WMM – Lab 8
Generowanie grafiki z wykorzystaniem popularnej biblioteki graficznej

Autor: Łukasz Dąbała

1 Wstęp

Celem laboratorium 8 jest zapoznanie się z biblioteką graficzną OpenGL oraz podstawami programowania na GPU. W tym celu do zrealizowania będą 2 ćwiczenia poruszające zagadnienia wyświetlania grafiki z jej użyciem.

Przydatne linki:

1. Dokumentacja OpenGL - https://www.khronos.org/registry/OpenGL-Refpages/gl4/

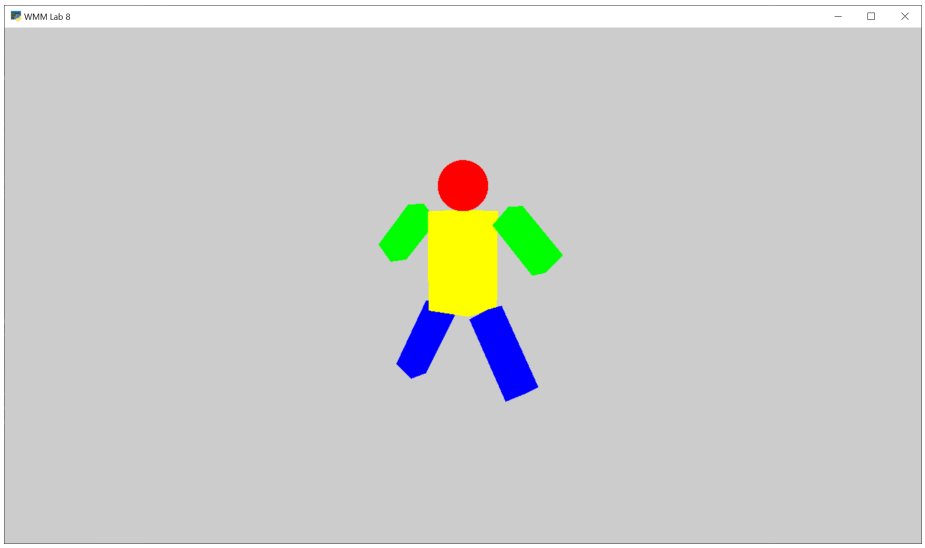
2 Wymagania

W celu realizacji laboratorium niezbędny jest interpreter języka Python w wersji 3.9. Aby ułatwić też sobie pracę, preferowanym narzędziem do pisania kodu będzie środowisko programistyczne PyCharm. Ze względu na pracę z shaderami, przydatna może być też wtyczka do tego programu: GLSL Support, która dodaje podstawowe kolorowanie składni.

W plikach dołączonych do zadania, znajduje się kod stanowiący podstawę do modyfikacji w trakcie trwania laboratorium. W środku archiwum znajdują się pliki Pythonowe oraz wzorcowe shadery, które będą mogły stanowić bazę do wykonania ćwiczeń. Sam program umożliwia wczytanie shaderów z plików *.frag* (*fragment shader*) i *.vert* (*vertex shader*) oraz wyświetlenie podstawowego okna. Aby ułatwić instalację niezbędnych pakietów w paczce znajduje się plik *requirements.txt*.

3 Sprawozdanie i kod

Kod tworzony w ramach laboratorium powinien być napisany w sposób zrozumiały oraz powinien być skomentowany. Komentarze w kodzie powinny tłumaczyć co dzieje się w danym miejscu np. *obliczanie wektora z danego wierzchołka w kierunku źródła światła*. Do opisu działania można wykorzystać zarówno język polski i angielski.



Rysunek 1: Przykładowe złożenie transformacji dla obiektów w cel ułożenia z nich robota.

Dodatkowo należy stworzyć sprawozdanie, które będzie dokumentować działanie stworzonych programów. Opis każdego zadania opisuje również elementy, które powinny znaleźć się w sprawozdaniu.

4 Składanie transformacji (2.0 pkt)

Celem tego zadania jest składanie transformacji obiektów w scenie 3D. Pierwszym etapem zadania jest wczytanie obiektów z plików *.obj* - sfery oraz sześcianu. Następnie należy stworzyć konkretną ilość obiektów oraz zaaplikować do nich serię przekształceń w celu zbudowania 'robota', który powinien wyglądać jak na rysunku 1.

Podstawowe kształty, które dostarczone są w ramach zasobów do laboratorium należy poddać modyfikacjom:

1. głowa - translacja o wektor $t = (0, 0, 5)$
2. ciało - translacja o wektor $t = (0, 0, 2)$, skalowanie w osi z ze współczynnikiem 2
3. ręce - translacja o wektor $t = (0, +/-3, 3)$, rotacja o kąt $+/-45^o$ wokół osi x, skalowanie ze współczynnikami $s = (0.5, 0.5, 1.25)$
4. nogi - translacja o wektor $t = (0, +/-2, -1.5)$, rotacja o kąt $+/-30^o$ wokół osi x, skalowanie ze współczynnikami $s = (0.5, 0.5, 1.75)$

Jak można zauważyć obiekty są również pokolorowane na różne sposoby, w związku z tym należy zmodyfikować podstawowe shadery.

1. **vertex shader** - dodanie macierzy transformacji, która jest identyczna dla wszystkich wierzchołków (*uniform mat*4) oraz jej aplikacja do wierzchołków
2. **fragment shader** - dodanie koloru, który jest identyczny dla wszystkich fragmentów (*uniform vec*3)

Do konstrukcji macierzy i wektorów należy wykorzystać pakiet *pyrr*. Przydatne mogą być następujące metody:

1. `Matrix44.from_x_rotation`

   - macierz obrotu wokół osi X tworzona z podanej rotacji

2. `Matrix44.from_translation`

   - macierz translacji tworzona z wektora

3. `Matrix44.from_scale`

   - macierz skalowania tworzona z wektora skali dla poszczególnych osi

4. `Matrix44.from_eulers`

   - macierz obrotu tworzona z podanych kątów Eulera

5. `Matrix44.perspective_projection`

   - macierz projekcji perspektywicznej

6. `Matrix44.look_at`

   - macierz w celu skierowania wzroku na dany punkt

W sprawozdaniu należy zawrzeć wizualizację stworzonego robota w innym kolorze niż ten pokazany w instrukcji.

5 Cieniowanie (3.0 pkt)

Celem tego zadania będzie dodanie cieniowania - obiekty nie powinny być jednorodną plamą na ekranie. Aby ułatwić sobie zadanie należy poczynić pewne założenia.

1. Algorytmem do implementacji jest cieniowanie Phonga.
2. Należy włączyć test głębi i usuwanie powierzchni niewidocznych.

3. W vertex shaderze konieczne jest zrobienie przekształcenia z wykorzystaniem macierzy projekcji oraz widoku.

4. W fragment shaderze należy zaimplementować cieniowanie - obsługujemy pojedyncze źródło światła:

   (a) pozycja światła i jego kolor (*ambient*, *diffuse*, *specular*) zdefiniowane są na stałe w shaderze

   (b) kolor obiektu (*diffuse*, *specular*) przekazywane są z zewnątrz

   (c) połyskliwość obiektu (*shininess*) - przekazywana jest z zewnątrz

GLSL oferuje szereg funkcji, które ułatwią wykonanie tego zadania:

1. `transpose(V/M)`

   - transpozycja macierzy M/wektora V

2. `inverse(M)`

   - odwrotność macierzy M

3. `normalize(V)`

   - normalizacja wektora V

4. `reflect(I, N)`

   - oblicza promień odbity od powierzchni z wykorzystaniem promienia padającego I oraz wektora normalnego N

5. `dot(V1, V2)`

   - iloczyn skalarny wektorów V1 i V2

6. `pow(a, b)`

   - podnosi liczbę a do potęgi b

7. `max(a, b)`

   - wartość maksymalna z dwóch wartości a i b

Modelem testowym do zadania może być zarówno pojedyncza kulka jak i robot skonstruowany w ramach poprzedniego zadania - należy wtedy pamiętać, żeby również przekazać kolor poszczególnych części robota.

W sprawozdaniu należy zawrzeć wizualizacje dla różnych kombinacji parametrów wejściowych (położenie źródła światła, różna połyskliwość obiektu, różne kolory obiektu oraz światła).