Atividade prática de INF 390 – Computação Gráfica

Thiago Luange Gomes

Objetivo:

- Estudar o programa apresentado em aula e adicionar funcionalidades ao programa. O programa está disponível no moodle com nome “SCENE-3D-0”.

Funcionalidades a serem adicionadas:

- Adicionar na classe scene as matrizes necessárias para realizar uma visualização 3D:

  glm::mat4 Projection_matrix;

  glm::mat4 View_matrix;

- Inicializar corretamente as matrizes, transmitir as matrizes para o shader e modificar o shader para receber e usar as matrizes:

```
#version 130

in vec3 Position;
uniform vec3 Color;
uniform mat4 Projection;
uniform mat4 Model;
uniform mat4 View;
out vec4 vs_Color;
void main()
{
        gl_Position = Projection*View*Model*vec4(Position, 1.0);
        vs_Color = vec4(Color.x,Color.y, Color.z, 1.0);

}
```

- Atualizar o método Ortho2D(float WL,float WR,float WB, float WT) para o método Ortho3D(float WL,float WR,float WB, float WT,float zNear,float zFar). Esse método deve gerar uma matriz que será atribuída a Projection_matrix. Remover a Ortho2D_matrix. Dica use a glm para gerar a matriz necessária (http://glm.g-truc.net/0.9.5/glm-0.9.5.pdf)
- Adicionar um método para definir o posicionamento da câmera: LookAt(float eyex,float eyey,float eyez, float centerx,float centery,float centerz, float upx,float upy,float upz); Esse método deve gerar uma matriz que será atribuída a View_matrix. Dica use a glm para gerar a matriz necessária (http://glm.g-truc.net/0.9.5/glm-0.9.5.pdf)

- Adicionar um método para definir a visualização perspectiva: perspective(float fovy, float aspect, float zNear, float zFar); Esse método deve gerar uma matriz que será atribuída a Projection_matrix. Dica use a glm para gerar a matriz necessária (http://glm.g-truc.net/0.9.5/glm-0.9.5.pdf)
- Desenhar um cubo: Usando o construtor object(int v_number,int i_number, GLfloat Vertices[],GLushort index[]), construa 6 objetos, uma para cada face e adicione esses objetos a cena (push_back_object). Os objetos devem possuir cores diferentes.
- Defina uma posição da câmera usando a LookAt para visualizar o cubo.
- Definir uma variável global para selecionar o tipo de câmera e na função de teclado ajustar o valor dessa variável:

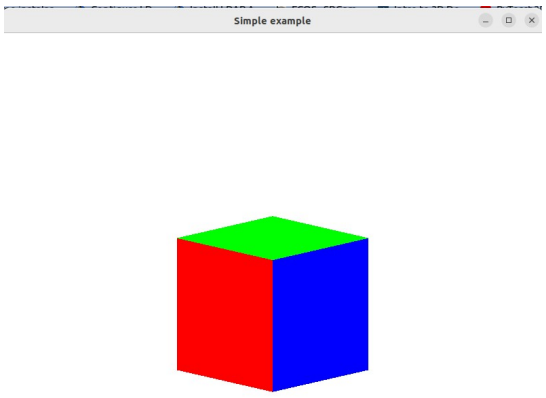
```
//global
bool ortho_per = false;
static void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
    {
        if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
            glfwSetWindowShouldClose(window, GLFW_TRUE);
        if ( key == GLFW_KEY_P && action == GLFW_PRESS )
            ortho_per = true;
        if ( key == GLFW_KEY_O && action == GLFW_PRESS )
            ortho_per = false;
    }
```

- Com base no valor da variável anterior, definir a câmera:

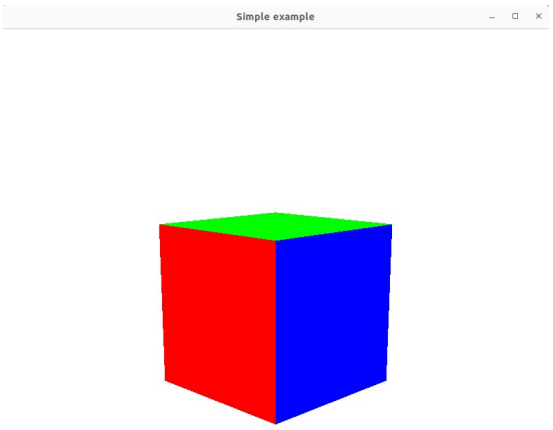
```
while (!glfwWindowShouldClose(window))
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    if (ortho_per){
        my_scene.perspective(0.75,1.0,0.1,10.0);
    }else{
        my_scene.Ortho3D(-2.0,2.0,-2.0,2.0,0.0,10.0);
    }
    my_scene.render();
    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

- Assim, o seu programa final dever mostrar um cubo na tela. Quando acionado a tecla ‘o’ o cubo deve ser mostrado com uma projeção ortográfica e quando acionado o ‘p’ o cubo deve ser mostrado com uma projeção perspectiva.

- Ortográfica:



- Perspectiva:



Instruções de envio:

- **Enviar uma captura de tela (print screen) do programa funcionando e os códigos modificados (pasta src).**