UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
INTRODUÇÃO À COMPUTAÇÃO GRÁFICA
ALUNOS: GABRIEL PONCIANO DE MIRANDA - 20190094837
ANTONI VINICIUS DA SILVA SOARES – 20190031597
DATA: 09/12/2020

Atividade 3

Esta atividade teve como objetivo praticar a implementação da teoria sobre *pipeline* gráfico e transformações geométricas em código, manipulando matrizes de transformação para obter um determinado resultado. Para este exercício, foi necessária a instalação da biblioteca **glm**.

Após clonar o repositório onde estão os arquivos necessários, a compilação foi executada a partir do comando **make** sem maiores problemas. A partir da execução do programa, foi possível verificar se as alterações feitas nas matrizes M_{model}, M_{view} e $M_{projection}$ foram as corretas para a consolidação dos exercícios. É importante ressaltar que para uma correta resolução das questões (devido à implementação do algoritmo), no código as colunas foram tidas como linhas e vice versa.

Para a resolução do exercício 1, foi necessário modificar M_{model} para que a imagem coincidisse com a do enunciado. A transformação foi feita com a multiplicação da matriz identidade com a matriz de escala, como feito a seguir:

$$I=\begin{vmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{vmatrix},\ E=\begin{vmatrix}E_x & 0 & 0 & 0\\ 0 & E_y & 0 & 0\\ 0 & 0 & E_z & 0\\ 0 & 0 & 0 & 1\end{vmatrix}$$

$$M_{model}=I*E=\begin{vmatrix}0.33 & 0 & 0 & 0\\ 0 & 1.5 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{vmatrix}$$

E_x, E_y e E_z foram substituídos pelos fatores de escala exibidos na legenda da figura no enunciado do exercício 1. Desta forma: $E_x = 0.33$, $E_y = 1.5$ e $E_z = 1$. Após o cálculo, os valores foram alterados no código e o programa foi compilado e executado, resultando na imagem esperada, como mostram as Figuras 1 e 2.

```
float model_array[16] = {0.33f, 0.0f, 0.0f, 0.0f,
                         0.0f, 1.5f, 0.0f, 0.0f,
                         0.0f, 0.0f, 1.0f, 0.0f,
                         0.0f, 0.0f, 0.0f, 1.0f};
glm::mat4 model_mat = glm::make_mat4(model_array);
```

Figura 1 - Alteração do float model_array[16] para a elucidação do Exercício 1.

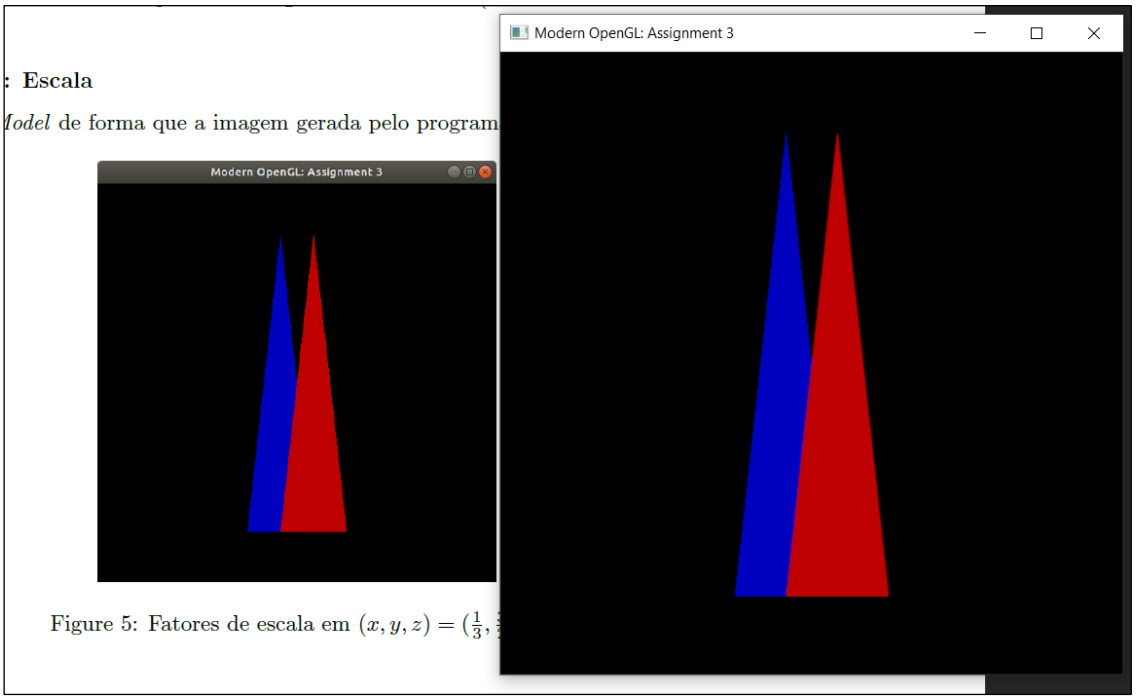

Figura 2 - Comparação entre a imagem desejada e a obtida.

O exercício 2 teve resolução semelhante ao do primeiro. Desta vez, ao invés de uma transformação de escala, é aplicada uma transformação de translação na matriz de modelagem. O cálculo de matrizes segue abaixo:

$$I = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}, \quad T = \begin{vmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$M_{model} = I * T = \begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Assim como na primeira tarefa, os coeficientes de translação foram obtidos da legenda da figura. Neste caso, os valores foram, 1, 0 e 0 para os valores de T_x, T_y e T_z, respectivamente. As Figuras 3 e 4 demonstram o código e a comparação entre a imagem pedida e a obtida.

```
float model_array[16] = {1.0f, 0.0f, 0.0f, 0.0f,
                         0.0f, 1.0f, 0.0f, 0.0f,
                         0.0f, 0.0f, 1.0f, 0.0f,
                         1.0f, 0.0f, 0.0f, 1.0f};
glm::mat4 model_mat = glm::make_mat4(model_array);
```

Figura 3 - Alterações na variável da matriz de modelagem para o Exercício 2.

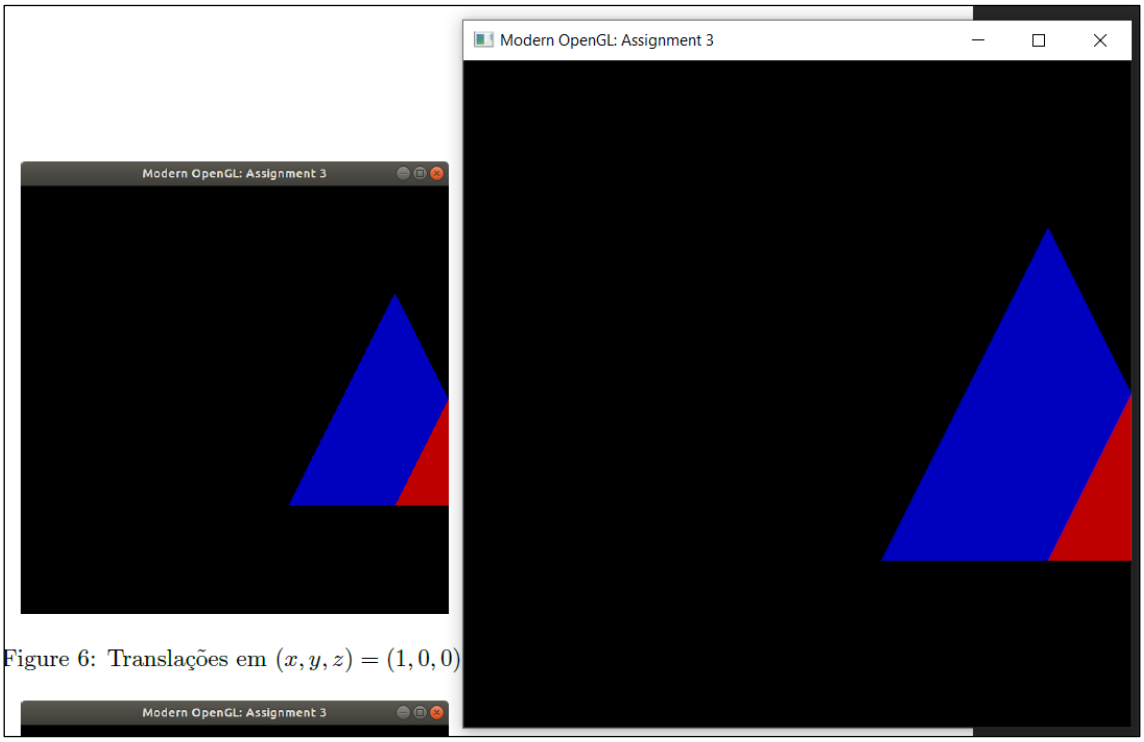

Figura 4 - Comparação entre a imagem desejada e a imagem obtida para o Exercício 2.

O exercício 3 tinha como enunciado alterar a matriz de projeção. Esta matriz foi dada em aula e no próprio enunciado.

$$M_{projection} = \begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & -1/d & 0 \end{vmatrix}$$

O valor de d foi dado na legenda e é $\frac{1}{8}$, que é a distância do centro de projeção até a origem do sistema de coordenadas. Substituindo na matriz temos:

$$M_{projection} = \begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.125 \\ 0 & 0 & -8 & 0 \end{vmatrix}$$

Isso resultou em uma alteração no código como mostrado na Figura 5. E a imagem produzida pelo programa coincidiu com a esperada (Figura 6).

```
float proj_array[16] = {1.0f, 0.0f, 0.0f, 0.0f,
                        0.0f, 1.0f, 0.0f, 0.0f,
                        0.0f, 0.0f, 1.0f,-8.0f,
                        0.0f, 0.0f, 0.125f, 0.0f};
```

Figura 5 - Matriz alterada no código do algoritmo.

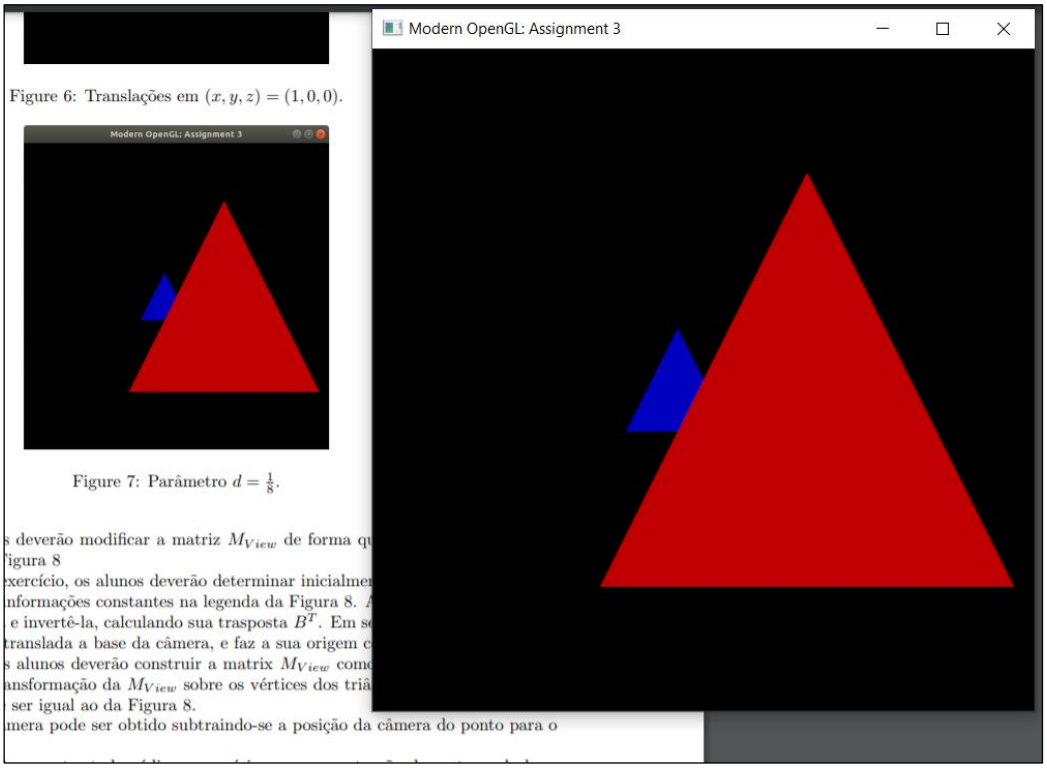

Figura 6 - Comparação entre a imagem pedida e a obtida para o Exercício 3.

Para o quarto problema, a matriz a ser modificada é a M_{view} e a perspectiva utilizada foi a do exercício anterior. Os parâmetros para a realização deste quesito foram dados na legenda da figura correspondente. Mas, para a devida alteração da matriz, é preciso calcular alguns parâmetros:

- O vetor direção da câmera (d):

$$d = p - a$$

$$d=(-\frac{1}{10},\frac{1}{10},\frac{1}{10}) - (0,0,-1) = (-\frac{1}{10},\frac{1}{10},\frac{11}{10})$$

- A matriz transposta da base da câmera (B^T):

$$B^T = \begin{vmatrix} X_{cx} & X_{cy} & X_{cz} & 0 \\ Y_{cx} & Y_{cy} & Y_{cz} & 0 \\ Z_{cx} & Z_{cy} & Z_{cz} & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

- Os coeficientes a serem inseridos na matriz B^T:

$$Z_c = \frac{d}{|d|}$$

$$Z_c = \frac{(-\frac{1}{10},\frac{1}{10},\frac{11}{10})}{\sqrt{(\frac{-1}{10})^2+(\frac{1}{10})^2+(\frac{11}{10})^2}}$$

$$Z_c = \frac{(-\frac{1}{10},\frac{1}{10},\frac{11}{10})}{\sqrt{\frac{123}{100}}} = (-0.09,\ 0.09,\ 0.99)$$

$$Z_c = (-0.1,\ 0.1,\ 1)$$

$$X_c = \frac{U \times Z_c}{|U \times Z_c|}$$

$$X_c = \frac{(0,\ 1,\ 0) \times (-0.1,\ 0.1,\ 1)}{\sqrt{|(0,1,0)\times(-0.1,0.1,1)|}}$$

$$X_c = \frac{(1,\ 0,\ 0.1)}{\sqrt{1}}$$

$$X_c = (1,\ 0,\ 0.1)$$

$$Y_c = \frac{Z_c \times X_c}{|Z_c \times X_c|}$$

$$Y_c = \frac{(-0.1,\ 0.1,\ 1)\times(1,\ 0,\ 0.1)}{\sqrt{|(-0.1,0.1,1)\times(1,0,0.1)|}}$$

$$Y_c = \frac{(0.01,\ 1.01, -0.01)}{\sqrt{1}}$$

$$Y_c = (0.01,\ 1.01, -0.01)$$

A matriz de visualização é obtida ao realizar a multiplicação entre a matriz B^T e a matriz de translação já conhecida para essa transformação.

$$B^T = \begin{vmatrix} 1 & 0 & 0.1 & 0 \\ 0.01 & 1.01 & -0.01 & 0 \\ -0.1 & 0.1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}, \quad T = \begin{vmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & -0.1 \\ 0 & 0 & 1 & -0.1 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$M_{view} = B^T * T = \begin{vmatrix} 1 & 0 & 0.1 & 0.1 \\ 0.01 & 1.01 & -0.1 & -0.1 \\ -0.1 & 0.1 & 1 & -0.12 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

As variáveis são finalmente alteradas no código e o programa é executado e compilado, registrando a imagem desejada (Figuras 7 e 8).

```
float view_array[16] = {1.0f, 0.01f,-0.1f, 0.0f,
                        0.0f, 1.01f, 0.1f, 0.0f,
                        0.1f, -0.1f, 1.0f, 0.0f,
                        0.1f, -0.1f,-0.12f, 1.0f};
```

Figura 7 - Alteração na Mview no código.

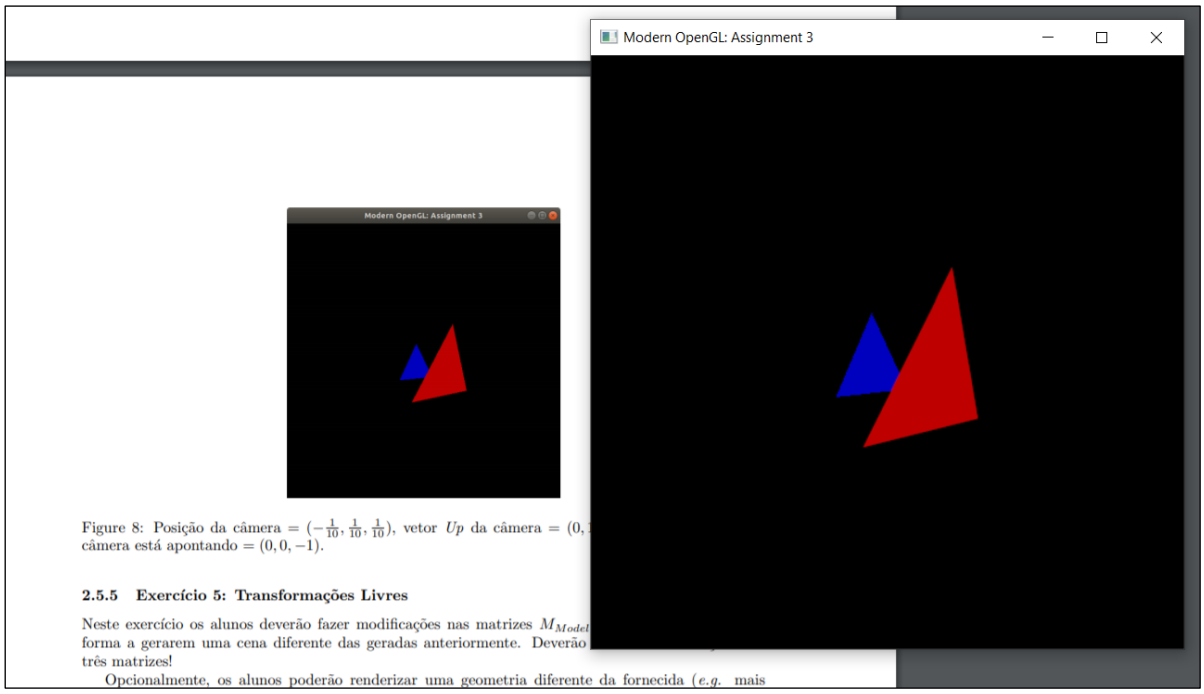

Figura 8 - Imagem requerida em comparação com a imagem obtida para o Exercício 4.

O quinto e último exercício tinha como objetivo simplesmente criar uma nova cena, distinta das demais já registradas com alterações nas três matrizes: M_{model}, M_{view} e $M_{projection}$.

As alterações nas três matrizes e a nova cena são mostradas nas Figuras 9, 10 e 11.

```
float model_array[16] = {0.75f,  0.0f, 0.0f, 0.0f,
                         0.0f, 0.37f, 0.0f, 0.0f,
                         0.0f,  0.0f, 0.25f, 0.0f,
                         0.0f,  0.0f, 0.0f, 1.0f};
```

Figura 9 - Matriz de modelagem para a nova cena.

```
float proj_array[16] = {1.0f, 0.0f, 0.0f, 0.0f,
                        0.0f, 1.0f, 0.0f, 0.0f,
                        0.0f, 0.0f, 1.0f,-2.0f,
                        0.0f, 0.0f, 0.5f, 0.0f};
```

Figura 10 - Matriz de projeção para a nova cena.

```
float view_array[16] = {1.5f, 0.55f,0.25f, 0.0f,
                        0.5f, 1.00f,0.50f, 0.0f,
                        0.25f,0.75f,-1.5f, 0.0f,
                        0.15f,0.35f,-0.75f,1.0f};
```

Figura 11 - Matriz de visualização para a nova cena.

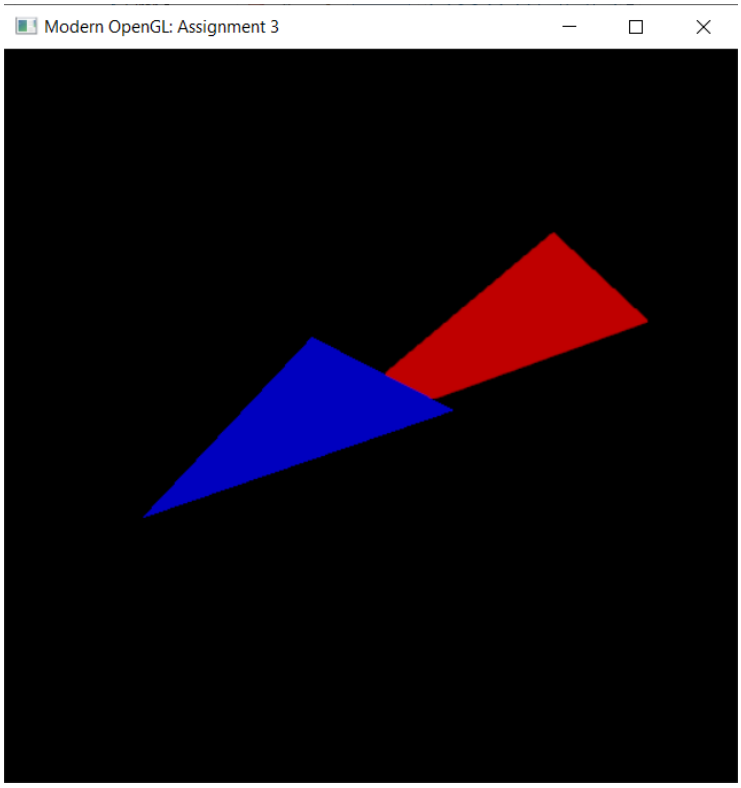

Figura 12 - A nova cena.

Por fim, foi possível se familiarizar com o ambiente de trabalho com o *pipeline* gráfico utilizando a biblioteca **glm** e o *OpenGL*.

REFERÊNCIAS:

- Video-aulas disponibilizadas pelo professor Christian Pagot sobre *pipiline* gráfico.