МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Компьютерная графика»
Тема: Примитивы OpenGL

Студент гр. 8383 _____ Бессуднов Г. И.

Преподаватель _____ Герасимова Т. В.

Санкт-Петербург
2021

Цель работы

Разработать программу, реализующую представление определенного набора примитивов из имеющихся в библиотеке OpenGL (GL_POINT, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON).

Разработанная на базе шаблона программа должна быть пополнена возможностями остановки интерактивно различных атрибутов примитивов рисования через вызов соответствующих элементов интерфейса пользователя

Основные теоретические положения

В данной лабораторной работе должны быть рассмотрены следующие примитивы:

GL_POINTS – каждая вершина рассматривается как отдельная точка, параметры которой не зависят от параметров остальных заданных точек. При этом вершина n определяет точку n. Рисуется N точек (n – номер текущей вершины, N – общее число вершин).

Основой графики OpenGL являются вершины. Для их определения используется команда glVertex.

void glVertex[2 3 4][s i f d](type coord)

Вызов команды определяется четырьмя координатами x, y, z и w. При этом вызов glVertex2* устанавливает координаты x и y, координата z полагается равной 0, а w – 1. Вызов glVertex3* устанавливает координаты x, y, z, а w равно 1.

GL_LINES – каждая пара вершин рассматривается как независимый отрезок. Первые две вершины определяют первый отрезок, следующие две – второй отрезок и т.д., вершины (2n-1) и 2n определяют отрезок n. Всего рисуется N/2 линий. Если число вершин нечетно, то последняя просто игнорируется.

GL_LINE_STRIP – в этом режиме рисуется последовательность из одного или нескольких связанных отрезков. Первая вершина задает начало первого отрезка, а вторая – конец первого, который является также началом второго. В общем случае, вершина n (n > 1) определяет начало отрезка n и конец отрезка (n - 1). Всего рисуется (N - 1) отрезок.

GL_LINE_LOOP – осуществляется рисование замкнутой кривой линии. Первая вершина задает начало первого отрезка, а вторая – конец первого, который является также началом второго. В общем случае, вершина n (n > 1) определяет начало отрезка n и конец отрезка (n - 1). Первая вершина является концом последнего отрезка. Всего рисуется N отрезков.

GL_TRIANGLES – каждая тройка вершин рассматривается как независимый треугольник. Вершины (3n-2), (3n-1), 3n (в таком порядке) определяют треугольник n. Если число вершин не кратно 3, то оставшиеся (одна или две) вершины игнорируются. Всего рисуется N/3 треугольника.

GL_TRIANGLE_STRIP - в этом режиме рисуется группа связанных треугольников, имеющих общую грань. Первые три вершины определяют первый треугольник, вторая, третья и четвертая – второй и т.д. для нечетного n вершины n, (n+1) и (n+2) определяют треугольник n. Для четного n треугольник определяют вершины (n+1), n и (n+2). Всего рисуется (N-2) треугольника.

GL_TRIANGLE_FAN - в этом режиме рисуется группа связанных треугольников, имеющих общие грани и одну общую вершину. Первые три вершины определяют первый треугольник, первая, третья и четвертая – второй и т.д. Всего рисуется (N-2) треугольника.

GL_QUADS – каждая группа из четырех вершин рассматривается как независимый четырехугольник. Вершины (4n-3), (4n-2), (4n-1) и 4n определяют четырехугольник n. Если число вершин не кратно 4, то оставшиеся (одна, две или три) вершины игнорируются. Всего рисуется N/4 четырехугольника.

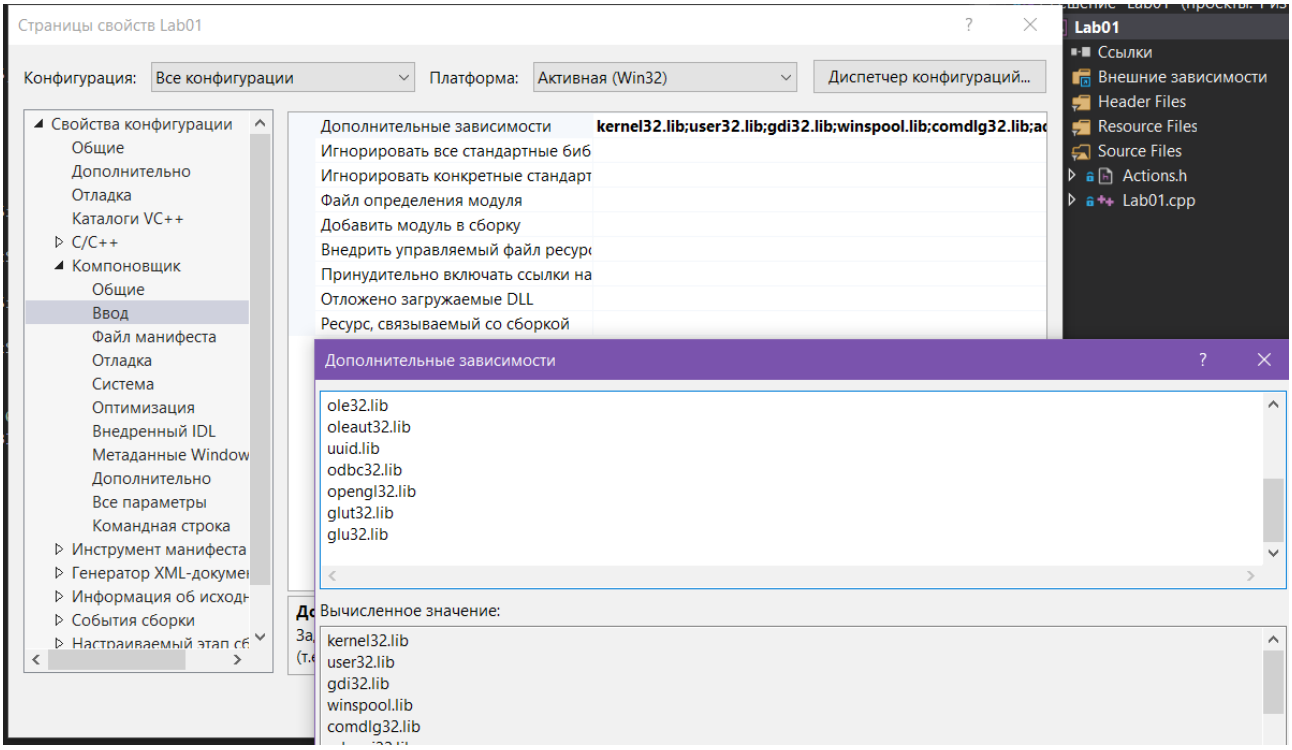
GL_QUAD_STRIP – рисуется группа четырехугольников, имеющих общую грань. Первая группа из четырех вершин задает первый

четырехугольник. Третья, четвертая, пятая и шестая задают второй четырехугольник.

GL_POLYGON – задет многоугольник. При этом число вершин равно числу вершин рисуемого многоугольника.

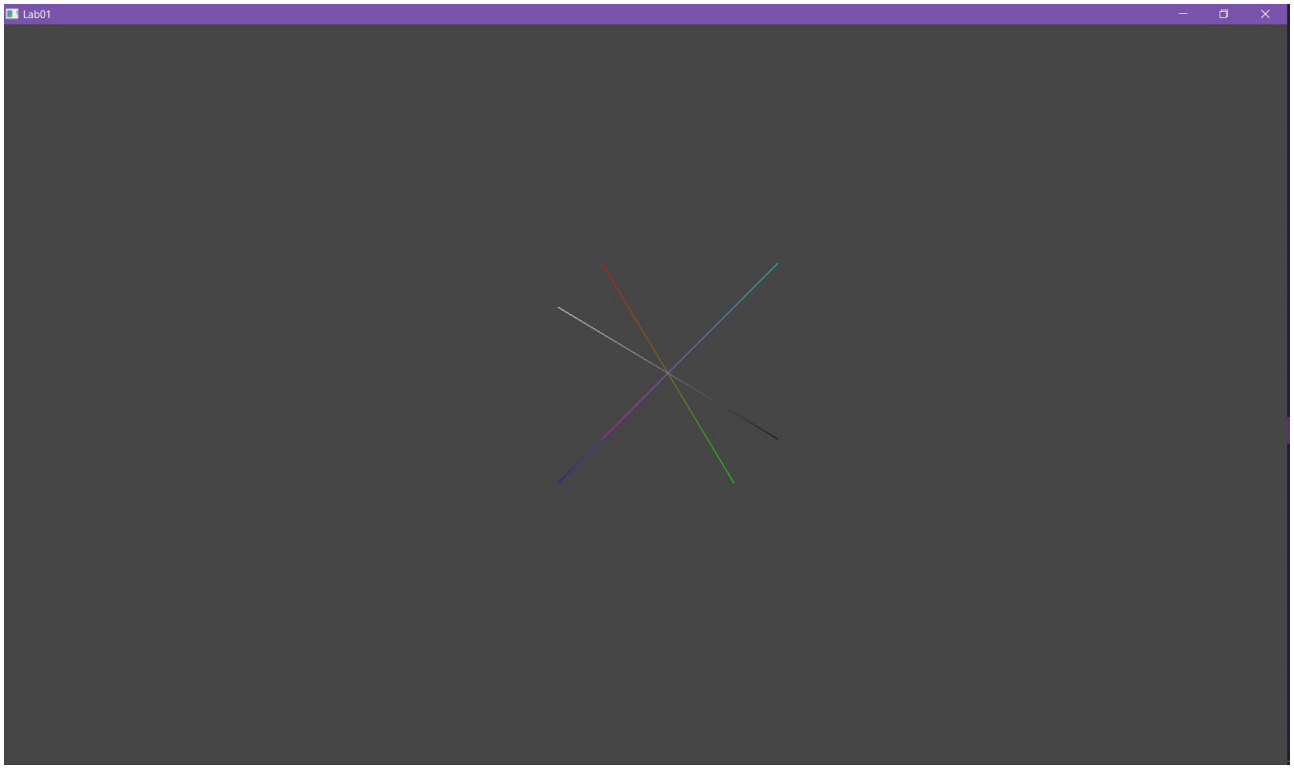
Выполнение работы

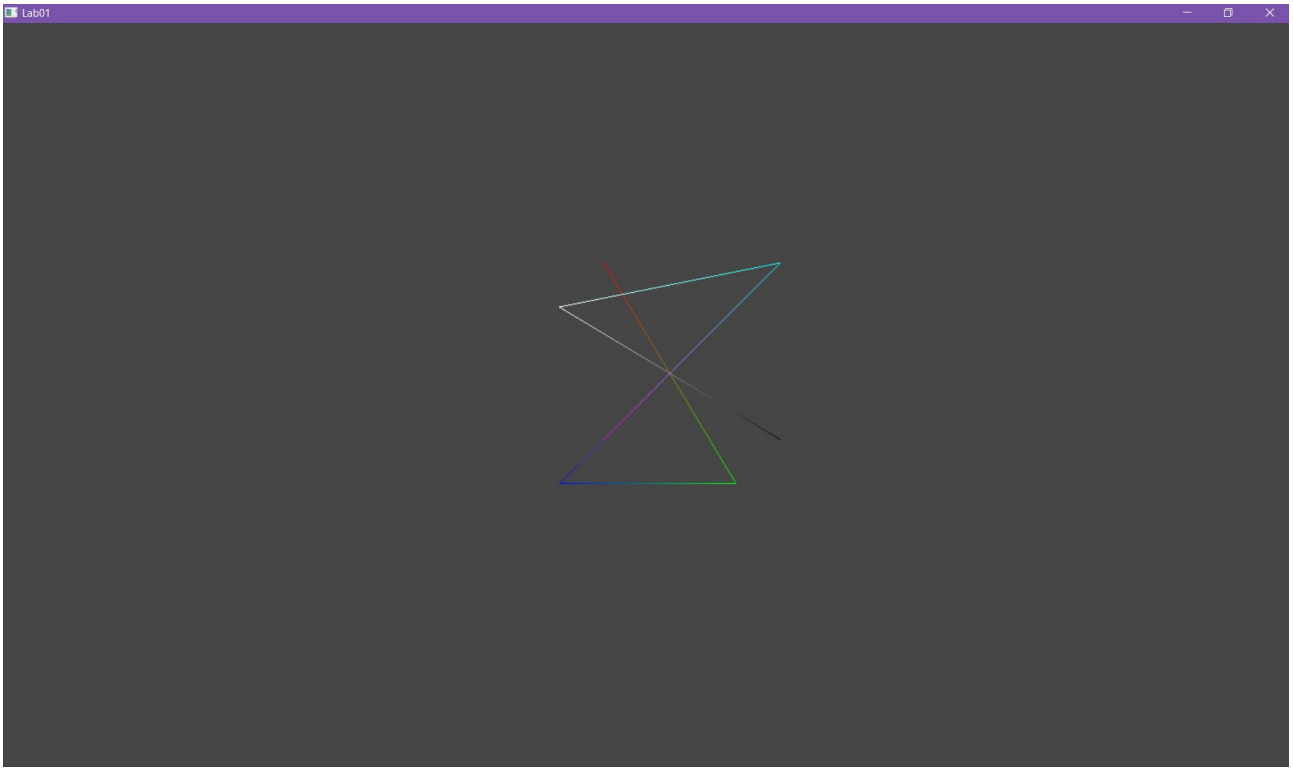
Работа была выполнена в среде Visual Studio 2019. Для подключения OpenGL были скопированы файлы glut.dll и glut32.dll в системную папку компьютера. Далее в папки библиотек Visual Studio были добавлены glut.lib и glut32.lib. В папку заголовочных файлов Visual Studio был добавлен файл glut.h. Затем в свойствах проекта были добавлены дополнительные зависимости: opengl32.lib, glut32.lib, glu32.lib.

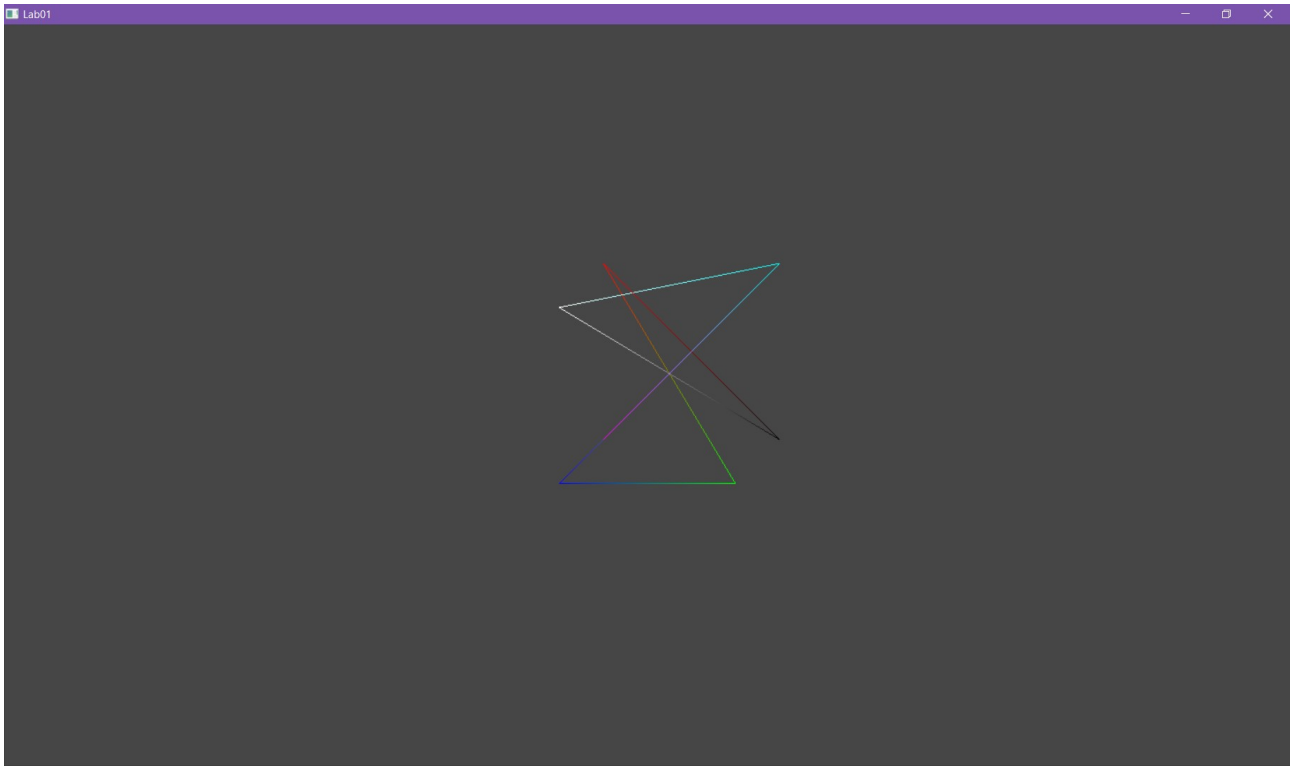


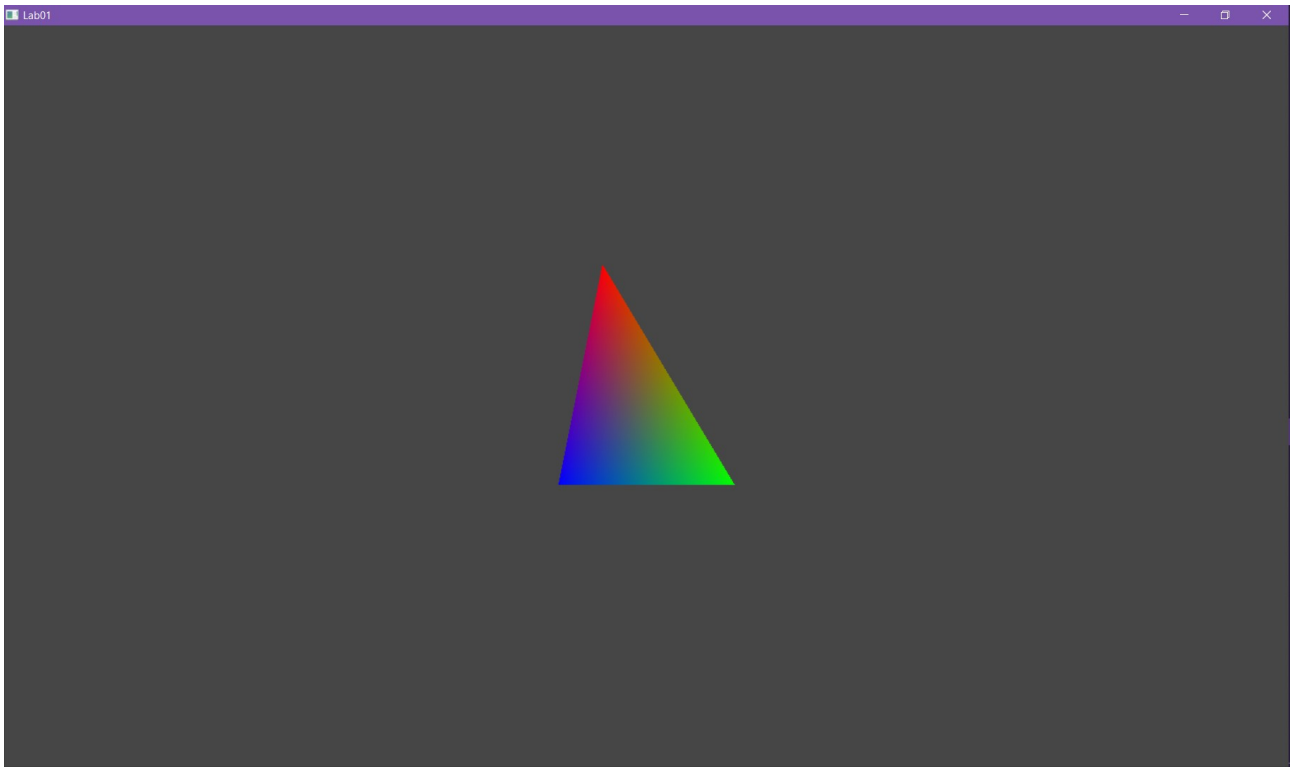
Код программы представлен в Приложении А. Результаты тестирования представлены ниже.

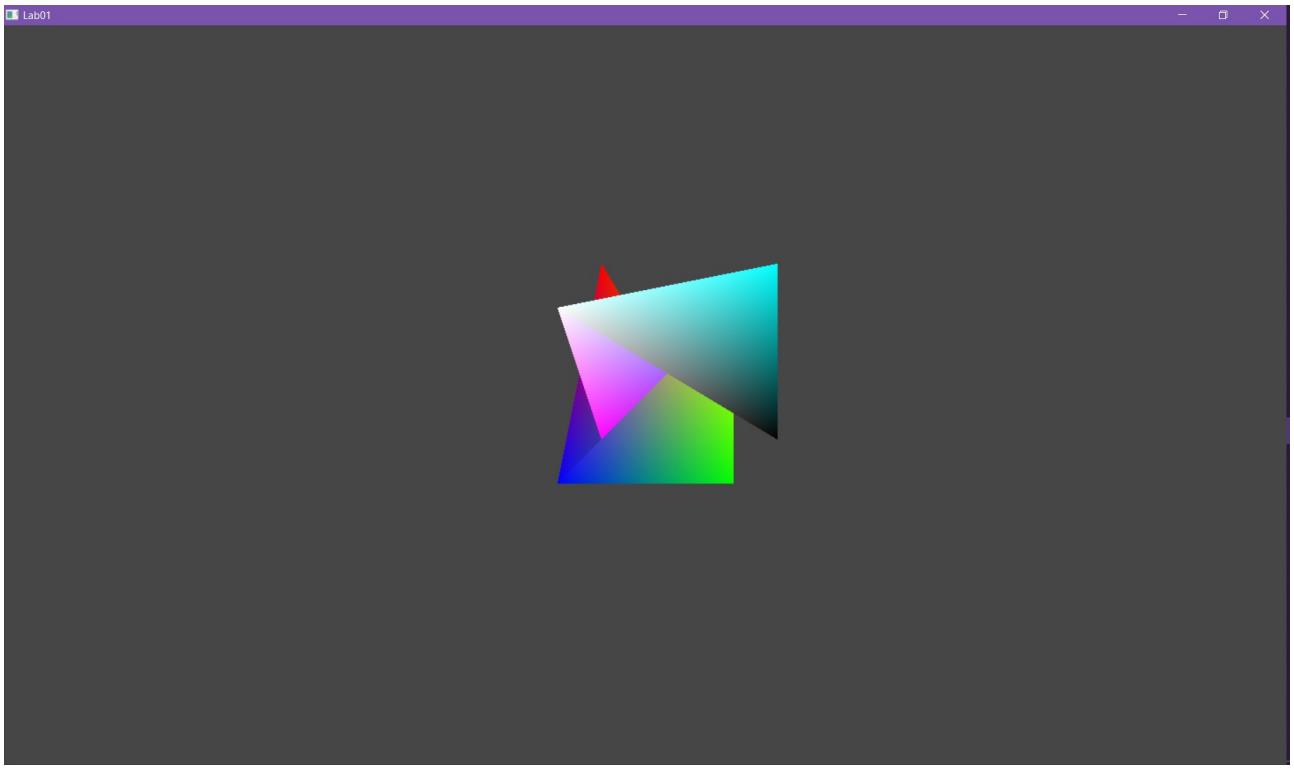

Lab01

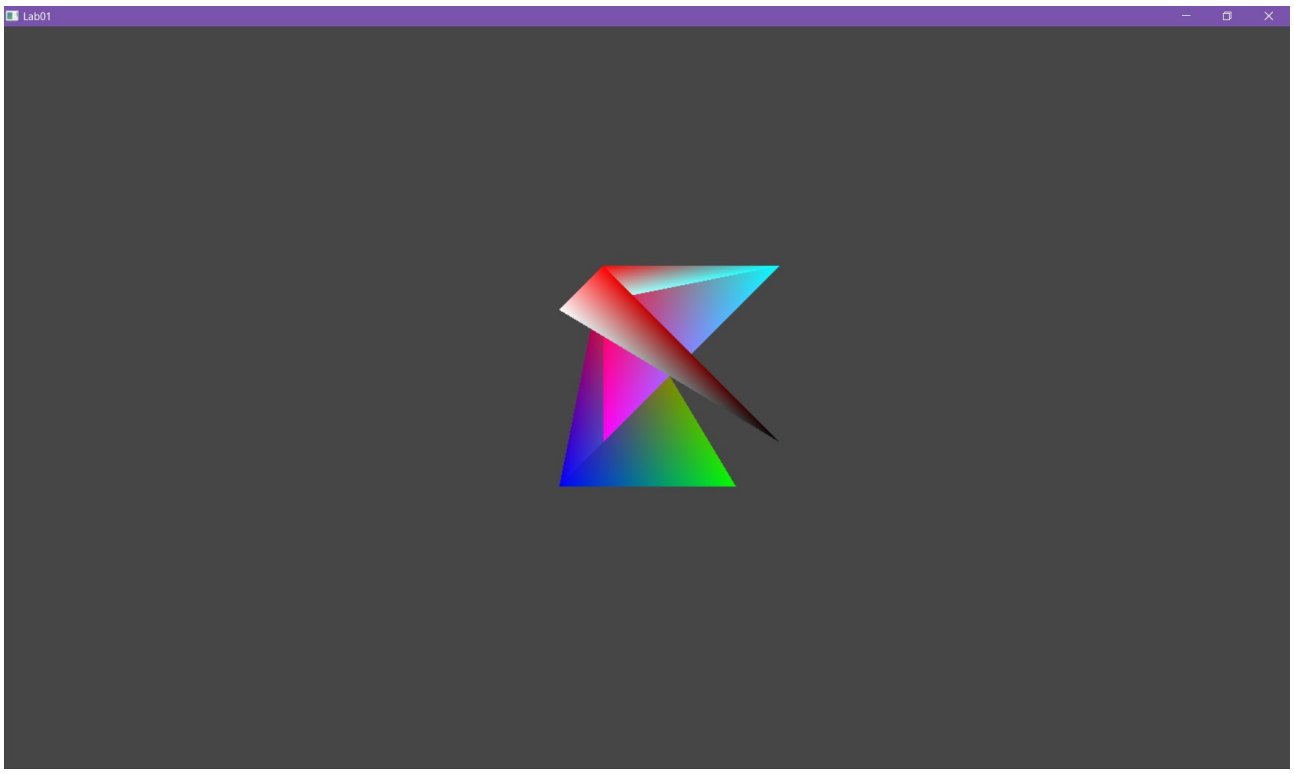

Lab01

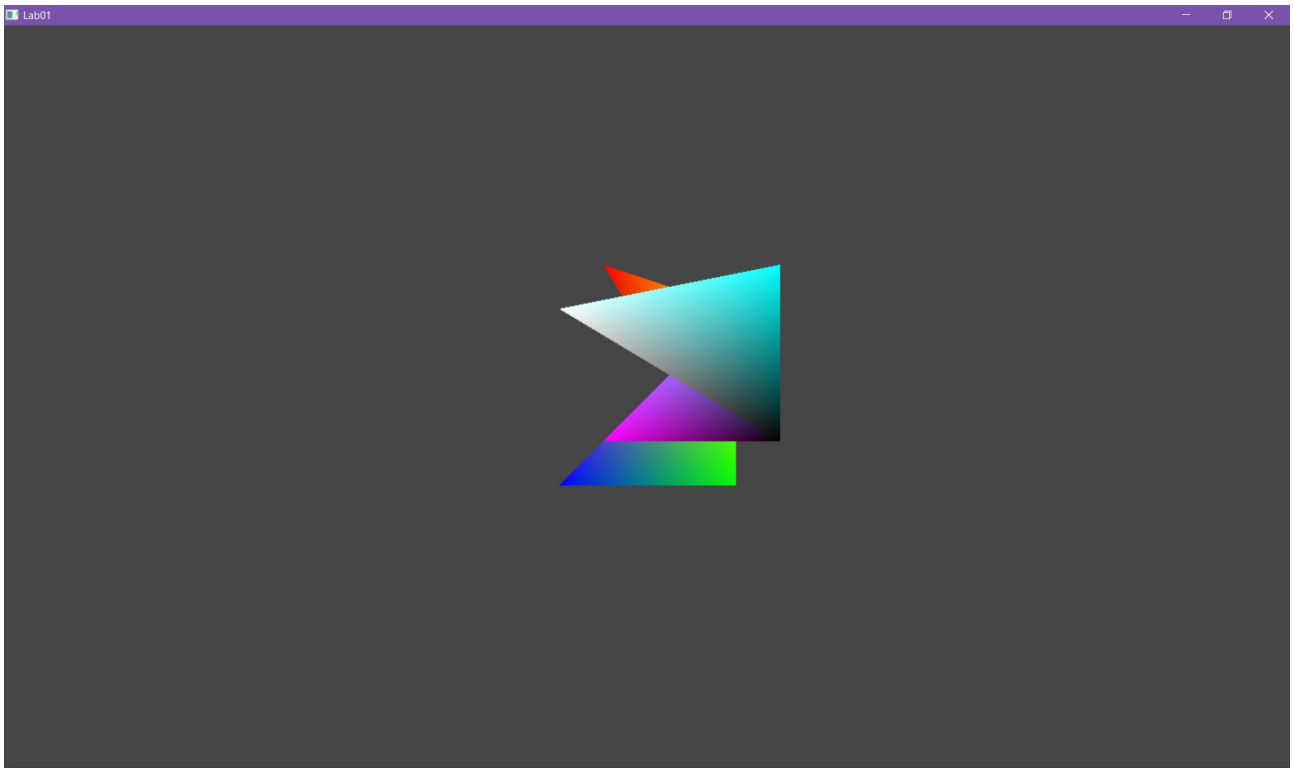

Lab01

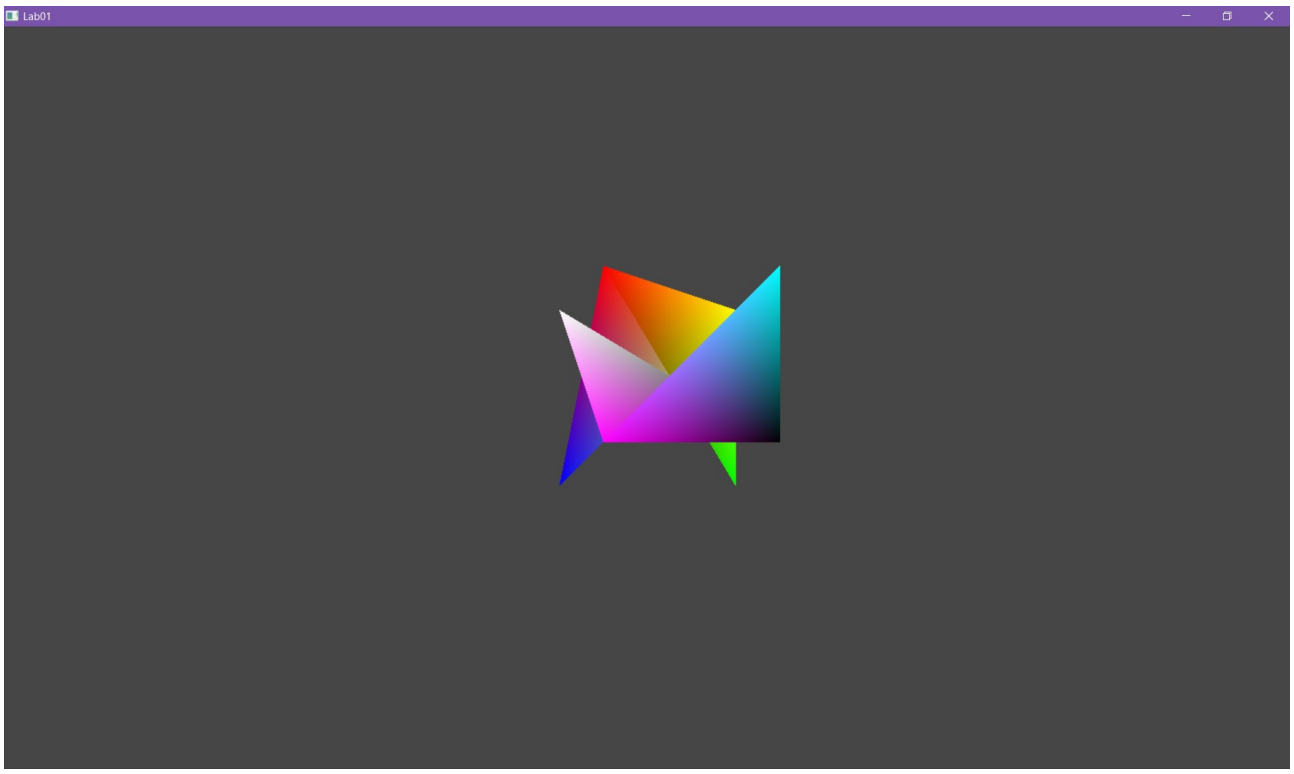

Lab01

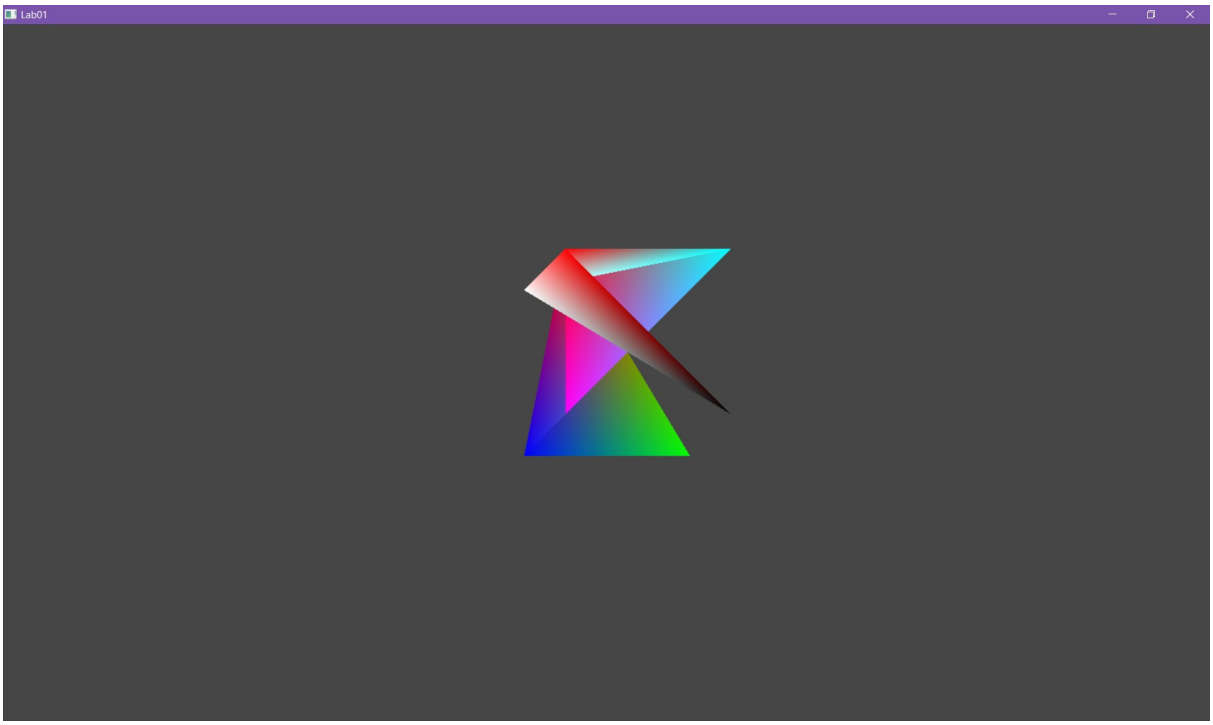

Lab01


Lab01


Lab01


Lab01





Выводы

В результате выполнения лабораторной работы была выполнена настройка OpenGL для Visual Studio 2019 и успешно написана программа, демонстрирующая разные примитивы.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
#INCLUDE <stdlib.h>

#include <glut.h>
#include <iostream>


#define ACTION_GL_POINTS '1'
#define ACTION_GL_LINES '2'
#define ACTION_GL_LINE_STRIP '3'
#define ACTION_GL_LINE_LOOP '4'
#define ACTION_GL_TRIANGLES '5'
#define ACTION_GL_TRIANGLE_STRIP '6'
#define ACTION_GL_TRIANGLE_FAN '7'
#define ACTION_GL_QUADS '8'
#define ACTION_GL_QUAD_STRIP '9'
#define ACTION_GL_POLYGON '0'

GLint Width = 512;
GLint Height = 512;

GLenum DrawMethod = GL_POINTS;

int RectSize = 200;
int mainMenu, DrawModeMenu;

void Display(void) {
float x1 = (Width - RectSize) / 2;
float x2 = x1 + RectSize;
float y1 = (Height - RectSize) / 2;
float y2 = y1 + RectSize;
float x3 = (Width - RectSize / 2) / 2;
float x4 = x3 + RectSize;
float y3 = (Height - RectSize / 2) / 2;
float y4 = y3 + RectSize;

glClearColor(0.25, 0.25, 0.25, 1);
glClear(GL_COLOR_BUFFER_BIT);

glBegin(DrawMethod);
glColor3f(1, 0, 0);
glVertex2f(x3, y4);
glColor3f(0, 1, 0);
glVertex2f(x2, y1);
glColor3f(0, 0, 1);
glVertex2f(x1, y1);
glColor3f(1, 1, 0);
glVertex2f(x2, y2);
glColor3f(1, 0, 1);
```

```
glVertex2f(x3, y3);
glColor3f(0, 1, 1);
glVertex2f(x4, y4);
glColor3f(1, 1, 1);
glVertex2f(x1, y2);
glColor3f(0, 0, 0);
glVertex2f(x4, y3);
glEnd();

glFinish();
}

void Reshape(GLint w, GLint h) {
Width = w;
Height = h;

glViewport(0, 0, w, h);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0, w, 0, h, -1.0, 1.0);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

void ChangeDrawMode(int DrawMode) {
switch(DrawMode) {
case ACTION_GL_POINTS:
DrawMethod = GL_POINTS;
break;
case ACTION_GL_LINES:
DrawMethod = GL_LINES;
break;
case ACTION_GL_LINE_STRIP:
DrawMethod = GL_LINE_STRIP;
break;
case ACTION_GL_LINE_LOOP:
DrawMethod = GL_LINE_LOOP;
break;
case ACTION_GL_TRIANGLES:
DrawMethod = GL_TRIANGLES;
break;
case ACTION_GL_TRIANGLE_STRIP:
DrawMethod = GL_TRIANGLE_STRIP;
break;
case ACTION_GL_TRIANGLE_FAN:
DrawMethod = GL_TRIANGLE_FAN;
break;
case ACTION_GL_QUADS:
DrawMethod = GL_QUADS;
```

```
break;
case ACTION_GL_QUAD_STRIP:
DrawMethod = GL_QUAD_STRIP;
break;
case ACTION_GL_POLYGON:
DrawMethod = GL_POLYGON;
break;
}
glutPostRedisplay();
}

void Keyboard(unsigned char key, int x, int y) {
if (key == '\033') {
exit(0);
}
else {
ChangeDrawMode(key);
}

}

void processMenuStatus(int status, int x, int y) {}

void processMainMenu(int option) {}

void processDrawModeMenu(int option) {
ChangeDrawMode(option);
}

void createPopupMenus() {
DrawModeMenu = glutCreateMenu(processDrawModeMenu);
glutAddMenuEntry("GL_POINTS", ACTION_GL_POINTS);
glutAddMenuEntry("GL_LINES", ACTION_GL_LINES);
glutAddMenuEntry("GL_LINE_STRIP", ACTION_GL_LINE_STRIP);
glutAddMenuEntry("GL_LINE_LOOP", ACTION_GL_LINE_LOOP);
glutAddMenuEntry("GL_TRIANGLES", ACTION_GL_TRIANGLES);
glutAddMenuEntry("GL_TRIANGLE_STRIP", ACTION_GL_TRIANGLE_STRIP);
glutAddMenuEntry("GL_TRIANGLE_FAN", ACTION_GL_TRIANGLE_FAN);
glutAddMenuEntry("GL_QUADS", ACTION_GL_QUADS);
glutAddMenuEntry("GL_QUAD_STRIP", ACTION_GL_QUAD_STRIP);
glutAddMenuEntry("GL_POLYGON", ACTION_GL_POLYGON);
mainMenu = glutCreateMenu(processMainMenu);
glutAddSubMenu("Polygon Mode", DrawModeMenu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMenuStatusFunc(processMenuStatus);
}

int main(int argc, char* argv[]) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(Width, Height);
```

```
glutCreateWindow("Lab01");

glutDisplayFunc(Display);
glutReshapeFunc(Reshape);
glutKeyboardFunc(Keyboard);

createPopupMenus();

glutMainLoop();
}
```