Grafika Komputerowa - Dokumentacja Projektu

Tomasz Herman

14 lutego 2020

1 Wstęp

Opis funkcjonalności, implementacji i instrukcja użycia. Źródła projektu dostępne na moim Githubie.

2 Użyte technologie

Program używa w 100% języka Java. Do interfejsu graficznego użyta została wbudowana w praktycznie każdą wersję Javy biblioteka Swing. Do wczytywania modeli program wykorzystuje bibliotekę Assimp, a do wczytywania tekstur bibliotekę STBImage. Obie te biblioteki są zawarte w większej bibliotece LWJGL. Jako bibliotekę matematyczną program wykorzystuje JOML.

3 Klawiszologia

- W,A,S,D - poruszanie się prosto, na prawo, do tyłu, na lewo
- Q,E - poruszanie się do góry, w dół
- Z,X - rotacja w okół osi Z kamery
- F - włączenie/wyłączenie trybu pełnoekranowego
- C - przełączanie się między kamerami
- 1,2,3,4,5,6 - zmiana funkcji cieniowania
- przeciąganie myszą - obrót kamery(tylko wolna kamera)
- kółko myszy - zmiana kąta widzenia(przybliżanie/oddalanie)

4 Implementacja

4.1 Window.java

Klasa odpowiadająca za obsługę okienka wyświetlanego na ekranie. Obsługuje przejście do trybu pełnoekranowego. Jest kontenerem dla innych komponentów graficznych.

4.2 Canvas.java

Klasa która pozwala rysować za pomocą funkcji setPixel(int x, int y, int rgb). Pod maską klasy znajduje się tablica intów odpowiadających kolorom, związana z obiektem który wyświetla te kolory na ekranie. Oprócz tablicy pikseli jest także tablica głębokości odpowiadająca z-buforowi. Canvas reaguje na zmiany rozdzielczości przeskalowując swoje tablice i aktualizując swój stan wewnętrzny.

4.3 Renderer.java

Przechowuje w sobie Canvas po którym rysuje. Zawiera w sobie informacje niezbędne do zbudowania macierzy rzutowania(oprócz tych zawartych w Canvas). Pomocnicza klasa Transformation pozwala w łatwy sposób uzyskiwać macierze potrzebne do przekształceń. Istotne jest pole przechowujące funkcję rysującą trójkąty, które pozwala na szybką jej podmianę. Metoda renderScene(Scene scene) czyści Canvas i rysuje zadaną scenę używając wybranej funkcji rysującej trójkąty, podczas rysowania funkcja wykorzystuje wielowątkowość. Klasa zawiera w sobie kilka wbudowanych funkcji do rysowania trójkątów:

- renderTriangleWireframe - do rysowania siatki trójkątów bez wypełniania, używa algorytmu Bresenhama
- renderTriangleSuperFlat - najprostsza funkcja cieniująca wypełnia trójkąt kolorem wyliczonym z modelu Phonga w środku trójkąta, nie używa tekstur
- renderTriangleFlat - funkcja cieniująca liczy intensywność światła w środku trójkąta, a potem wypełnia trójkąt teksturą z uwzględnieniem wyliczonego światła
- renderTriangleGouraud - wylicza intensywność światła w wierzchołkach trójkąta, a potem wypełnia trójkąt teksturą z uwzględnieniem wyliczonego światła zinterpolowanego w wierzchołkach
- renderTrianglePhong - wypełnia trójkąt licząc kolor w każdym punkcie trójkąta, interpoluje atrybuty w wierzchołkach, teksturuje
- renderTrianglePhongSpecularPhong - wypełnia trójkąt licząc kolor w każdym punkcie trójkąta, interpoluje atrybuty w wierzchołkach, teksturuje, używa specular z modelu Phonga

Warto zaznaczyć że wszystkie funkcje wypełniające trójkąt korzystają z algorytmu opisanego dokładniej tutaj. Dzięki temu algorytmowi możemy wypełniać trójkąty w pętlach operujących na liczbach całkowitych oraz mamy darmowe obcinanie do prostokąta ekranu. Podczas renderowania dla wydajności zastosowany jest back face culling oraz frustum view culling.

4.4 Scene.java

Prosta klasa agregująca wszystkie światła, obiekty i kamery.

4.5 LightSetup.java

Klasa agregująca światło. Zawsze zawiera jedno AmbienLight, jedno DirectionalLight oraz dowolną ilość SpotLight i PointLight.

4.6 Camera.java

Interfejs reprezentujący kamerę. Klasy implementujące muszą zapewnić dostęp do maczierzy widoku, oraz reagować w jakiś sposób na ruch i obrót(albo i nie).

4.7 GameObject.java

Klasa reprezentująca jakiś obiekt znajdujący się w świecie. Przechowuje referencję na model obiektu oraz pozycję i rotację obiektu.

4.8 Model.java

Obiekt składający się z listy siatek.

4.9 Mesh.java

Siatka składająca się z trójkątów. Zawiera jeden Material będący fizycznym odzwierciedleniem cech danej siatki. Przechowuje tablicę trójkątów oraz dodatkowo tablicę wierzchołków, żeby przy operacjach per vertex nie przetwarzać wielokrotnie tych samych wierzchołków. Dodatkowo podczas tworzenia obiektu klasy Mesh wyliczana jest bryła brzegowa siatki potrzebna później do frustum cullingu.

4.10 Material.java

Zbiór cech fizycznych takich jak: ambientColor, diffuseColor, specularColor i shininess. Zawiera także różne tekstury czyli diffuseTexture, specularTexture, ambientTexture, normalsTexture oraz informację czy te tekstury istnieją.

4.11 Triangle.java

Agreguje trzy wierzchołki.

4.12 Vertex.java

Zawiera informacje o wierzchołku takie jak pozycja w koordynatach świata, kamery i ekranu, oraz tangent space w koordynatach świata i kamery. Przez funkcję transform(Matrix4f MVP, Matrix4f modelViewMatrix, Matrix3f normalMatrix, int right, int bottom) pozwala w prosty sposób przekształcić pozycję i tangent space do odpowiednich systemów koordynatów.

4.13 Texture

Klasa reprezentująca teksturę z której można pobierać próbkę będącą współrzędnymi tekstury. Oprócz kanałów czerwonego niebieskiego i zielonego zawiera kanał Alpha. Podczas tworzenia tekstury wyliczany jest średni kolor tekstury.

4.14 ModelLoader

Ma za zadanie wczytywać modele razem z teksturami we wszystkich formatach jakie istnieją.

4.15 TextureManager

Dba o to aby podczas wczytywania modelu tekstury które się powtarzają nie były wczytywane wielokrotnie.

4.16 Transformation

Klasa pomocnicza zwracająca macierz przekształceń dla zadanego obiektu/kamery.

4.17 Color3f

Reprezentuje kolor jako trzy liczby float od 0 do 1. Nie dba o to żeby kolory były z odpowiedniego zakresu, aż do wywołania funkcji clamp.

4.18 Engine

Główna klasa odpowiadająca za całą logikę świata. W pętli wywołuje w odpowiednich momentach funkcję render i update. Zawiera scenę, renderer i canvas. Obsługuje zdarzenia klawiatury/myszy.

5 Kamery

5.1 FirstPersonCamera

Pozwala na dowolne ruchy i obroty w przestrzeni.

5.2 FollowingCamera

Śledzi zadany GameObject. Pozycja jest niezmienna. Kamera obraca się tylko wtedy gdy rusza się obiekt, który kamera śledzi.

5.3 ThirdPersonCamera

Śledzi zadany GameObject. Kamera dostosowuje swoją pozycję i rotację do pozycji i rotacji śledzonego obiektu.

6 Światła

Wszystkie światła zawierają funkcję przekształcającą ich atrybuty do koordynatów kamery dla zadanej macierzy widoku. Każde światło posiada też swój kolor.

6.1 AmbientLight

Globalne oświetlenie sceny.

6.2 DirectionalLight

Światło naśladujące działanie Słońca. Posiada kierunek.

6.3 PointLight

Światło skupione w punkcie. Posiada pozycję.

6.4 SpotLight

Reflektor o zadanej pozycji i kerunku.

7 Funkcje cieniujące

Dla wszystkich technik wypełniania gdzie użyte jest teksturowanie użyta jest techinka alpha discardingu. Piksele w których dla tekstury alpha jest mniejsza od ustalonej wartości nie są rysowane. Oświetlenie jest wyliczane dla każdego światła znajdujące się na scenie. Dla każdej metody dla każdego piksela przeprowadzany jest depth test.

7.1 renderTriangleWireframe

Rysuje druty siatki trójkątów. Używa algorytmu Bresenhama do rysowania odcinków. Odcinki są zabarwiane średnim kolorem tekstury obiektu lub jeżeli obiekt nie posiada tekstury to kolorem białym.

7.2 renderTriangleSuperFlat

Najprostsza funkcja cieniująca wypełnia trójkąt jednym kolorem wyliczonym z modelu Phonga w środku trójkąta, nie używa tekstur.

7.3 renderTriangleFlat

Funkcja cieniująca liczy intensywność światła w środku trójkąta, a potem wypełnia trójkąt teksturą z uwzględnieniem wyliczonego światła.

7.4 renderTriangleGouraud

Wylicza intensywność światła w wierzchołkach trójkąta, a potem wypełnia trójkąt teksturą z uwzględnieniem wyliczonego światła.

7.5 renderTrianglePhong

Wypełnia trójkąt licząc kolor w każdym punkcie trójkąta, interpoluje atrybuty w wierzchołkach, teksturuje.

7.6 renderTrianglePhongSpecularPhong

Wypełnia trójkąt licząc kolor w każdym punkcie trójkąta, interpoluje atrybuty w wierzchołkach, teksturuje, używa specular z modelu Phonga.