POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

GRAFIKA KOMPUTEROWA

DOKUMENTACJA

Jakub Biel

EF-DI 2

L01

1.Wstęp.

Celem projektu było utworzenie wiernej symulacji przelotu balonu wykonanej w środowisku 3D.

2. Założenia projektowe, środowisko programowe i zasoby.

2.1 Wybór obiektu modelowania graficznego.

Balonu który wybrałem jeden z największych dostępnych balonów: RQ10-1

2.2 Dane techniczne balonu.

- Wysokość: 27 m
- Maksymalna ładowność: 1000 kg
- Powłoka:
  - Objętość: 6000m3
  - Materiał: Poliester
  - Wysokość: 25 m.
  - Maksymalna średnica pozioma: 23 m.
  - Waga 168 kg.
  - Maksymalna dopuszczalna temperatura: 120^{o}
- Kosz:
  - Materiał: Retten
  - Kształt: Kwadrat
  - Wymiary: 2,7 x 1,3 x 1,1 [m]
- Palink:
  - Ilość: 4
  - Paliwo: Propan bądź LPG
- Zbiornik Paliwa:
  - Ilość: 6
  - Objętość Zbiornika: 48 l.

2.3 Środowisko programowe.

Cały kod projektu jest wykonany w języku programowania c++, jako IDE użyłem środowiska Visual Studio 2013.

2.4 Zastosowane biblioteki i zasoby.

Używam biblioteki OpenGL 4, OpenGL Mathematics(glm), freeGlut do obsługi okna oraz AntTweakBar do interfejsu graficznego.

Cały geometria została utworzona w kodzie programu, bez użycia zewnętrznych aplikacji do modelowania, bądź generowania. Kalkulacje normalnych czy budowanie list sąsiedztwa również jest wykonywane w trakcie działania programu.

3. Etapy projektowania

3.1 Tworzenie obiektów bryłowych.

By zaznajomić się z api opengl'a zacząłem od konstruowania oraz renderowania prostych obiektów bryłowych typu kula czy sześcian.

Lecz zanim do tego doszło musiałem nauczyć składni języka oraz zasady działania shaderów GLSL. Po napisaniu podstawowego shader'a przyjmującego tylko jeden parametr - pozycje wierzchołka zacząłem konstruowanie geometrii.

Na tym etapie nie używałem jeszcze indeksowania wierzchołków służącego do zwiększania wydajności i czytelności kodu.

Po udanym renderowaniu podstawowych brył zacząłem konstrukcje bryły balonu.

Główna część balona została skonstruowana jako walec z wierzchołkami rozmieszczonymi w funkcji sinusoidy. Konstrukcja postępowała od najwyższego wierzchołka do najniższego.

Od pewnego punktu średnica walca było po prostu skalowana co pozwoliło uzyskać poprawny kształt balonu.

3.2 Modelowanie obiektów pomocniczych.

Na tym etapie wprowadziłem indeksowanie wierzchołków oraz trzy tryby renderowania:

1. Pełna geometria.
2. Siatka.
3. Punkty.

Sam płaszcz balona bez koszyka czy lin wspomagających wygląda jak żarówka. By zapobiec tej błędnej percepcji należy dodać obiekty pomocnicze. Dodałem więc koszyk i liny łączące z balonem. By nadać wrażenie wielkości i odległości od powierzchni utworzyłem siatkę służąca jako powierzchnia ziemi.

3.3 Wprowadzanie kolorystyki oraz oświetlenia

By nadać większą kontrole dla obserwacji wprowadziłem kamerę oraz 3 tryby jej pracy.

1. Kamera z perspektywy 1 osoby - wolno latająca.
2. Kamera z balona.
3. Kamera w której balon jest zawsze w centrum pola widzenia .

By wprowadzić kolorystykę musiałem napisać nowy vertex shader przyjmujący 2 attrybuty - pozycje oraz kolor.

Po wprowadzeniu kolorystyki dla balonu i środowiska zacząłem prace nad oświetleniem.

Jak poprzednio wymagało to utworzenia shader'a tym razem przyjmującego 3 atrybuty pozycje, kolor oraz normalną.

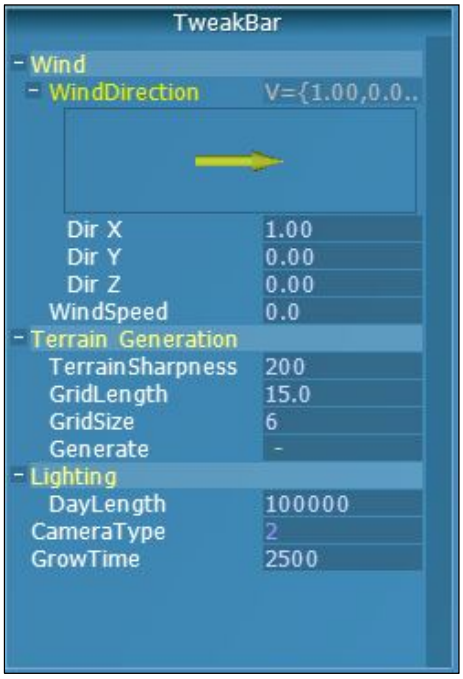
Jako że normalna jest przypisywana do każdego wierzchołka i interpolowana po powierzchni trójkąta wymaga dość sporo kalkulacji.

Najpierw należy obliczyć normalne wszystkich ścian danego wielokątu, odnaleźć ściany należący do każdego wielokątu a następnie zsumować oraz znormalizować.

Po dodaniu uniform'a z kierunkiem światła oświetlenie działało już poprawnie.

3.4 Włączanie interfejsu operatora

Do utworzenia interfejsu użyłem biblioteki AntTweakBar. Po ustawieniu funkcji przekierowujących wejście klawiatury i myszki do funkcji AntTweakBar w bardzo prosty sposób udało mi się utworzyć interfejs graficzny. Jego wygląd znajduje się na rys.1.



Rys. 1 Interfejs AntTweakBar

3.5 Projektowanie scenariuszy dla aplikacji.

Podstawowym scenariuszem jest przelot balonem nad krajobrazem pod wpływem wiatru.

Istnieje także możliwość skoku z balonu gdy jest sie w kamerze widoku z balona.

Podczas przebywania w kamerze wolno latającej z pierwszej osoby, jest możliwość ruchem przyśpieszonym przyciągniecie do balona. Jest szczególnie istotne jeżeli chcemy dostrzec wielkość balona w stosunku do gór.

3.6 Generacja terenu.

Do generowania terenu użyłem algorytmu "Diamond-Square" opartego na metodzie "Midpoint Displacement" by stosunkowo prostu proceduralnie wygenerować realistyczny teren.

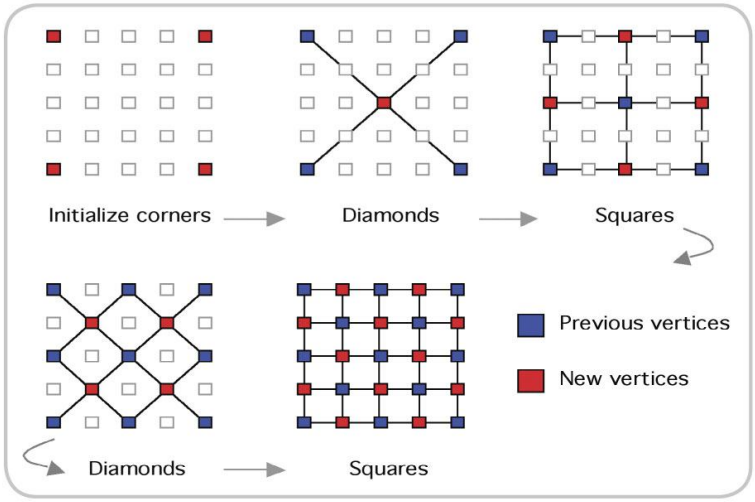


Figure 10 Steps of the diamond-square algorithm

Rys. 2 Graficzne przedstawienie działania algorytmu "Diamond-Square".

Na początku dowolne wartości ustawiane są na 4 krawędzie mapy. Następnie do momentu ustawienia wszystkich wartości wykonywane są kolejno kroki "Diamond" i "Square". Z każdą iteracją zmniejsza się o połowę wielkość szerokości kwadratu jak i losową wartość offsetową.

3.7 Testowanie aplikacji.

Na tym etapie głównie poświęciłem czas na dostosowywanie generatora terenu i dostosowywanie wielkości obiektów.

4. Poprawianie i ulepszanie aplikacji

4.1 Redukcja zbędnych linii kodu.

Poprawiłem wiele funkcji redukując liczbę linie kodu głównie w funkcjach indeksujący wierzchołki oraz obliczających listy sąsiedztwa czy normalne.

4.2 Dodanie komentarzy.

Po przeglądnięciu kodu zobaczyłem dużo miejsc które mogły sprawić trudność osobie czytającej kod w jego zrozumieniu. Dodałem więc komentarze bardziej opisujące działanie w miejscach dwuznacznych oraz by opisać części funkcji.

4.3 Poprawienie struktury programu.

Podczas nauki api opengl'a wielokrotnie zdarzyło mi się dość długo szukać miejsca występowania jakieś funkcji czy obiektu, dlatego jako, że c++ jest językiem obiektowym postanowiłem w pełni pogrupować funkcjonalności i parametry obiektów w klasy.

Pogrupowałem też części funkcji odpowiedzialne za różne czynności w długich funkcjach w następne funkcji.

Czynność ta bardzo zwiększyła czytelność i łatwość rozszerzania kodu.

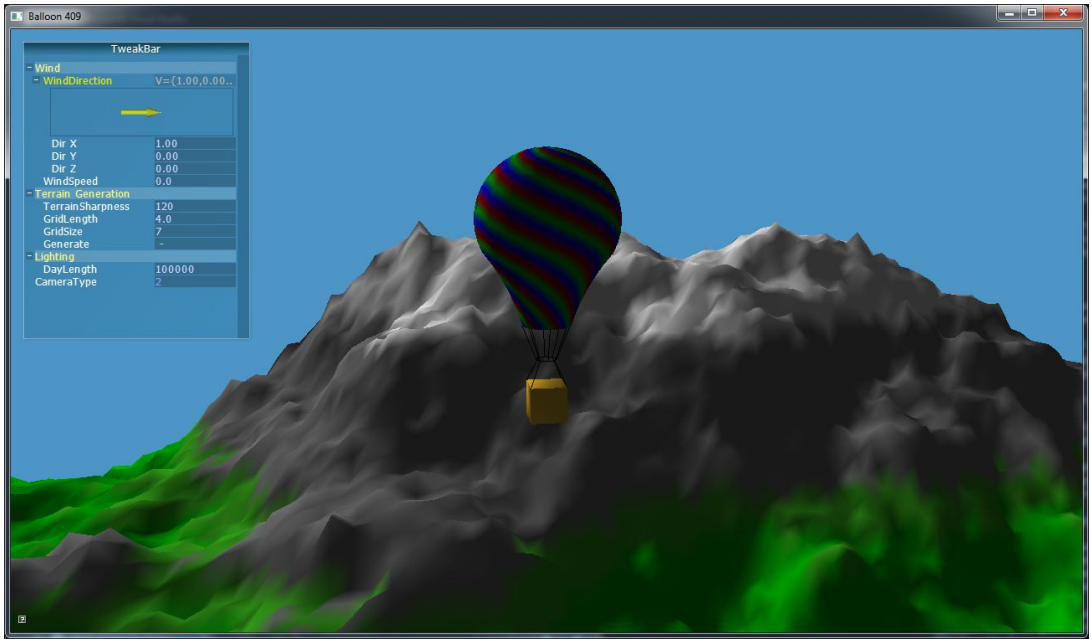
4.4 Poprawienie wydajności.

Nie musiałem dużo szukać by znaleźć najwolniejsze części mojego kodu.

Najwolniejszą rzeczą okazała kalkulacja normalnych, niespodziewanie to szukanie obecności wierzchołka w ścianie wymagała dużej ilości czasu dlatego też utworzyłem funkcje tworzącą listę sąsiedztwa dla siatki o konkretnym rozmiarze.

Poprawiłem też obsługę wektorów gdyż ze względu na specyfikę ich obsługi pamięci wiedząc ich rozmiar końcowy oczywiste było alokacja pamięci na początku zamiast używania metody push_back.

5. Obsługa Aplikacji



Rys. 3 Główne okno aplikacji.

5.1. Panel TweakBar

W panelu TweakBar można odnaleźć pola i kontrolki służące do sterowania aplikacją.

W karcie Wind można sterować kierunkiem wiatru, oraz ustawić jego prędkość poprzez zmianę wartości pola WindSpeed.

W karcie Terrain Generation można odnaleźć kontrolki służące do sterowania parametrami Generatora Terenu. Pole TerrainSharpness odpowiada za górzystość terenu, pole GridLength za odstęp pomiędzy wierzchołkami siatki terenu, pole GridSize za logarytm szerokości siatki, szerokość siatki jest określana wzorem:

$$szerSiatki = 2^{GridSize} + 1$$

Rozmiar i kształt wymiarów siatki jest wymuszony specyfiką algorytmu generującego.
Po naciśnięciu przycisku Generate teren zostanie wygenerowany ponownie.
Pole DayLength zawiera ilość sekund potrzebnych do pełnego okrążenia wektora kierunku światła wokół osi y.

Pole CameraType służy do wyboru typu kamery. Istnieją 3 typy kamery.

1. Kamera z perspektywy 1 osoby - wolno latająca.
2. Kamera z balona.
3. Kamera w której balon jest zawsze w centrum pola widzenia.

By zmienić tryb renderowania należy wcisnąć klawisz 'q' co spowoduje cykliczne jego zmienianie, kolejno - punkty - siatka - trójkąty.

5.1. Sterowanie ruchem

Podczas przebywania w perspektywie 1 osoby można się poruszać klawiszami WSAD, po wciśnięciu klawisza '4' zostaniemy przyciągnięci do balona i zostanie zmieniony tryb kamery na tryb balona. Będąc w perspektywie balona istnieje możliwość skoku po wciśnięciu klawisza '3'.

Skok z balona trwa do kolizji z terenem, po kolizji z terenem wykonywany jest powrót do balona.

Dodatkowe funkcje klawiszy:

- 'y' - zmienia kąt światła kierunkowego wokół osi z,
- 'm' - zwiększa prędkość balonu,
- 'k', 'l' - rotacja balonu wokół osi y,
- 'r' - regeneracja terenu,
- '5', '6' - modyfikacja GridLength,
- '1', '2' - modyfikacja wartości ostrości terenu,
- 'c' - zmiana typu kamery
- 'scrollem' w myszce można regulować odległość od balonu w kamerze nr.3.

6. Podsumowanie

Celem laboratorium było poznanie biblioteki opengl, współczesnych metod graficznych oraz utworzenie symulacji przelotu balonem.

Uważam, że cel ten w większości zostało przeze mnie osiągnięty.

Dużo rzeczy które wydawałyby się łatwe okazały się trudne do implementacji.

Zaskoczyło mnie także jak bardzo użyteczne mogą być funkcje sinusoidalne czy też krzywe beziera do implementacji dość skomplikowanych brył.

6. Literatura

[1] http://www.arcsynthesis.org/gltut/

[2] http://en.wikipedia.org/wiki/Diamond-square_algorithm