Centro de Informática

Universidade Federal da Paraíba - UFPB

Engenharia da computação

Atividade 03

Relatório

Disciplina: Introdução a Computação Gráfica

Professor: Christian Azambuja Pagot

Dupla: JOHAN KEVIN ESTEVÃO DE FREITAS - 20170171741

GABRIEL FORMIGA BATISTA - 11503377

2020

Objetivo

O objetivo deste trabalho prático é familiarizar os alunos com a estrutura do pipeline gráfico através da implementação das transformações geométricas que o compõem. Esta implementação será feita com auxílio da biblioteca glm e sua execução ocorrerá nos shaders do OpenGL.

Preparação do ambiente

Antes de tudo, precisaremos fazer a instalação das bibliotecas **glm** e **GLEW** disponibilizadas pelo professor. O processo de instalação é mostrado nas figuras abaixo.

```
$ git clone --recurse-submodules https://github.com/capagot/icg.git
```

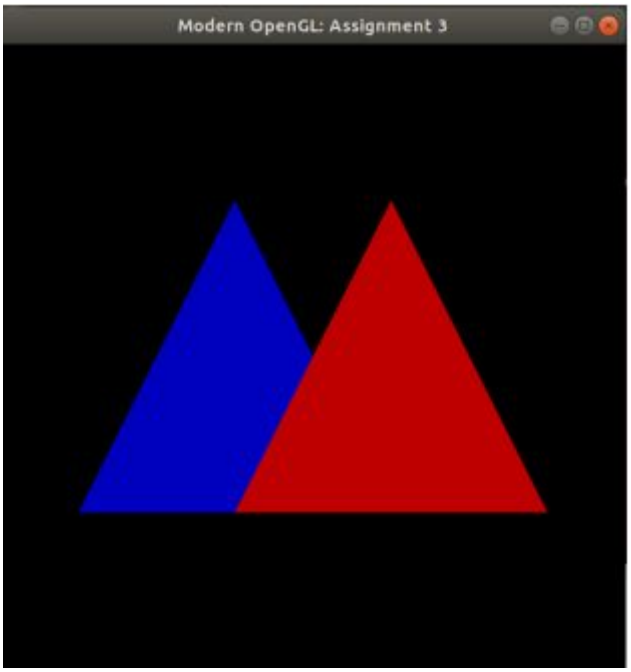
```
$ git clone https://github.com/nigels-com/glew.git glew
```

Logo após a instalação das bibliotecas, utilizamos os comandos ***make*** *e* ***./transform_gl*** *para compilar e executar o programa.*

Assim como na primeira atividade prática, não obtivemos êxito na compilação por vários erros relacionados a versão usada pelo *shader,* então utilizamos a mesma linha de comando mostrada na figura abaixo utilizada para resolver o mesmo problema da atividade 1 e funcionou.

```
$ MESA_GL_VERSION_OVERRIDE=3.3 MESA_GLSL_VERSION_OVERRIDE=330 ./transform_gl
```

Então, depois de resolver este problema, conseguimos êxito na execução. Veja na figura abaixo.



É importante destacar que a fim de facilitar nosso trabalho, adicionamos a biblioteca *cmath* para realizar operações matemáticas mais complexas, permitindo a criação das funções *CrossP* que é responsável por calcular o produto vetorial entre dois vetores de 3 dimensões, função *norm* que é responsável por calcular a norma de um vetor de 3 dimensões.

Matrizes

De acordo com as recomendações especificadas no roteiro deste exercício três matrizes serão manipuladas, sendo elas, Model, View e Projection.

Matriz Model: É a matriz responsável por determinar as transformações geométricas de cada cena. Essas transformações podem ser determinadas pelo procedimento que é multiplicar a transformação de escala pela translação, essa multiplicação é feita pelo trecho de código abaixo:

```
model_mat = scale_mat * translation_mat;
```

Os parâmetros de ambas as transformações têm seu valor padrão definido de forma a gerar como resultado a matriz identidade, assim fazendo com que a transformação que não esteja sendo aplicada não afete o resultado final.

Escala: Para a escala no espaço 3D são necessários 3 parâmetros, um para cada dimensão, chamados sx, sy e sz. A partir destes a matriz será:

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translação: Utilizamos três parâmetros: dx, dy e dz. Esses parâmetros representam o deslocamento em cada eixo relativo à origem do sistema de coordenadas. A matriz que representa esta operação é dada por:

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

O valor padrão dos três parâmetros é 0.

Matriz View: É responsável por mostrar a perspectiva da visão da câmera na cena. Essa matriz pode ser encontrada quando você multiplica a base ortonormal por uma matriz que representa a translação da posição da câmera. Para este fim, vamos precisar de três parâmetros assim como foi determinado no roteiros pelo professor que são a posição da câmera, o ponto para o qual ela aponta, e seu vetor up A partir desses devemos calcular o vetor direção da câmera, obtida subtraindo a posição da câmera do ponto para o qual ela aponta. Em seguida, utilizamos o vetor direção calculado e o vetor up definido

para gerar uma base ortonormal para a câmera, utilizando as fórmulas a seguir.

$$z_{cam} = -\frac{d_{cam}}{\|d_{cam}\|} \qquad x_{cam} = \frac{u \times z_{cam}}{|u \times z_{cam}|}$$

,

$$y_{cam} = z_{cam} \times x_{cam}$$

A partir dessas fórmulas, temos a matriz que representa a base ortonormal:

$$\begin{bmatrix} x_{cam(i)} & x_{cam(j)} & x_{cam(k)} & 0 \\ y_{cam(i)} & y_{cam(j)} & y_{cam(k)} & 0 \\ z_{cam(i)} & z_{cam(j)} & z_{cam(k)} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e a matriz de transação é dada por

$$\begin{bmatrix} 1 & 0 & 0 & -p_i \\ 0 & 1 & 0 & -p_j \\ 0 & 0 & 1 & -p_k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

onde, P= (pi, pj e pk) representa a posição da câmera.

Depois de calcular isso tudo, utilizamos a biblioteca **glm** para desenvolver a matriz view através do produto entre a matriz que representa a base ortonormal e a matriz de translação. Por fim, o ponto posição da câmera, o ponto para o qual ela olha e o vetor up podem ser representados através dos seguintes pontos: P = (0, 0, 0), D = (0, 0, -1) e *u = (0)î + (1)j + (0)k.*

Matriz Projection: é construída utilizando o parâmetro ‘d’, que é a distância do centro de projeção ao centro do sistema. A matriz projection será diferente de acordo com o valor do parâmetro ‘d’, seu tratamento irá depender se o valor de ‘d’ será igual a um ou a zero. Para este fim, utilizamos um condicional que checa o valor do parâmetro ‘d’, caso ‘d’ seja igual a zero, utilizamos o array definido pelo professor para gerar a matriz identidade utilizando a biblioteca **glm**. Caso contrário, utilizaremos da biblioteca e do array e aplicaremos algumas mudanças para que a matriz identidade seja igual a:

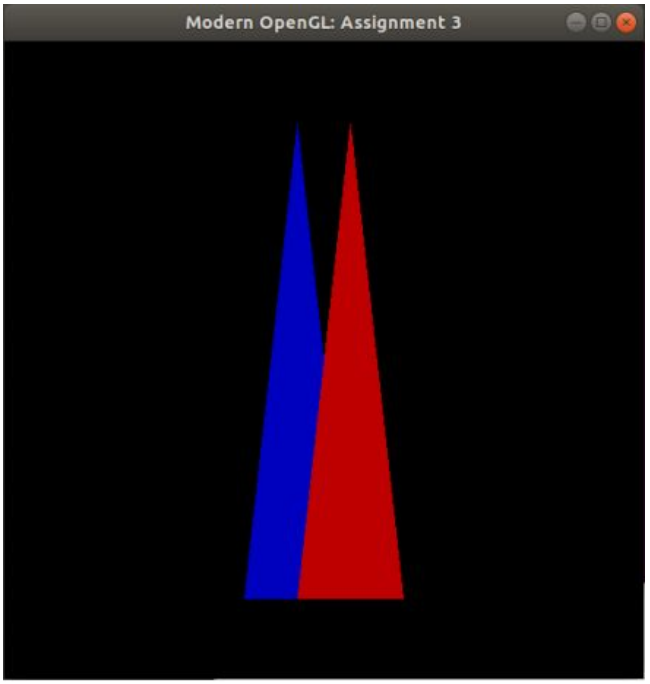
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & -1/d & 0 \end{bmatrix}$$

Exercícios

É importante ressaltar que em todos os exercícios a seguir, as imagens serão geradas aplicando os parâmetros dados pelo professor sobre específicas matrizes.

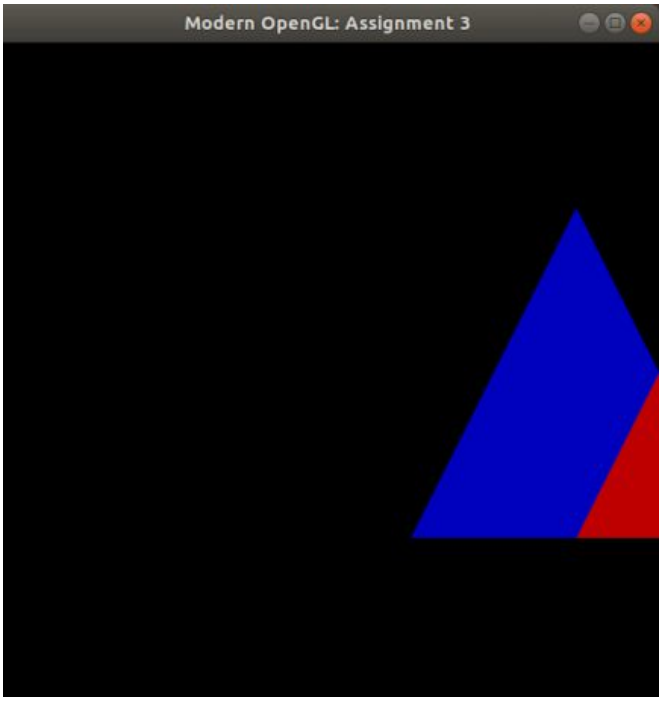
Escala

Neste exercício aplicamos os parâmetros sx = 1/3, sy = 3/2 e sz = 1, dados pelo professor, na matriz view. A figura abaixo mostra o resultado da execução.



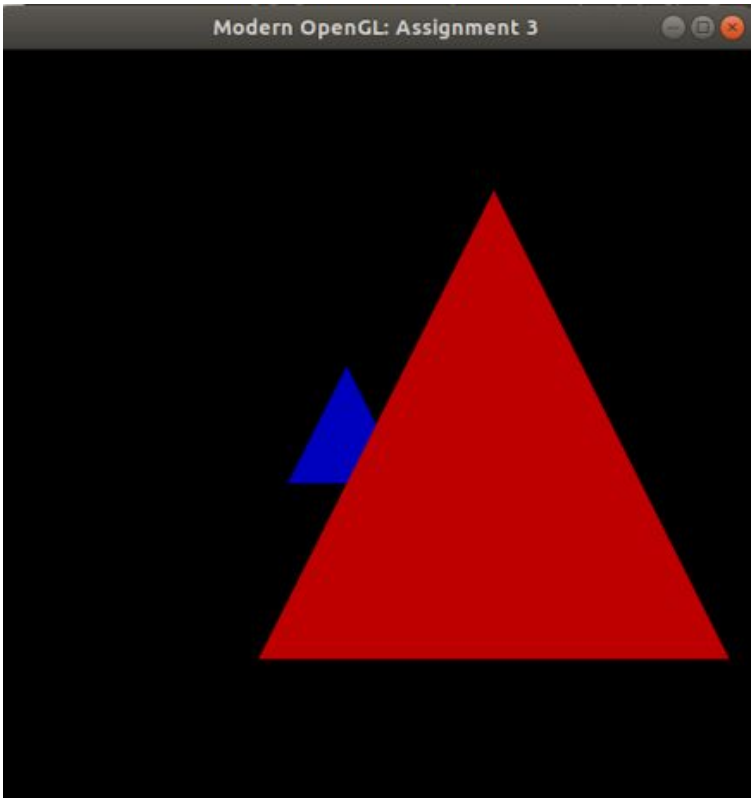
Translação

Neste exercício, faremos o mesmo procedimento do exercício anterior, onde aplicamos os parâmetros dx = 1, dy = 0 e dz = 0 definidos pelo professor na matriz view. A figura abaixo mostra o resultado da execução.



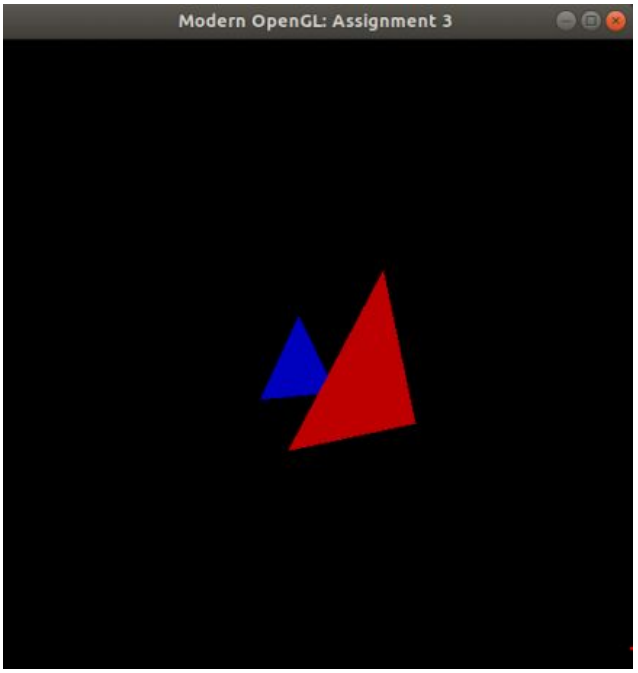
Projeção Perspectiva

Para este exercício, em vez de usar a matriz view, utilizaremos a matriz projection, usando o parâmetro d = ⅛, definido pelo professor. A figura abaixo mostra o resultado da execução.



Posição da câmera

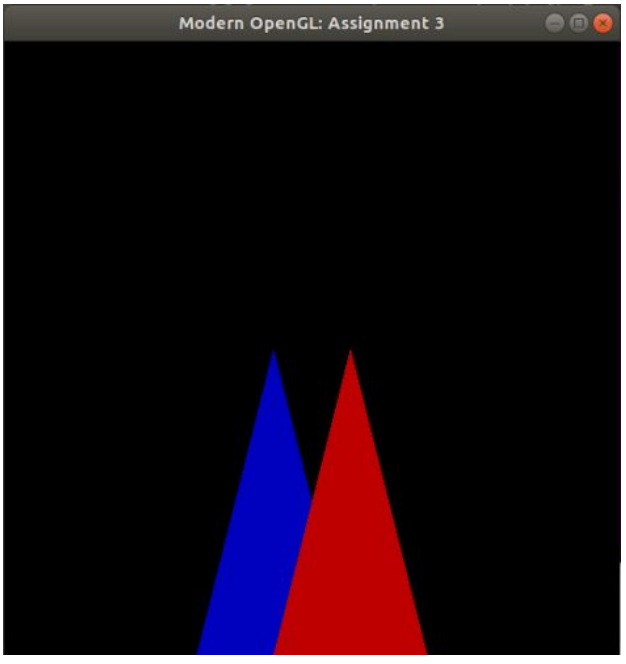
Neste exercício, que é um pouco mais complexo, utilizaremos os mesmos parâmetros usados no exercício anterior para a matriz projection. Enquanto isso, os parâmetros da matriz view, serão os parâmetros dados pelo professor que são P = (-1/10, 1/10, 1/10), u = (0)î + (1)ĵ + (0)k̂ e D = (0, 0, -1), onde P, u e D são o ponto posição da câmera, o vetor up da câmera, e o ponto para o qual a câmera aponta, respectivamente. A figura abaixo mostra o resultado da execução.



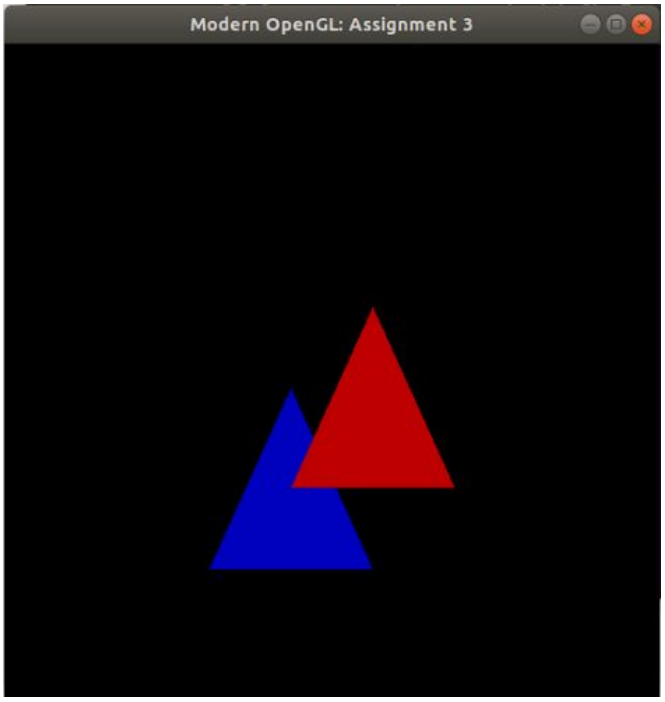
Transformações livres

Neste exercício o objetivo é modificar as matrizes model, view e projection de forma a gerar diferentes cenas. A seguir foram feitas duas transformações livres no âmbito de:

Escala e translação, com os parâmetros sx = 1/2 dy = -½ como mostra a figura abaixo:



Projeção de Câmera, com os parâmetros P = (0, -1/2, 0) D = (0, 1, -1) , como mostra a figura abaixo:



Referências: *Slides da disciplina, vídeos no youtube e material de sala.*