ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асп. |  |  |  | Д.А. Смолиенко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7 |
| Освоение работы с программируемыми шейдерами. |
| по дисциплине:  КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4631 |  |  |  | С.А. Гришин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Задание на лабораторную работу**

Выполнить на выбор один из вариантов:

1. Шейдер с сохранением данных в текстуре,
2. Процедурный, текстурный шейдер,
3. Традиционные шейдер,
4. Фрагментный шейдер,
5. Вершинный шейдер.

**Текст программы**

#include <stdio.h>

#include <stdlib.h>

#include <GL/glew.h>

#include <GL/glut.h>

GLuint program;

GLint attribute\_coord2d;

GLfloat triangle\_vertices[] = {

-0.8, 0.8, -0.8, -0.8, 0.8, -0.8, 0.8, 0.8,

};

void onDisplay()

{

GLint compile\_ok = GL\_FALSE, link\_ok = GL\_FALSE;

GLuint vs = glCreateShader(GL\_VERTEX\_SHADER);

const char\* vs\_source = "#version 120\n" // OpenGL 2.1

"attribute vec2 coord2d; "

"void main(void) { "

" gl\_Position = vec4(coord2d, 0.0, 1.0); "

"}";

glShaderSource(vs, 1, &vs\_source, NULL);

glCompileShader(vs);

glGetShaderiv(vs, GL\_COMPILE\_STATUS, &compile\_ok);

if (0 == compile\_ok) {

fprintf(stderr, "Error in vertex shader\n");

}

GLuint fs = glCreateShader(GL\_FRAGMENT\_SHADER);

const char\* fs\_source = "#version 120 \n"

"void main(void) { "

"gl\_FragColor[0] = gl\_FragCoord.x / 640.0 ;"

"gl\_FragColor[1] = gl\_FragCoord.y / 480.0;"

"gl\_FragColor[3] = noise1(gl\_FragCoord.x / 640.0) \* 0.5 + 0.5;"

"}";

glShaderSource(fs, 1, &fs\_source, NULL);

glCompileShader(fs);

glGetShaderiv(fs, GL\_COMPILE\_STATUS, &compile\_ok);

if (!compile\_ok) {

fprintf(stderr, "Error in fragment shader\n");

}

program = glCreateProgram();

glAttachShader(program, vs);

glAttachShader(program, fs);

glLinkProgram(program);

glGetProgramiv(program, GL\_LINK\_STATUS, &link\_ok);

if (!link\_ok) {

fprintf(stderr, "glLinkProgram:");

}

const char\* attribute\_name = "coord2d";

attribute\_coord2d = glGetAttribLocation(program, attribute\_name);

if (attribute\_coord2d == -1) {

fprintf(stderr, "Could not bind attribute %s\n", attribute\_name);

}

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glUseProgram(program);

glEnableVertexAttribArray(attribute\_coord2d);

glVertexAttribPointer(attribute\_coord2d, 2, GL\_FLOAT, GL\_FALSE, 0,

triangle\_vertices);

glDrawArrays(GL\_QUADS, 0, 4);

glDisableVertexAttribArray(attribute\_coord2d);

glutSwapBuffers();

}

void free\_resources() { glDeleteProgram(program); }

GLfloat Mz[2][2] = { { 1, -0.001 }, { 0.001, 1 } };

void idle(void)

{

triangle\_vertices[0] = triangle\_vertices[0] \* Mz[0][0] + triangle\_vertices[1] \* Mz[0][1];

triangle\_vertices[1] = triangle\_vertices[0] \* Mz[1][0] + triangle\_vertices[1] \* Mz[1][1];

triangle\_vertices[2] = triangle\_vertices[2] \* Mz[0][0] + triangle\_vertices[3] \* Mz[0][1];

triangle\_vertices[3] = triangle\_vertices[2] \* Mz[1][0] + triangle\_vertices[3] \* Mz[1][1];

triangle\_vertices[4] = triangle\_vertices[4] \* Mz[0][0] + triangle\_vertices[5] \* Mz[0][1];

triangle\_vertices[5] = triangle\_vertices[4] \* Mz[1][0] + triangle\_vertices[5] \* Mz[1][1];

glutPostRedisplay();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowSize(640, 480);

glutCreateWindow("LR7");

GLenum glew\_status = glewInit();

if (glew\_status != GLEW\_OK) {

fprintf(stderr, "Error: %s\n", glewGetErrorString(glew\_status));

return EXIT\_FAILURE;

}

glutIdleFunc(idle);

glutDisplayFunc(onDisplay);

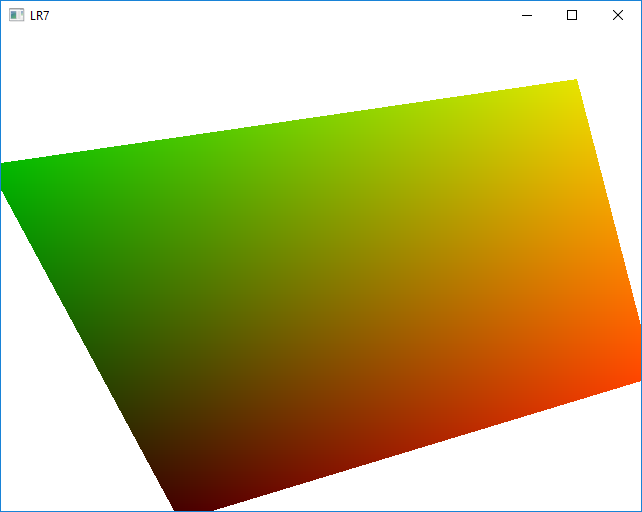
glutMainLoop();

free\_resources();

return EXIT\_SUCCESS;

}

**Результат работы программы**



**Выводы**

В результате лабораторной работы разработано приложение выводящее треугольник, созданный при использовании вершинного и фрагментного шейдера.