**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. 0304 |  | Максименко Е.М, |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

2023

**Цель работы.**

- ознакомление с основными примитивами OpenGL.

- освоение возможности подключения графической библиотеки в среду разработки.

**Задание.**

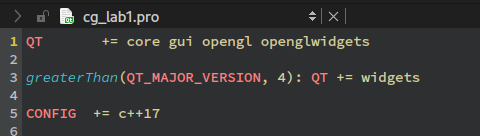
Разработать программу, реализующую представление определенного набора примитивов (4) из имеющихся в OpenGL (GL\_POINT, GL\_LINES, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_TRIANGLES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_QUADS, GL\_QUAD\_STRIP, GL\_POLYGON).

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

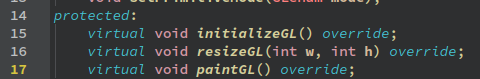
**Выполнение работы.**

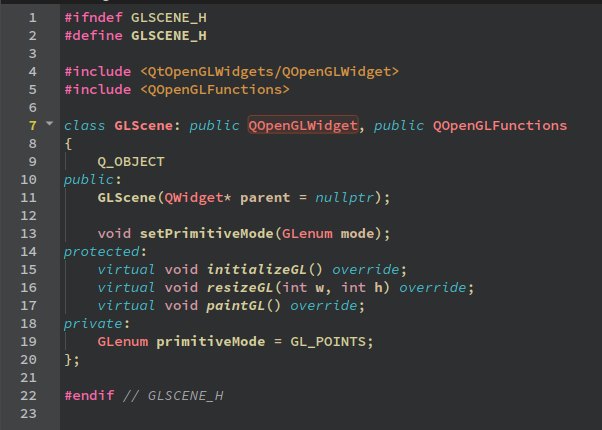
Работа была выполнена с использованием языка программирования C++ и фреймворка Qt 6. Данный фреймворк имеет поддержку виджетов, с которыми можно взаимодействовать средствами библиотеки OpenGL.

Включение поддержки OpenGL в проекте Qt происходит путем добавления в файл проекта (.pro) указаний на использование данной библиотеки (см. рис. 1).

Рисунок 1. Включение поддержки OpenGL в фреймворке Qt 6

Для отрисовки графики с помощью библиотеки OpenGL в Qt 6 используется виджет QOpenGLWidget, а также класс QOpenGLFunctions. Данный виджет предоставляет 3 основных метода для работы с графикой (см. рис. 2). Класс QOpenGLFunctions предоставляет доступ к функциям OpenGL. Устройство класса виджета см. на рис. 3.

Рисунок 2. Основные функции для работы с OpenGL

Рисунок 3. Устройство класса виджета GLScene

Метод *initializeGL* вызывается до первого вызова методов *resizeGL* и *paintGL*. В данном методе функции OpenGL привязываются к контексту, а также виджет заполняется определенным цветом. Код данного метода см. в листинге 1.

Листинг 1. Код метода *initializeGL.*

void GLScene::initializeGL()

{

QColor bgc(255, 255, 255);

initializeOpenGLFunctions();

glClearColor(bgc.redF(), bgc.greenF(), bgc.blueF(), bgc.alphaF());

}

Метод *resizeGL* вызывается при изменении размеров отрисовываемой области. Код данного метода см. в листинге 2.

Листинг 2. Код метода  *resizeGL.*

void GLScene::resizeGL(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

Метод *paintGL* вызывается при обновлении виджета. В данном методе реализована отрисовка примитивов. Внутри метода происходит очистка буффера, после чего происходит непосредственно отрисовка заданного примитива (задается в *primitiveMode*). Код данного метода см. в листинге 3.

Листинг 3. Код метода *paintGL*.

void GLScene::paintGL()

{

QColor colors[10] = {

QColor("cyan"), QColor("magenta"), QColor("darkGreen"),

QColor("darkRed"), QColor("darkCyan"), QColor("darkMagenta"),

QColor("green"), QColor("red"), QColor("yellow"), QColor("blue")};

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glPointSize(5.f);

glLineWidth(3.f);

glBegin(primitiveMode);

glColor3d(colors[0].redF(), colors[0].greenF(), colors[0].blueF());

glVertex2d(-0.1f, 0.8f);

glColor3d(colors[1].redF(), colors[1].greenF(), colors[1].blueF());

glVertex2d(-0.4f, 0.6f);

glColor3d(colors[2].redF(), colors[2].greenF(), colors[2].blueF());

glVertex2d(-0.7f, 0.05f);

glColor3d(colors[3].redF(), colors[3].greenF(), colors[3].blueF());

glVertex2d(-0.6f, -0.35f);

glColor3d(colors[4].redF(), colors[4].greenF(), colors[4].blueF());

glVertex2d(-0.35f, -0.7f);

glColor3d(colors[5].redF(), colors[5].greenF(), colors[5].blueF());

glVertex2d(0.2f, -0.8f);

glColor3d(colors[6].redF(), colors[6].greenF(), colors[6].blueF());

glVertex2d(0.5f, -0.75f);

glColor3d(colors[7].redF(), colors[7].greenF(), colors[7].blueF());

glVertex2d(0.75f, -0.2f);

glColor3d(colors[8].redF(), colors[8].greenF(), colors[8].blueF());

glVertex2d(0.6f, 0.25f);

glColor3d(colors[9].redF(), colors[9].greenF(), colors[9].blueF());

glVertex2d(0.25f, 0.7f);

glEnd();

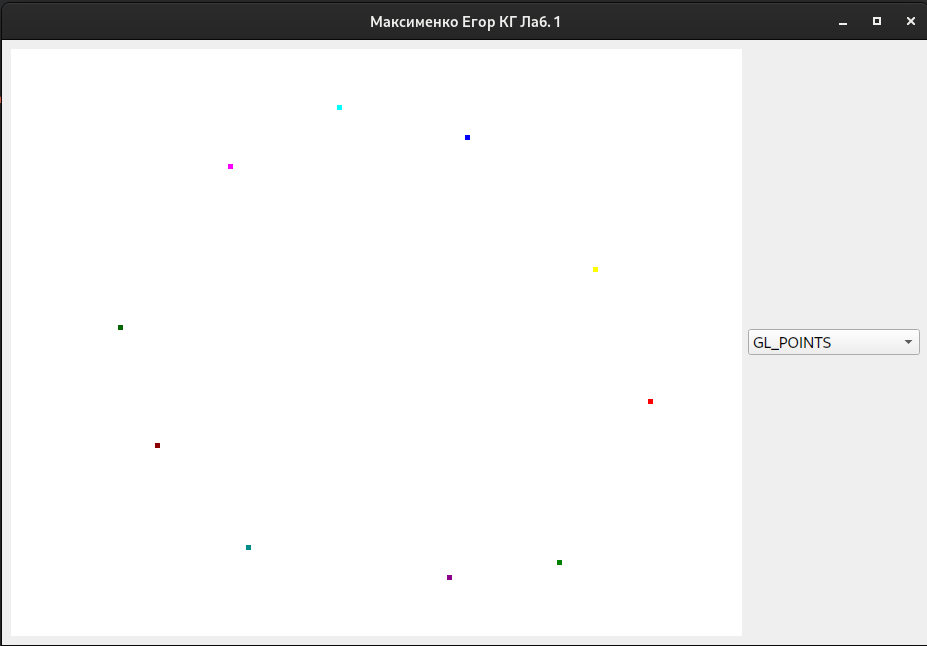
}

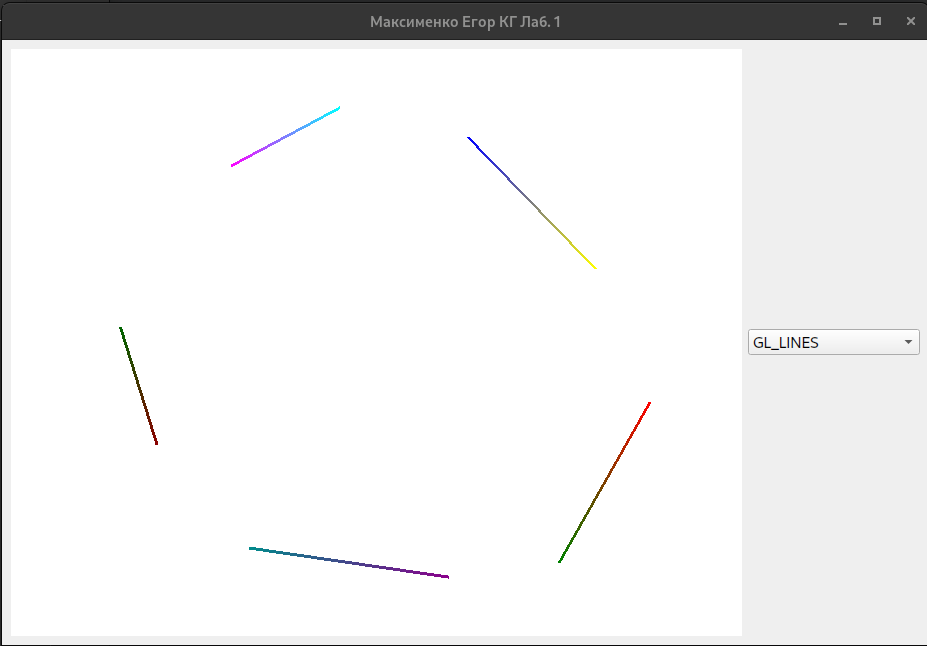
В методе *paintGL* заданы несколько (10) точек, которые служат каркасом для различных примитивов, а также ширина линий и размер точек для примитивов.

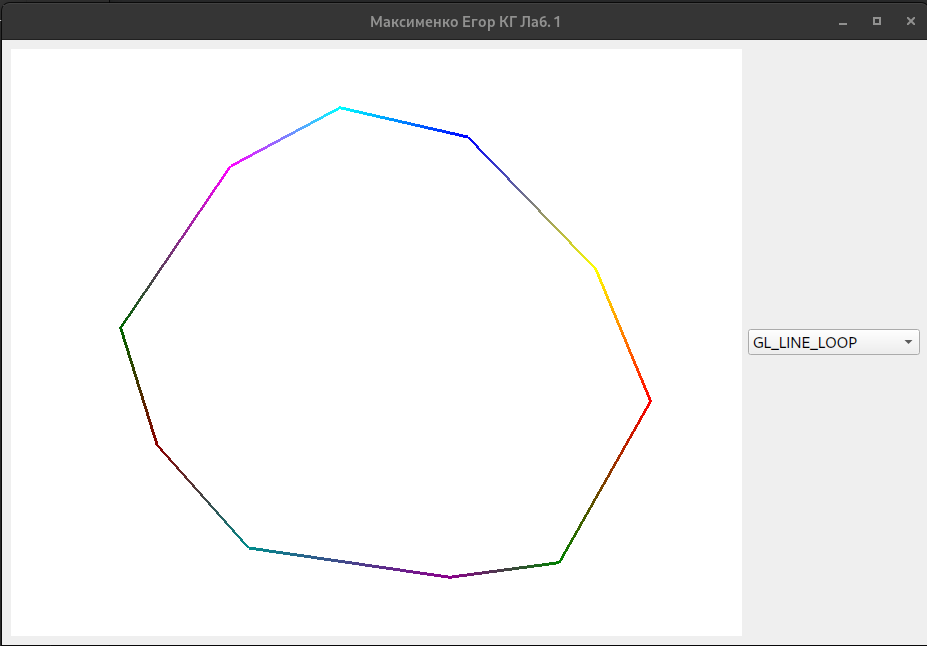
Пользователь может выбрать тип примитива в интерфейсе, после чего виджет GLScene будет перерисован (будет вызван метод *paintGL*) и будет отображен выбранный примитив.

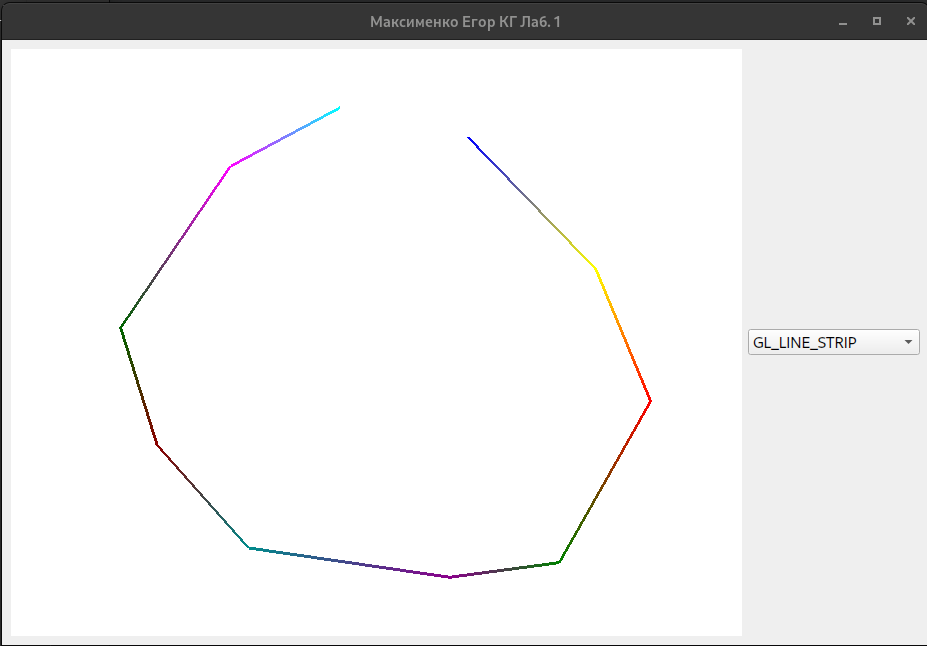
**Тестирование.**

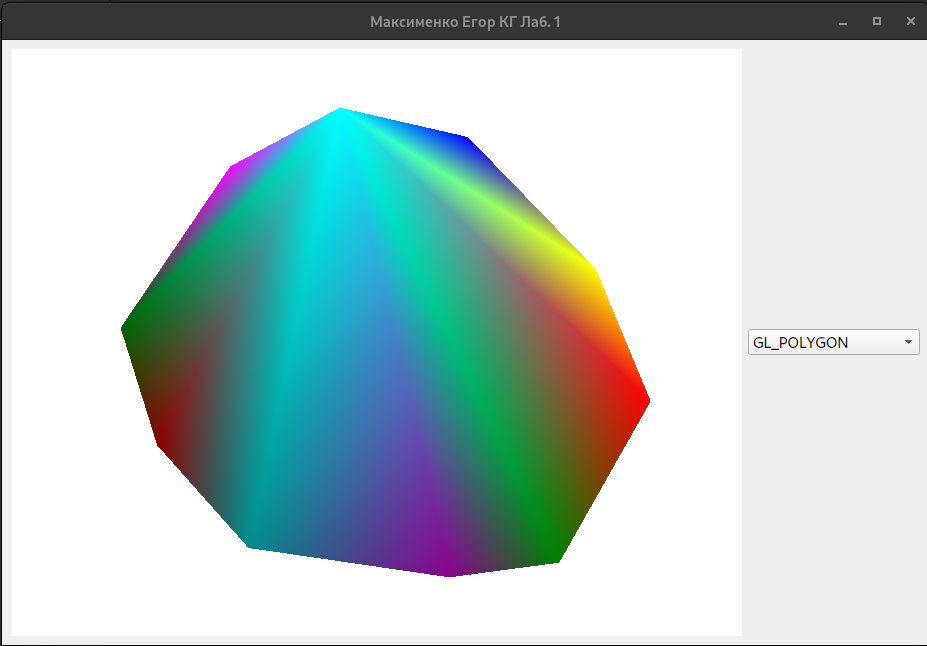
Программа была протестирована для различных типов примитивов (см. Задание). По результатам тестирования были получены скриншоты работы программы, представленные ниже (см. рис. 4-13).

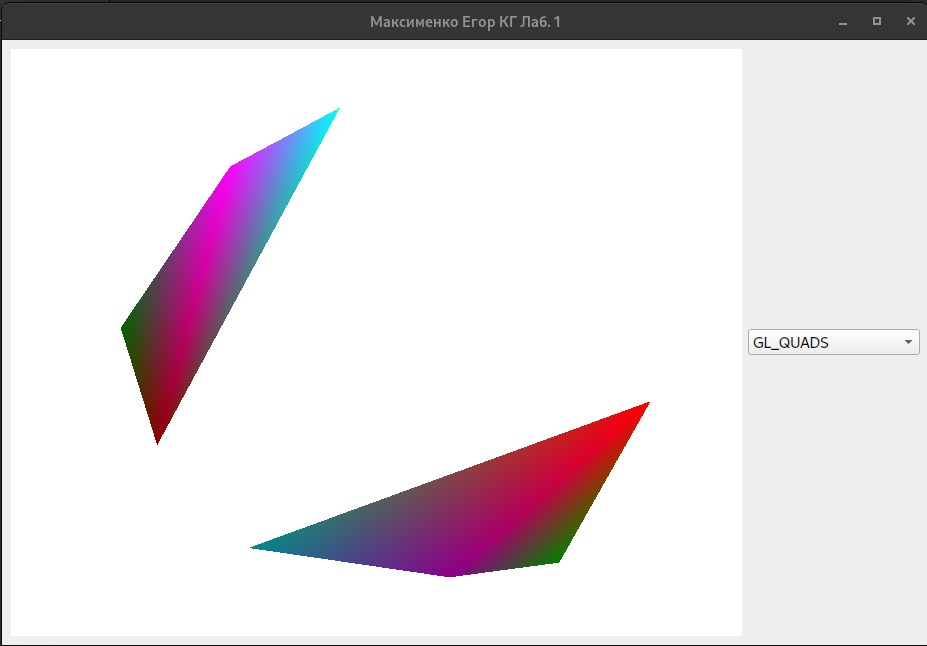
Рисунок 4. Результат запуска для примитива GL\_POINTS

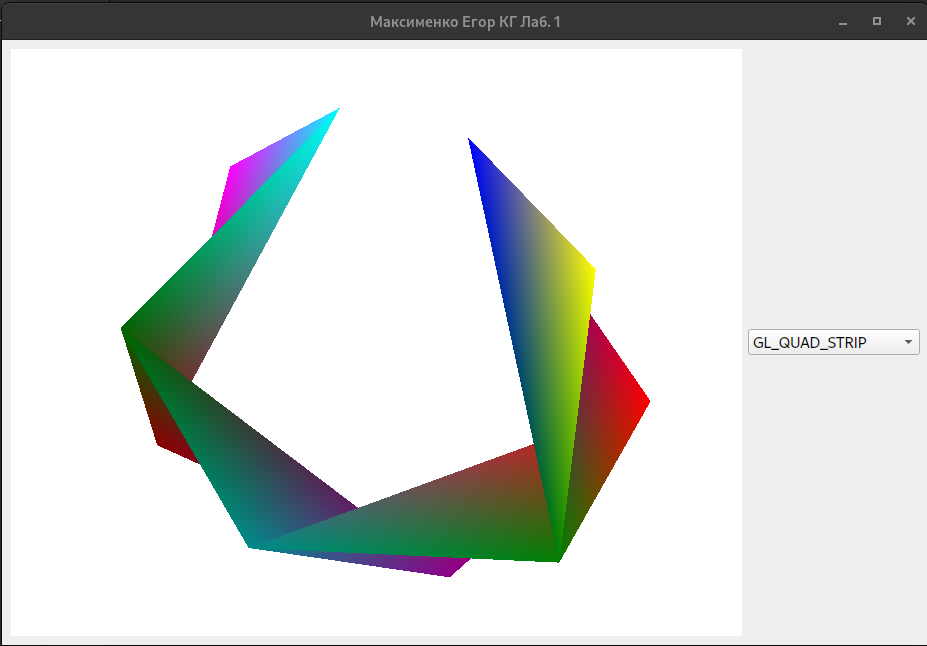
Рисунок 5. Результат запуска для примитива GL\_LINES

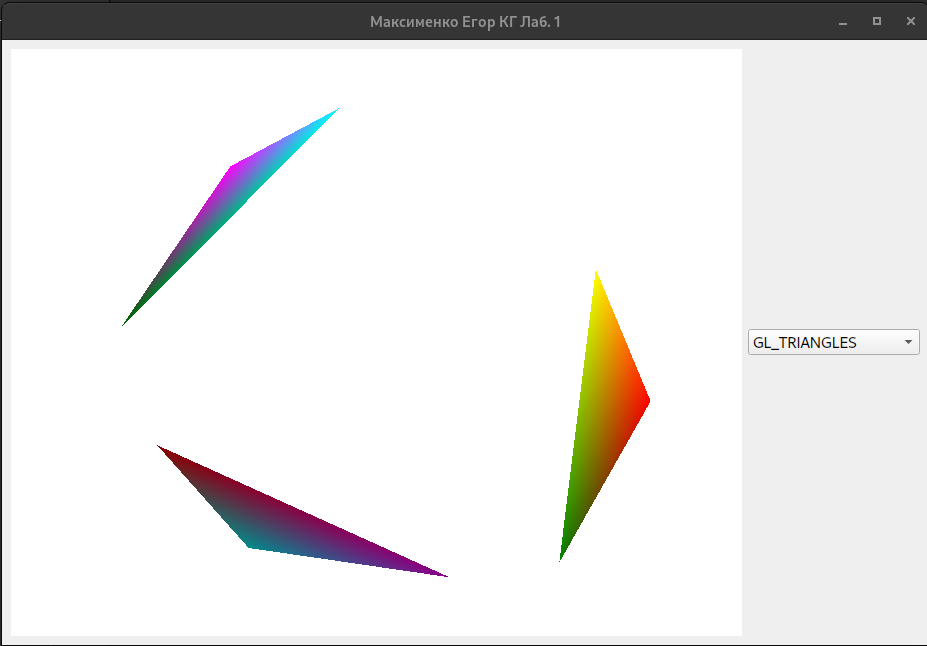
Рисунок 6. Результат запуска для примитива GL\_LINE\_LOOP

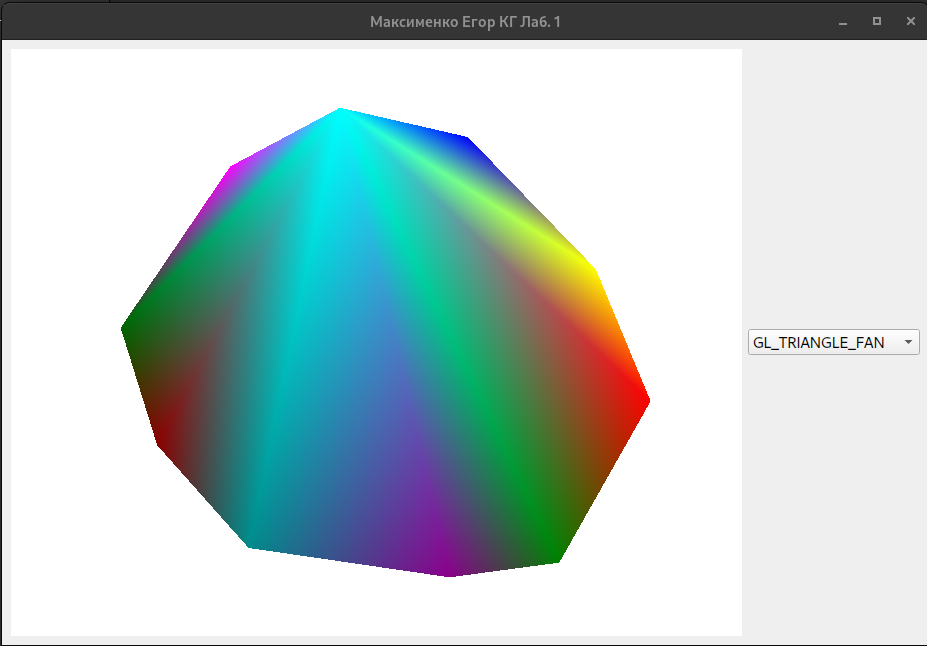
Рисунок 7. Результат запуска для примитива GL\_LINE\_STRIP

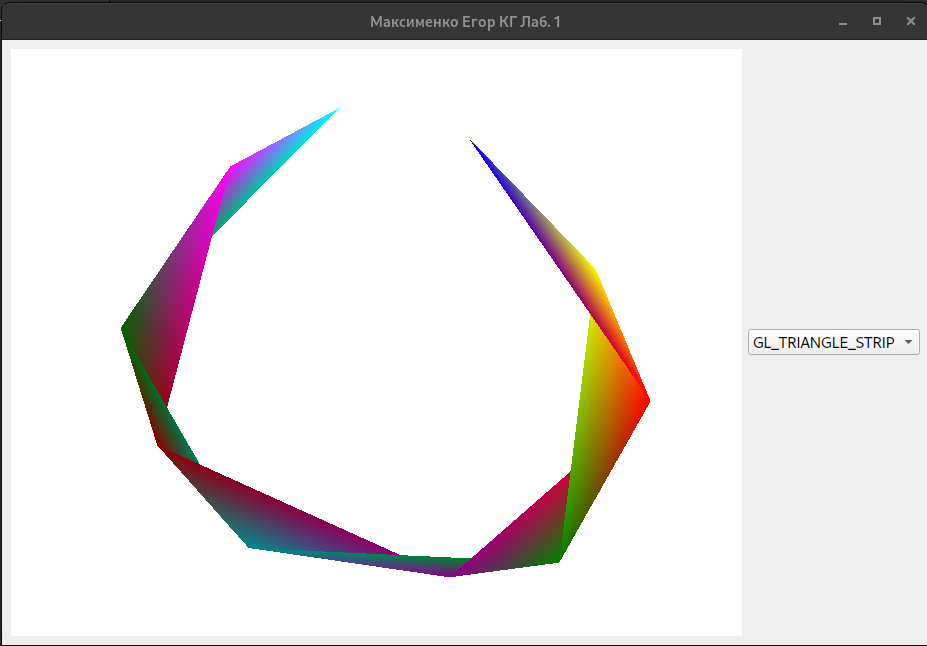
Рисунок 8. Результат запуска для примитива GL\_POLYGON

Рисунок 9. Результат запуска для примитива GL\_QUADS

Рисунок 10. Результат запуска для примитива GL\_QUAD\_STRIP

Рисунок 11. Результат запуска для примитива GL\_TRIANGLES

Рисунок 12. Результат запуска для примитива GL\_TRIANGLE\_FAN

Рисунок 13. Результат запуска для примитива GL\_TRIANGLE\_STRIP

**Выводы.**

В результате выполнения лабораторной работы была разработана программа, создающая графические примитивы OpenGL. Программа работает корректно. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.