DA1MCTA008-17SA - Computação Gráfica - Bruno Augusto Dorta Marques - 2021.3

Atividade 2

Porta Aviões

Guilherme Assencio RA: 1120181063

Henrique Queiroz Reuter RA:11201812261

Implementação:

- Foi utilizado o `OpenGLWindow::handleEvent(SDL_Event& ev)` para controlar a posição da câmera, alterando a componente da velocidade horizontal ou vertical para +1 ou -1 dependendo do botão apertado. Ao soltar o botão a componente da velocidade alterada é setada para 0.
- Dentro do initializeGL é criado o m_program a partir do lookat.frag e lookat.vert fornecidos em aula.

```
float i = 1.0 - (-posEyeSpace.z / 50.0);
fragColor = vec4(i, i, i, 1) * color;
```

  Dividimos o posEyeSpace por 50 para aumentar o campo de visão, já que os modelos utilizados são grandes.
- Ainda dentro do initializeGL executamos os seguintes comandos:

```
m_ground.initializeGL(m_program);

m_rex.loadFromFile(getAssetsPath() + "REX_2.obj");
m_ship.loadFromFile(getAssetsPath() +"CV - Essex class/essex_scb-125_generic.obj");

m_rex.initializeGL(m_program);
m_ship.initializeGL(m_program);
```

- O ground foi implementado do mesmo modo demonstrado em aula. Alteramos a cor e altura que o ground é instanciado para simular o mar e não deixar o porta-aviões flutuando.
- A função loadFromFile foi implementada do mesmo modo demonstrado em aula.
- Dentro do initializeGL do m_rex e do m_ship, criamos e fazemos o bind dos EBOs e VAOs passando o m_program como parâmetro

```
abcg::glGenBuffers(1, &m_VBO);
abcg::glBindBuffer(GL_ARRAY_BUFFER, m_VBO);
abcg::glBufferData(GL_ARRAY_BUFFER, sizeof(m_vertices[0]) * m_vertices.s
                   m_vertices.data(), GL_STATIC_DRAW);
abcg::glBindBuffer(GL_ARRAY_BUFFER, 0);

// Generate EBO of Porta
abcg::glGenBuffers(1, &m_EBO);
abcg::glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_EBO);
abcg::glBufferData(GL_ELEMENT_ARRAY_BUFFER,
                   sizeof(m_indices[0]) * m_indices.size(), m_indices.da
                   GL_STATIC_DRAW);
abcg::glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);

//Processo de criação do VAO do Porta-aviões
// Create VAO
abcg::glGenVertexArrays(1, &m_VAO);

// Bind vertex attributes to current VAO
abcg::glBindVertexArray(m_VAO);

abcg::glBindBuffer(GL_ARRAY_BUFFER, m_VBO);
const GLint positionAttribute{
    abcg::glGetAttribLocation(m_program, "inPosition")};
abcg::glEnableVertexAttribArray(positionAttribute);
abcg::glVertexAttribPointer(positionAttribute, 3, GL_FLOAT, GL_FALSE,
                            sizeof(Vertex), nullptr);
abcg::glBindBuffer(GL_ARRAY_BUFFER, 0);

abcg::glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_EBO);
```

- Dentro do paintGL do openglwindow.cpp fazemos:

```
// Get location of uniform variables (could be precomputed)
const GLint viewMatrixLoc{
    abcg::glGetUniformLocation(m_program, "viewMatrix")};
const GLint projMatrixLoc{
    abcg::glGetUniformLocation(m_program, "projMatrix")};
const GLint modelMatrixLoc{
    abcg::glGetUniformLocation(m_program, "modelMatrix")};
const GLint colorLoc{abcg::glGetUniformLocation(m_program, "color")};

// Set uniform variables for viewMatrix and projMatrix
// These matrices are used for every scene object
abcg::glUniformMatrix4fv(viewMatrixLoc, 1, GL_FALSE,
                         &m_camera.m_viewMatrix[0][0]);
abcg::glUniformMatrix4fv(projMatrixLoc, 1, GL_FALSE,
                         &m_camera.m_projMatrix[0][0]);

//draw ship
m_ship.paintGL(modelMatrixLoc, colorLoc);

//draw rex
m_rex.paintGL(modelMatrixLoc, colorLoc);

// Draw ground
m_ground.paintGL();

abcg::glUseProgram(0);
```

  Desenhamos o m_ship, m_rex e m_ground com o paintGL respectivo de cada classe.

- Dentro de cada paintGL, fazemos o bind do VAO, transladamos, rotacionamos e definimos a escala de cada modelo desenhado, para depois desenhar.

```
 //Bind vertex attributes to current VAO(Metal Gear Rex)
 abcg::glBindVertexArray(m_VAO);
 //Metal gear Rex(verde militar)
 glm::mat4 model{1.0f};
 model = glm::mat4 {1.0f};
 model = glm::translate(model, glm::vec3(-1.3, 2.9f, 1.0f));
 model = glm::rotate(model, glm::radians(270.0f), glm::vec3(0, 1, 0));
 model = glm::scale(model, glm::vec3(0.4f));

 abcg::glUniformMatrix4fv(modelMatrixLoc, 1, GL_FALSE, &model[0][0]);
 abcg::glUniform4f(colorLoc, 0.29f, 0.33f, 0.13f, 1.0f);
 abcg::glDrawElements(GL_TRIANGLES, m_indices.size(), GL_UNSIGNED_INT,
                      nullptr);
```