# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №5**

**по дисциплине «Компьютерная графика»**

**Тема: «Программируемый графический конвейер. Шейдеры.»**

|  |  |
| --- | --- |
| Студент гр. 6381 | Фиалковский М.С. |
| Студент гр. 6381 | Афийчук И.И. |
| Преподаватель | Герасимова Т.В. |

## Санкт-Петербург 2019

**Задание.**

Реализовать программу, симулирующую движение фонариком по стене при отсутствии других источников света с помощью современных подходов по использованию библиотеки OpenGL.

**Общие сведения.**

Под современным подходом, указанным выше, имеется ввиду core-profile режим – более эффективный и гибкий чем официально устаревший в OpenGL 3.3 immediate режим. В нем мы сами специфицируем некоторые этапы графического конвейера, разрабатывая так называемый шейдеры.

В данной работе мы будем разрабатывать вершинный и фрагментный шейдеры.

**Ход работы.**

Сборка и компиляция программы осуществляется с помощью утилиты cmake и компилятора gcc из пакета MinGW. Графический интерфейс выполнен с помощью библиотеки [FLTK.](https://github.com/ocornut/imgui) В качестве обертки над библиотекой OpenGL, используется GLEW. FLTK обеспечивает создание контекста и имеет интерфейс для обращения к некоторым командам OpenGL.

Загрузка шейдеров происходит в несколько этапов. В самом начале их исходный код читается из указанных файлов в *vShaderCode* и *fShaderCode*. Затем этот исходный код компилируется:

*vertex = glCreateShader(GL\_VERTEX\_SHADER);* glShaderSource(vertex, 1, &vShaderCode, NULL); glCompileShader(vertex);

*…*

*fragment = glCreateShader(GL\_FRAGMENT\_SHADER);* glShaderSource(fragment, 1, &fShaderCode, NULL); glCompileShader(fragment);

## При удачной компиляции полученные шейдерные программы линкуются в один шейдер с помощью команд:

*this->Program = glCreateProgram();* glAttachShader(this->Program, vertex); glAttachShader(this->Program, fragment); glLinkProgram(this->Program);

## Полученный таким образом шейдер мы затем используем в игровом цикле при отрисовке требуемой графики.

Рассмотрим сам код шейдеров:

1. Фрагментный:

*#version 330 core* in vec3 ourColor; in vec2 TexCoord; out vec4 color;

*uniform vec2 mousePos;* uniform sampler2D ourTexture; uniform float fadeDistance;

*float getDistance(vec2 p1, vec2 p2) {*

*return sqrt(pow(p1.x-p2.x, 2) + pow(p1.y-p2.y, 2));*

*}*

*float getLightIntensity() {*

*float dist = getDistance(gl\_FragCoord.xy, mousePos);* return fadeDistance / dist;

*}*

*void main(){*

*color = texture(ourTexture, TexCoord) \* getLightIntensity();*

*}*

## Вершинный:

*#version 330 core*

*layout (location = 0) in vec3 position;* layout (location = 1) in vec3 color; layout (location = 2) in vec2 texCoord;

*out vec3 ourColor;* out vec2 TexCoord;

*void main()*

*{*

*gl\_Position = vec4(position, 1.0f);* ourColor = color;

*TexCoord = texCoord;*

*}*

## Для получения требуемого эффекта в шейдер требуется передавать

текущие координаты мыши при их изменении и текстуру для наложения. Для этого объявим в фрагментном шейдере (так как именно от отвечает за вычисление цвета) переменные:

*uniform vec2 mousePos;* uniform sampler2D ourTexture;

## Их значения будем менять в с помощью специальных команд OpenGL, например:

*if (event == FL\_MOVE){*

*auto x = Fl::event\_x();* auto y = Fl::event\_y();

*glUniform2f(glGetUniformLocation(shaderProgram.Program, "mousePos"),*

*x, y);*

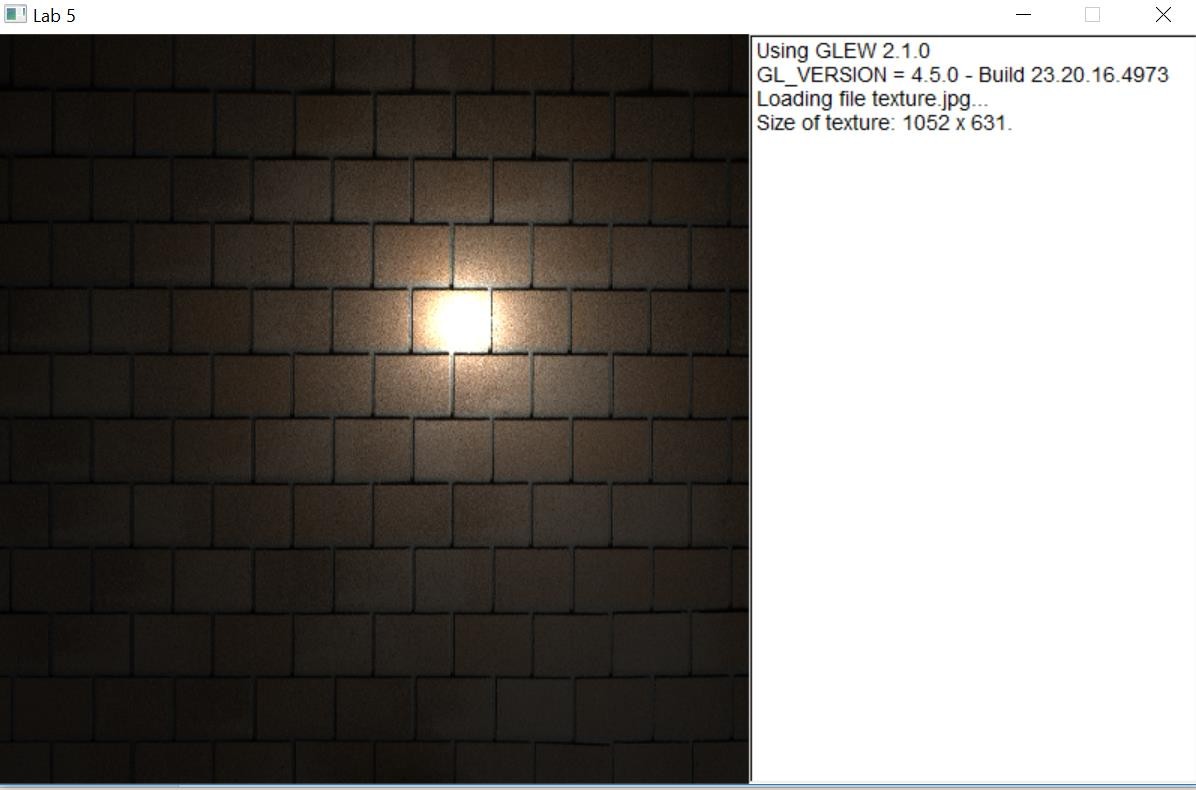
*}*

*redraw();*

# Тестирование.

## Справа от области отрисовки находятся элементы UI, с помощью которых выводятся сообщения от графической системы.

Вслед за движением мыши по области отрисовки меняется и честь осветленной фонариком стены. Двигая колёсиком мыши можно отдалять и приближать источник света.





**Вывод.**

В процессе выполнения лабораторной работы была разработана программа, требуемый по заданию эффект. При выполнении работы были приобретены навыки работы с графическим конвейером и шейдернымм программам из графической библиотеки OpenGL.

Исходный код метода Draw:

*#include "SimpleGL3Window.hpp"*

*SimpleGL3Window::SimpleGL3Window(int x, int y, int w, int h) :* Fl\_Gl\_Window(x, y, w, h) {

*mode(FL\_RGB8 | FL\_DOUBLE | FL\_OPENGL3);*

*}*

*void SimpleGL3Window::draw(void) {* shaderProgram.readAndCompile("Shaders/vertex.shader",

*"Shaders/fragment.shader");* loadTexture("texture.jpg"); loadBuffers();

*glClearColor(0.2f, 0.3f, 0.3f, 1.0f);* glClear(GL\_COLOR\_BUFFER\_BIT);

*glBindTexture(GL\_TEXTURE\_2D, texture);* shaderProgram.Use();

*glUniform1f(glGetUniformLocation(shaderProgram.Program, "fadeDistance"),* fadeDistance);

*glBindVertexArray(VAO);*

*glDrawElements(GL\_TRIANGLES, 6, GL\_UNSIGNED\_INT, 0);* glBindVertexArray(0);

*}*

*int SimpleGL3Window::handle(int event) {* static int first = 1;

*if (first && event == FL\_SHOW && shown()) {* first = 0;

*make\_current();*

*{*

*GLenum err = glewInit(); // defines pters to functions of OpenGL* V 1.2 and above

*if (err)*

*Fl::warning("glewInit() failed returning %u", err);* else

*add\_output("Using GLEW %s\n", glewGetString(GLEW\_VERSION));*

*}*

*const uchar\* glv = glGetString(GL\_VERSION);* add\_output("GL\_VERSION = %s\n", glv);

*}*

*if (event == FL\_MOVE){*

*auto x = Fl::event\_x();* auto y = Fl::event\_y();

*x, y);*

*}*

*glUniform2f(glGetUniformLocation(shaderProgram.Program, "mousePos"),* redraw();

*if (event == FL\_MOUSEWHEEL) {*

*auto scroll\_size = Fl::event\_dy();*

*fadeDistance -= static\_cast<GLfloat>(scroll\_size);* if (fadeDistance < 0)

*fadeDistance = 0.0f;* glUniform1f(glGetUniformLocation(shaderProgram.Program,

*"fadeDistance"), fadeDistance);* redraw();

*}*

*return Fl\_Gl\_Window::handle(event);*

*}*

*void SimpleGL3Window::reset(void) {*

*}*

*void SimpleGL3Window::loadTexture(const char \*file){* static bool isLoaded = false;

*if (isLoaded)*

*return;*

*add\_output("Loading file %s...\n", file);*

*image = SOIL\_load\_image(file, &width, &height, 0, SOIL\_LOAD\_RGB);* add\_output("Size of texture: %d x %d.\n", width, height);

*// Load and create a texture* glGenTextures(1, &texture); glBindTexture(GL\_TEXTURE\_2D, texture);

*// Set the texture wrapping parameters* glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT); glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

*// Set texture filtering parameters* glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR); glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

*// Load image, create texture and generate mipmaps* glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height, 0, GL\_RGB,

*GL\_UNSIGNED\_BYTE, image);*

*glGenerateMipmap(GL\_TEXTURE\_2D);* SOIL\_free\_image\_data(image); glBindTexture(GL\_TEXTURE\_2D, 0); isLoaded = true;

*}*

*void SimpleGL3Window::loadBuffers(){* static bool isLoaded = false;

*if (isLoaded)*

*return;*

*GLfloat vertices[] = {*

*// Positions // Colors // Texture Coords*

*1.0f, 1.0f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, // Top Right*

*1.0f, -1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, // Bottom Right*

*-1.0f, -1.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, // Bottom Left*

*-1.0f, 1.0f, 0.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f // Top Left*

*};*

*GLuint indices[] = {*

*0, 1, 3, // First Triangle*

*1, 2, 3 // Second Triangle*

*};*

*GLuint VBO, EBO;*

*glGenVertexArrays(1, &VAO);* glGenBuffers(1, &VBO); glGenBuffers(1, &EBO);

*glBindVertexArray(VAO);* glBindBuffer(GL\_ARRAY\_BUFFER, VBO);

*glBufferData(GL\_ARRAY\_BUFFER, sizeof(vertices), vertices, GL\_STATIC\_DRAW);* glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, EBO); glBufferData(GL\_ELEMENT\_ARRAY\_BUFFER, sizeof(indices), indices,

*GL\_STATIC\_DRAW);*

*// Position attribute*

*glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 8 \* sizeof(GLfloat),* (GLvoid\*)0);

*glEnableVertexAttribArray(0);*

*// Color attribute*

*glVertexAttribPointer(1, 3, GL\_FLOAT, GL\_FALSE, 8 \* sizeof(GLfloat),* (GLvoid\*)(3 \* sizeof(GLfloat)));

*glEnableVertexAttribArray(1);*

*// TexCoord attribute*

*glVertexAttribPointer(2, 2, GL\_FLOAT, GL\_FALSE, 8 \* sizeof(GLfloat),* (GLvoid\*)(6 \* sizeof(GLfloat)));

*glEnableVertexAttribArray(2);* glBindVertexArray(0);

*}*