Tarefa 2 – Computação Gráfica

Aluno: Bento Bruno Contarini Gonçalves
Matrícula: 2311122

Arquivo “main.cpp”:

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>

#define GLM_ENABLE_EXPERIMENTAL

#include "error.h"
#include "triangle.h"
#include "shader.h"
#include "polygon.h"
#include "circle.h"
#include "line.h"

#include <chrono>

#include <stdio.h>
#include <stdlib.h>

#include <glm/glm.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <glm/ext/matrix_transform.hpp>
#include <glm/gtx/string_cast.hpp>

#include <time.h>

static CirclePtr circ;
static TrianglePtr tri;
static ShaderPtr shd;
static PolygonPtr poly;
static LinePtr line;
static GLint uModelLoc = -1;

static float angSeg = 0.0f, angMin = 0.0f, angHr = 0.0f;

static void error (int code, const char* msg)
{
printf("GLFW error %d: %s\n", code, msg);
glfwTerminate();
exit(1);
}

static void keyboard (GLFWwindow* window, int key, int scancode, int action, int mods)
```

```
{
if (key == GLFW_KEY_Q && action == GLFW_PRESS)
glfwSetWindowShouldClose(window, GLFW_TRUE);
}

static void resize (GLFWwindow* win, int width, int height)
{
glViewport(0,0,width,height);
}

static void initialize ()
{
glClearColor(1.0f,1.0f,1.0f,1.0f);
tri = Triangle::Make();
shd = Shader::Make();
poly = Polygon::Make();
circ = Circle::Make(0.01f);
line = Line::Make();
shd->AttachVertexShader("shaders/vertex.glsl");
shd->AttachFragmentShader("shaders/fragment.glsl");

//inicializar hora
using clock = std::chrono::system_clock;
auto now = clock::now();

std::time_t tt = clock::to_time_t(now);
std::tm local = *std::localtime(&tt);

float seg = local.tm_sec;
float min = local.tm_min + seg / 60.0;
float hr = (local.tm_hour % 12) + min / 60.0;

angSeg = (M_PI * 2.0f) * (seg / 60.0f);
angMin = (M_PI * 2.0f) * (min / 60.0f);
angHr = (M_PI * 2.0f) * (hr / 12.0f);

shd->Link();
Error::Check("initialize");
}

static void display (GLFWwindow* win)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
shd->UseProgram();

// desenhar o relógio
// contorno
```

```
glm::mat4 circRelogio(1.0f);
circRelogio = glm::scale(circRelogio, glm::vec3(0.93f, 0.93f, 1.0f));
shd->SetUniform("M", circRelogio);
circ->Draw();

// linha das horas
for (int i = 0; i < 12; i++)
{
glm::mat4 lineRelogio(1.0f);
float ang = (i / 12.0f) * (M_PI * 2.0f);
lineRelogio = glm::rotate(lineRelogio, ang, glm::vec3(0, 0, 1));
lineRelogio = glm::translate(lineRelogio, glm::vec3(0.7f, 0.0f, 0.0f));
lineRelogio = glm::scale(lineRelogio, glm::vec3(0.45f, 2.0f, 1.0f));
shd->SetUniform("M", lineRelogio);
line->Draw();
}
Error::Check("display");
}

void update(float dt)
{
angSeg += ((M_PI * 2.0f) / 60.0) * dt;
angMin += (M_PI * 2.0f) / (60.0 * 60.0) * dt;
angHr += (M_PI * 2.0f) / (12.0 * 60.0 * 60.0) * dt;

glm::mat4 M(1.0f);

glm::mat4 Mhr = glm::rotate(M, (float) M_PI_2 -angHr, glm::vec3(0,0,1));
glm::mat4 Mmin = glm::rotate(M, (float) M_PI_2 -angMin, glm::vec3(0,0,1));
glm::mat4 Mseg = glm::rotate(M, (float) M_PI_2 -angSeg, glm::vec3(0,0,1));

Mhr = glm::scale(Mhr, glm::vec3(0.9f, 4.0f, 1.0f));
Mmin = glm::scale(Mmin, glm::vec3(1.3f, 2.5f, 1.0f));
Mseg = glm::scale(Mseg, glm::vec3(1.50f, 0.90f, 1.0f));

shd->SetUniform("M", Mseg);
shd->SetUniform("icolor", glm::vec4(1.0f, 0.0f, 0.0f, 1.0f));
line->Draw();

shd->SetUniform("M", Mmin);
shd->SetUniform("icolor", glm::vec4(0.7f, 0.7f, 0.7f, 1.0f));
line->Draw();

shd->SetUniform("M", Mhr);
shd->SetUniform("icolor", glm::vec4(0.0f, 0.0f, 0.0f, 1.0f));
line->Draw();
}
```

```
int main ()
{
glfwInit();
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR,4);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR,1);
glfwWindowHint(GLFW_OPENGL_PROFILE,GLFW_OPENGL_CORE_PROFILE);
glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT,GL_TRUE); // required for mac os
//glfwWindowHint(GLFW_COCOA_RETINA_FRAMEBUFFER,GLFW_TRUE); // option for mac os

glfwSetErrorCallback(error);

GLFWwindow* win = glfwCreateWindow(600,400,"Relogio teste",nullptr,nullptr);
glfwSetFramebufferSizeCallback(win, resize); // resize callback
glfwSetKeyCallback(win, keyboard); // keyboard callback
glfwMakeContextCurrent(win);

if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
{
printf("Failed to initialize GLAD OpenGL context\n");
exit(1);
}

printf("OpenGL version: %s\n", glGetString(GL_VERSION));

initialize();

double t0 = glfwGetTime();
double t;
while(!glfwWindowShouldClose(win)) {
t = glfwGetTime();
display(win);
update(t-t0);
t0 = t;
glfwSwapBuffers(win);
glfwPollEvents();
}
glfwTerminate();
return 0;
}
```

Tudo aparenta estar funcionando como devido, o video seguirá em anexo. Pode haver uma certa distorçao de proporção, devido a tela do meu laptop ser larga, porém não muito alta.