UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

CARLOS MAGNO DA SILVA – Matrícula: 20160143331

INTRODUÇÃO A COMPUTAÇÃO GRÁFICA

Atividade Prática 3

Implementação do Pipeline Gráfico

JOÃO PESSOA
2020

UFPB – Universidade Federal da Paraíba
Centro de Informática – CI

Disciplina: Introdução a Computação Gráfica – ICG
Professor: Christian Azambuja Pagot

Aluno: Carlos Magno da Silva – **Matrícula:** 20160143331
Curso: Ciência da Computação

Semestre: 2019.4

Atividade Prática 3
Implementação do Pipeline Gráfico

1. INTRODUÇÃO

A presente atividade foi solicitada pelo professor da disciplina e consiste na terceira atividade prática da disciplina de ICG, no presente período, tendo como finalidade familiarizar os alunos com a estrutura do pipeline gráfico através da implementação das transformações geométricas que o compõem. Esta implementação foi feita com o auxílio da **biblioteca glm** e sua execução ocorrerá nos **shaders** do **OpenGL**.

2. ATIVIDADE

A atividade solicitada consistiu em fazer download, do código **template C++17** disponibilizado pelo professor e disponível no repositório da disciplina no site: **https://github.com/capagot/icg/tree/master/03_transformations**, e a partir deste implementar a atividade solicitada pelo professor.

3. DESENVOLVIMENTO

Nesta atividade foi utilizada a **IDE CodeBlocks**, utilizando a **Linguagem C/C++** com as **bibliotecas GLUT** e o **OpenGL**, previamente instalados na **plataforma Windows 10 – 64 bits**. Além dessas bibliotecas foram utilizadas e devidamente instaladas e configuradas na **IDE CodeBlocks,** as bibliotecas **glm** e **GLEW**, conforme solicitação e orientações nos requisitos do presente trabalho

De posse dos arquivos disponibilizados, baixados do link supracitado, foi feita a importação dos arquivos para a **IDE CodeBlocks**, com as devidas configurações necessárias. A partir daí se fez necessário fazer alterações no programa template, especificamente no conteúdo das matrizes $\mathbf{M_{Model}}$, $\mathbf{M_{View}}$ e $\mathbf{M_{Projection}}$ visando a geração das imagens presentes nos requisitos da presente atividade. Antes da realização de cada atividade foi verificado se as matrizes supracitadas continham a matriz identidade, exceto no exercício 4.

Seguem abaixo, as imagens de alguns trechos de códigos, referentes às funções presentes no projeto.

3.1. Codificação da Declaração dos Triângulos

```
52
53  // ********************************************************************* //
54  // Array contendo as coordenadas X,Y e Z de 03(tres) vertices (um triangulo) //
55  // ********************************************************************* //
56  float vertices[] = {
57
58                      -0.25f, -0.5f, -0.1f, 0.75f, 0.0f, 0.0f, // Triangulo vermelho(mais proximo)
59                       0.25f,  0.5f, -0.1f, 0.75f, 0.0f, 0.0f,
60                       0.75f, -0.5f, -0.1f, 0.75f, 0.0f, 0.0f,
61
62                      -0.75f, -0.5f, -0.4f, 0.0f, 0.0f, 0.75f, // Triangulo azul(mais longe)
63                      -0.25f,  0.5f, -0.4f, 0.0f, 0.0f, 0.75f,
64                       0.25f, -0.5f, -0.4f, 0.0f, 0.0f, 0.75f
65
66                    };
67
```

Figura 1 – Codificação da declaração dos Triângulos

3.2. Codificação da Função de Loader de Leitura dos Shader's

```c
void LoadShader(char* file_name, char** shader_source) {

    long length;
    FILE* f = fopen(file_name, "rb");

    if (f) {

        fseek(f, 0, SEEK_END);
        length = ftell(f);
        fseek(f, 0, SEEK_SET);

        (*shader_source) = (char*)malloc(length + 1);
        if ((*shader_source)) {

            fread((*shader_source), 1, length, f);
            (*shader_source)[length] = '\0';

        }

        fclose(f);

    } else {

           printf("Nao foi possivel carregar o arquivo de shader(sombreador)%s\nSaindo ...", file_name);

           exit(EXIT_FAILURE);

    }
}
```

Figura 2 – Codificação da Função de Leitura dos Shader's

3.3. Codificação da Matriz Model

```c
// ******************************************* //
// Seleciona o Shader Program a ser utilizado //
// ******************************************* //
glUseProgram(shader_program);

// ************ //
// Matriz Model //
// ************ //
// ***************************************************************** //
// Voce precisara alterar o conteudo dessa matriz para os exercicios //
// ***************************************************************** //
float model_array[16] = {
                            1.0f,  0.0f, 0.0f, 0.0f, // Exercicio ??
                            0.0f, -1.0f, 0.0f, 0.0f,
                            0.0f,  0.0f, 1.0f, 0.0f,
                            0.0f, -0.5f, 0.0f, 1.0f

                        };

glm::mat4 model_mat = glm::make_mat4(model_array);


// ************ //
```

Figura 3 – Exemplo de Codificação da Matriz Model com a bilblioteca GLM

3.4. Codificação da Matriz View

```
941     // ************* //
942     // Matriz View  //
943     // ************* //
944     // ********************************************************************** //
945     // Voce precisara alterar o conteudo dessa matriz para os exercicios //
946     // ********************************************************************** //
947     float view_array[16] = {
948
949                                1.0f, 0.0f,  0.0f, 0.0f,
950                                0.0f, 1.0f,  0.0f, 0.0f,
951                                0.0f, 0.0f,  1.0f, 0.0f,
952                               -1.0f, 0.0f, -1.0f, 1.0f
953
954                             };
955
956
957     glm::mat4 view_mat = glm::make_mat4(view_array);
958
959
960     // ****************** //
```

Figura 4 – Exemplo de Codificação da Matriz View com a bilblioteca GLM

3.5. Codificação da Matriz Projection

```
959
960     // ****************** //
961     // Matriz Projection //
962     // ****************** //
963     // ********************************************************************** //
964     // Voce precisara alterar o conteudo dessa matriz para os exercicios //
965     // ********************************************************************** //
966     float proj_array[16] =  {
967
968                             1.0f, 0.0f,  0.0f, 0.0f,
969                             0.0f, 1.0f,  0.0f, 0.0f,
970                             0.0f, 0.0f,  1.0f, -3.0f,
971                             0.50f, 0.50f, 0.50f, 1.0f
972
973                           };
974
975     glm::mat4 proj_mat = glm::make_mat4(proj_array);
976
977
```

Figura 5 – Exemplo de Codificação da Matriz Projection com a bilblioteca GLM

4. TELA DE RESULTADO DO PROCESSAMENTO

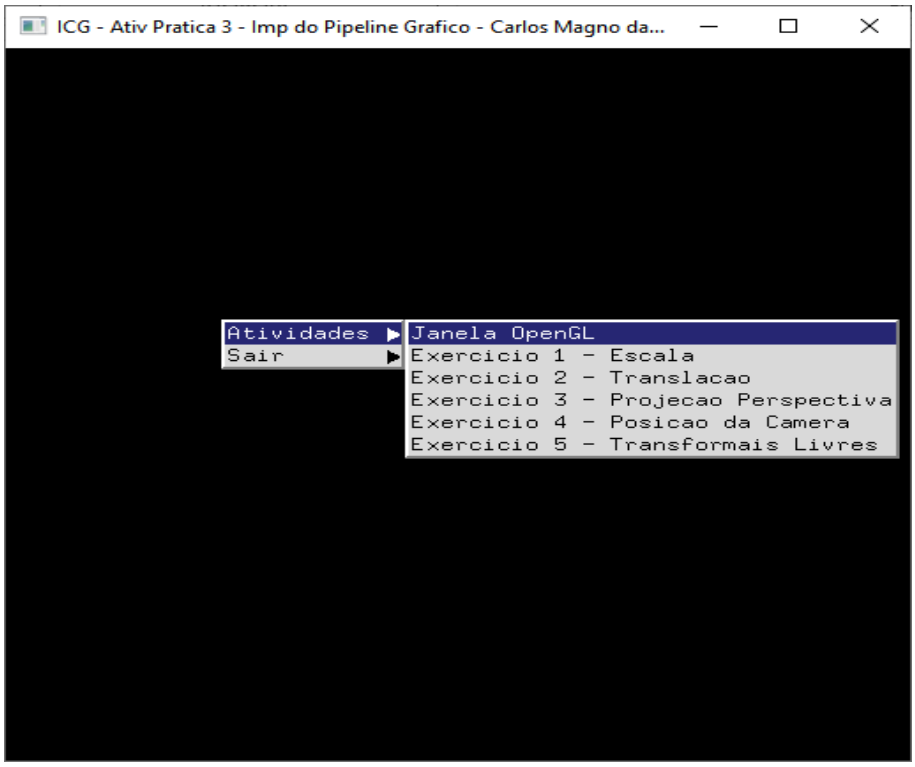

Figura 6 – Janela Menu Principal de Opções

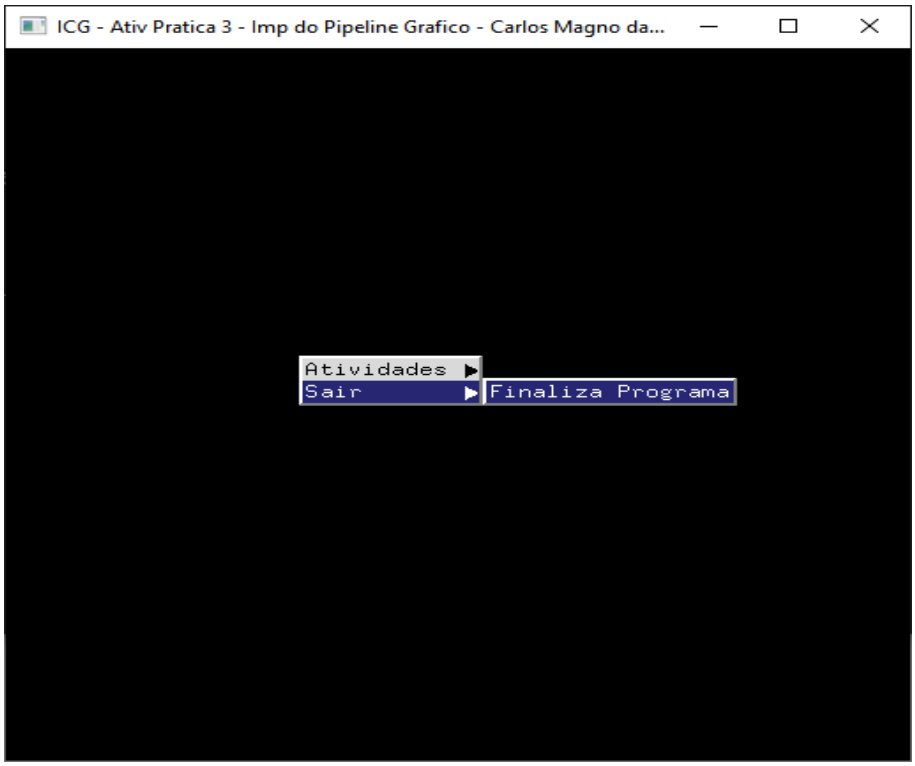

Figura 7 – Janela de saída do programa

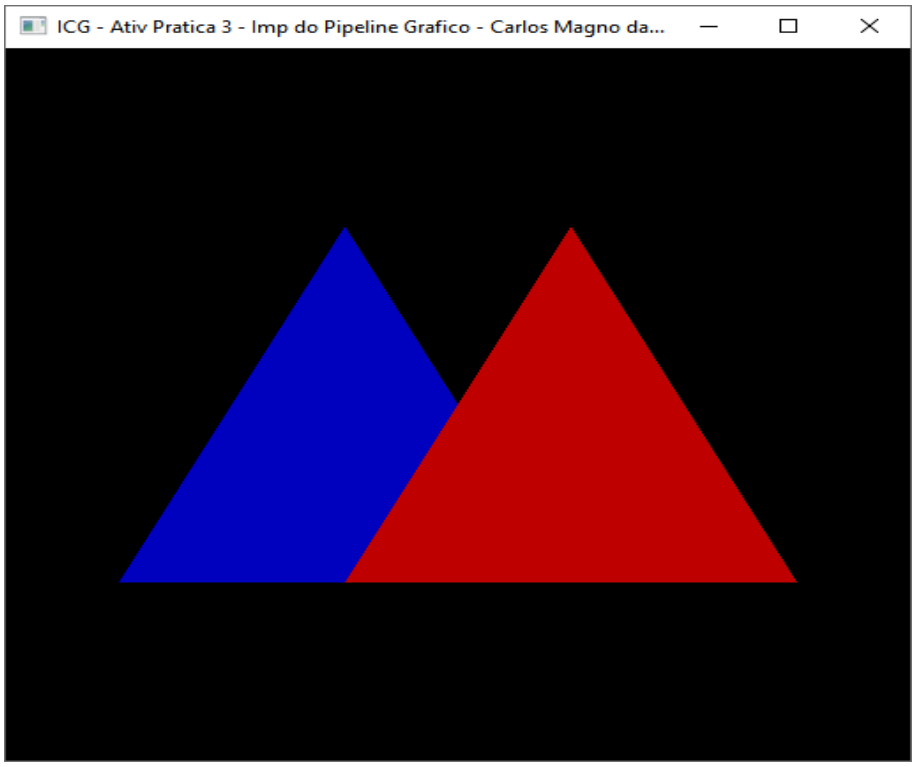

Figura 8 – Janela OpenGL criada durante a execução do programa – Exercício 1

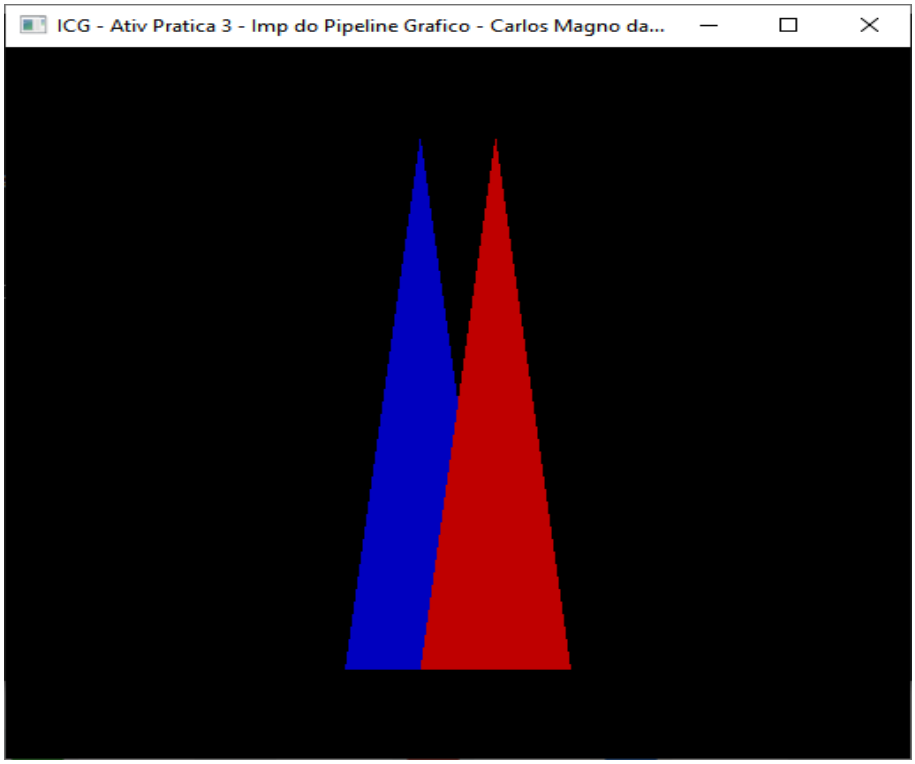

Figura 9 – Janela Fatores de Escala em (x, y, z) = (1/3, 3/2, 1) – Exercício 2

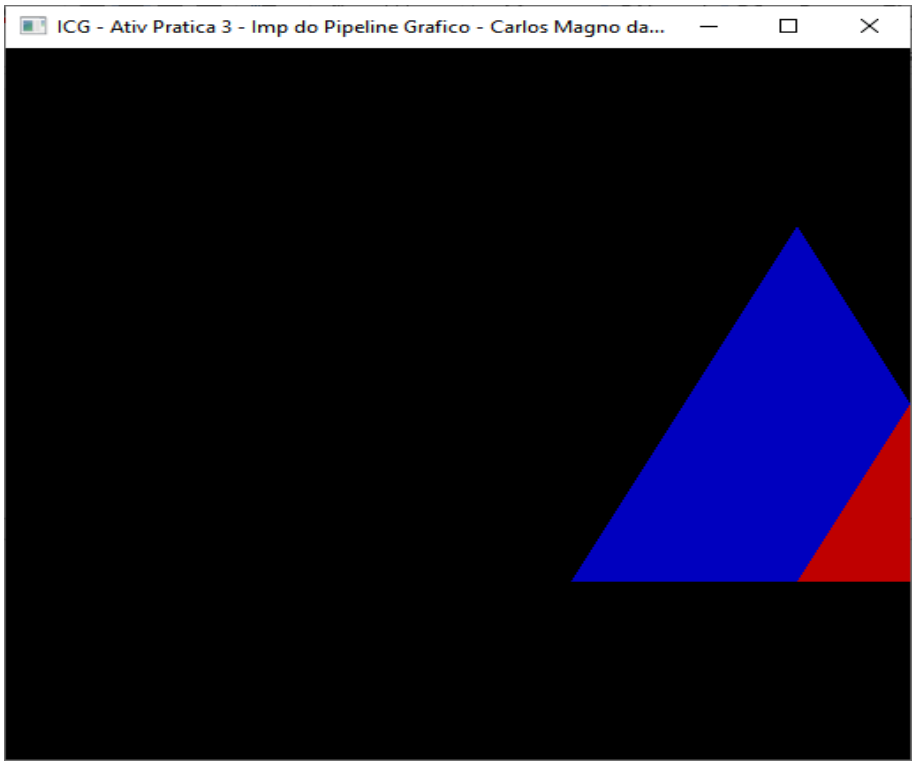

Figura 10 – Janela Translações em (x, y, z) = (1, 0, 0) – Exercício 3

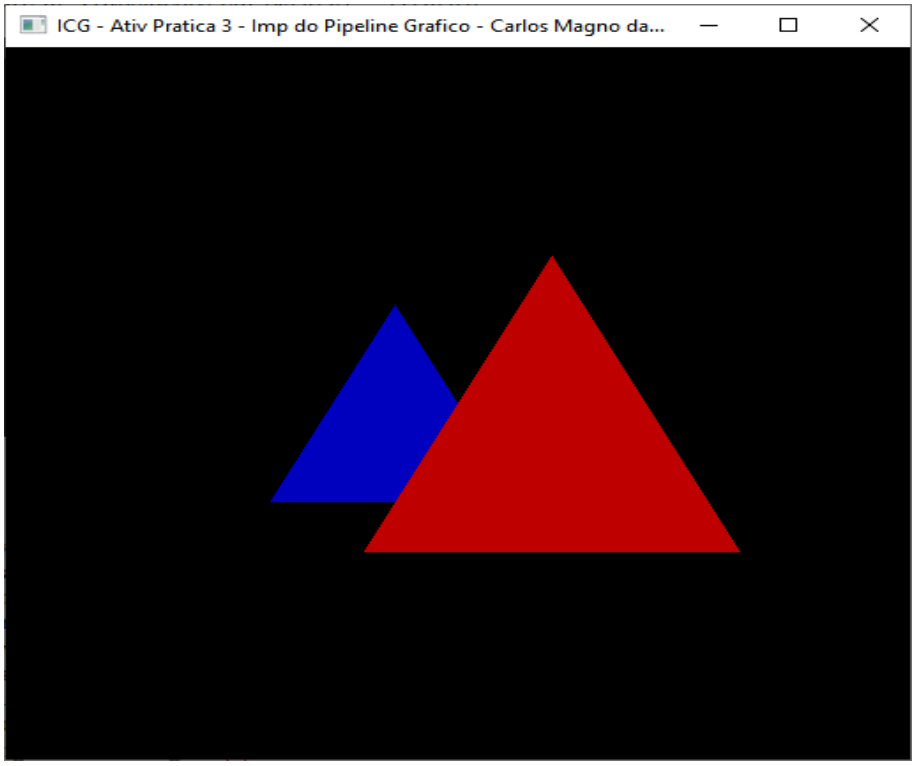

Figura 11 – Janela Parâmetro d = ½ – Exercício 4

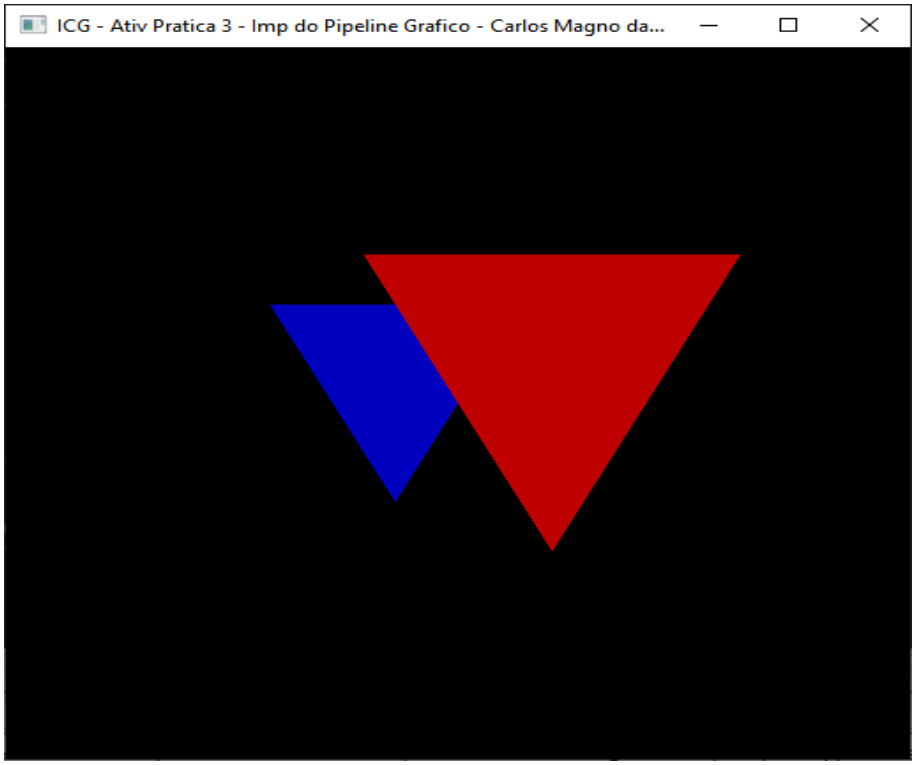

Figura 12 – Janela com Modificação da Matriz Model /View/Projection – Exercício 5

5. CONCLUSÃO

Conforme solicitado nesta **Atividade Prática 3**, o objetivo foi alcançado. Isso pode ser observado nas imagens das telas de saídas (janelas), após a execução do programa.

Principais Dificuldades

Algumas dificuldades surgiram durante a instalação e configuração das bibliotecas do **glm e GLEW** na **IDE do CodeBlocks**, que foram utilizadas neste projeto;

Possíveis Melhorias

Verificou-se que uma possível melhoria no projeto, poderia ser feita tornado o código mais modularizado, dividindo-os em módulos de funções, arquivos header's, classes, etc. Com isso, as implementações realizadas poderão ser reaproveitadas em outros projetos, se necessário, ou seja, atendendo às boas práticas de programação.

5. Referências Bibliográficas

1 – Tutorial de Utilização de OpenGL – Marcionílio Barbosa Sobrinho – Belo Horizonte – MG – 2003.

2 – Introdução à OpenGL - Professora Isabel Harb Manssour – **Link: https://www.inf.pucrs.br/~manssour/OpenGL/Desenhando.html**
Acesso em: 25/07/2020.