta opierała się na każdorazowym generowaniu całej siatki, przez co płynne działanie gry było ograniczone przez rozmiar terenu. W związku z tym musiałem wymyślić nowy sposób generowania terenu. Wybór padł na wykorzystanie pamięci RAM, gdzie przechowywane są dane potrzebne do wygenerowania terenu. Metoda ta polega na wczytaniu dwóch plików: mapy wysokości, która jest przerabiana na koordynaty x,y,z reprezentujące punkt na mapie i jego wysokość oraz mapy kolorów, która przypisuje każdemu punktu kolor, podobnie jak w mapie wysokości.

Po wykorzystaniu drugiej metody, powstały świat można było rozszerzyć do rozmiaru kwadratu o boku 60 km (tyle obejmuje mapa wysokości użyta do stworzenia terenu).

Widok po wstępnym wczytaniu terenu (bez koloryzacji):



Widok końcowy: