

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: Прimitives 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.

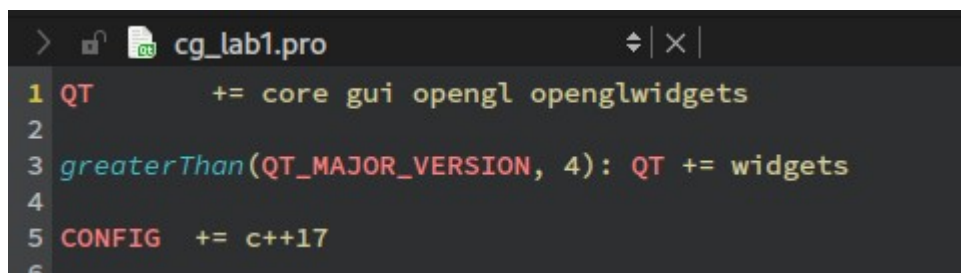
Разработать программу, реализующую представление определенного набора примитивов (4) из имеющихся в OpenGL (GL_POINT, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON).

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).



```
> cg_lab1.pro
1 QT      += core gui opengl openglwidgets
2
3 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
4
5 CONFIG += c++17
6
```

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.

```

13     void setPrimitiveMode(GLenum mode);
14 protected:
15     virtual void initializeGL() override;
16     virtual void resizeGL(int w, int h) override;
17     virtual void paintGL() override;

```

Рисунок 2. Основные функции для работы с OpenGL

```

1  #ifndef GLSCENE_H
2  #define GLSCENE_H
3
4  #include <QtOpenGLWidgets/QOpenGLWidget>
5  #include <QOpenGLFunctions>
6
7  class GLScene: public QOpenGLWidget, public QOpenGLFunctions
8  {
9      Q_OBJECT
10 public:
11     GLScene(QWidget* parent = nullptr);
12
13     void setPrimitiveMode(GLenum mode);
14 protected:
15     virtual void initializeGL() override;
16     virtual void resizeGL(int w, int h) override;
17     virtual void paintGL() override;
18 private:
19     GLenum primitiveMode = GL_POINTS;
20 };
21
22 #endif // GLSCENE_H
23

```

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 буфера, 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).

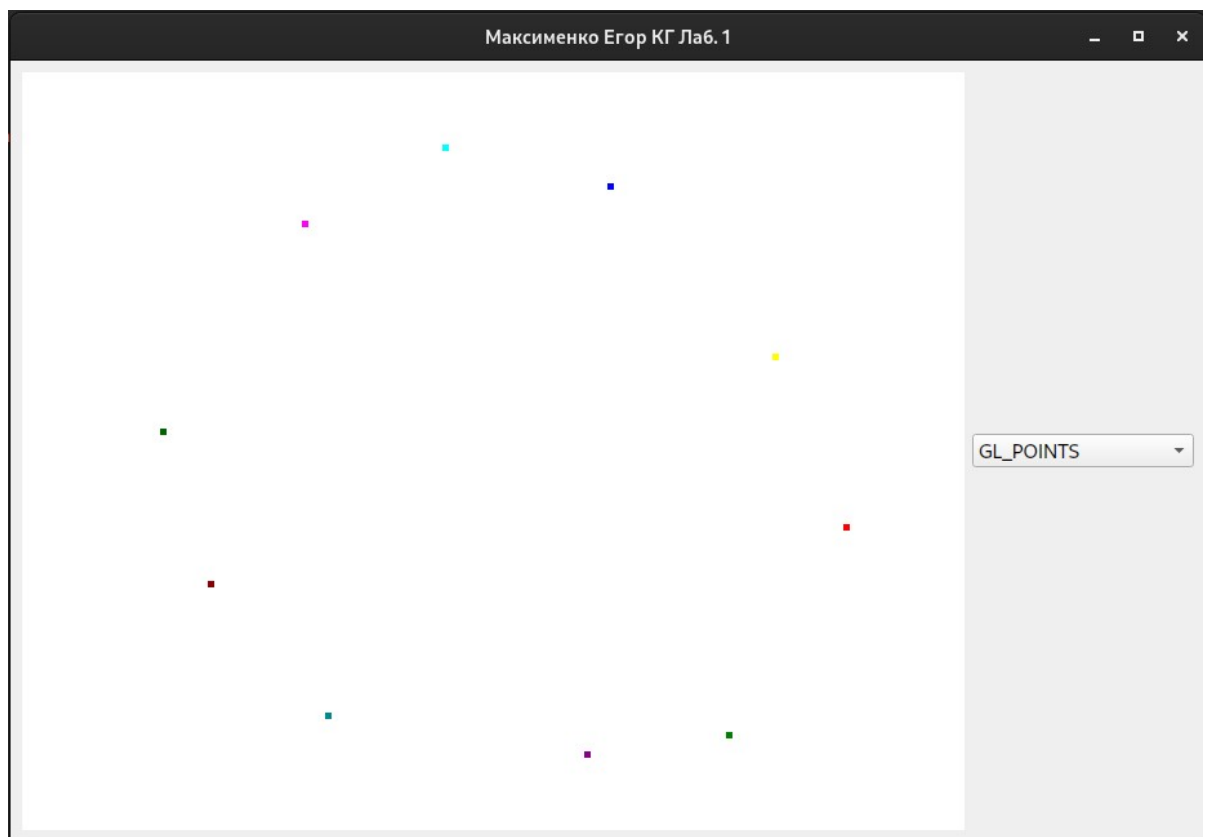


Рисунок 4. Результат запуска для примитива GL_POINTS

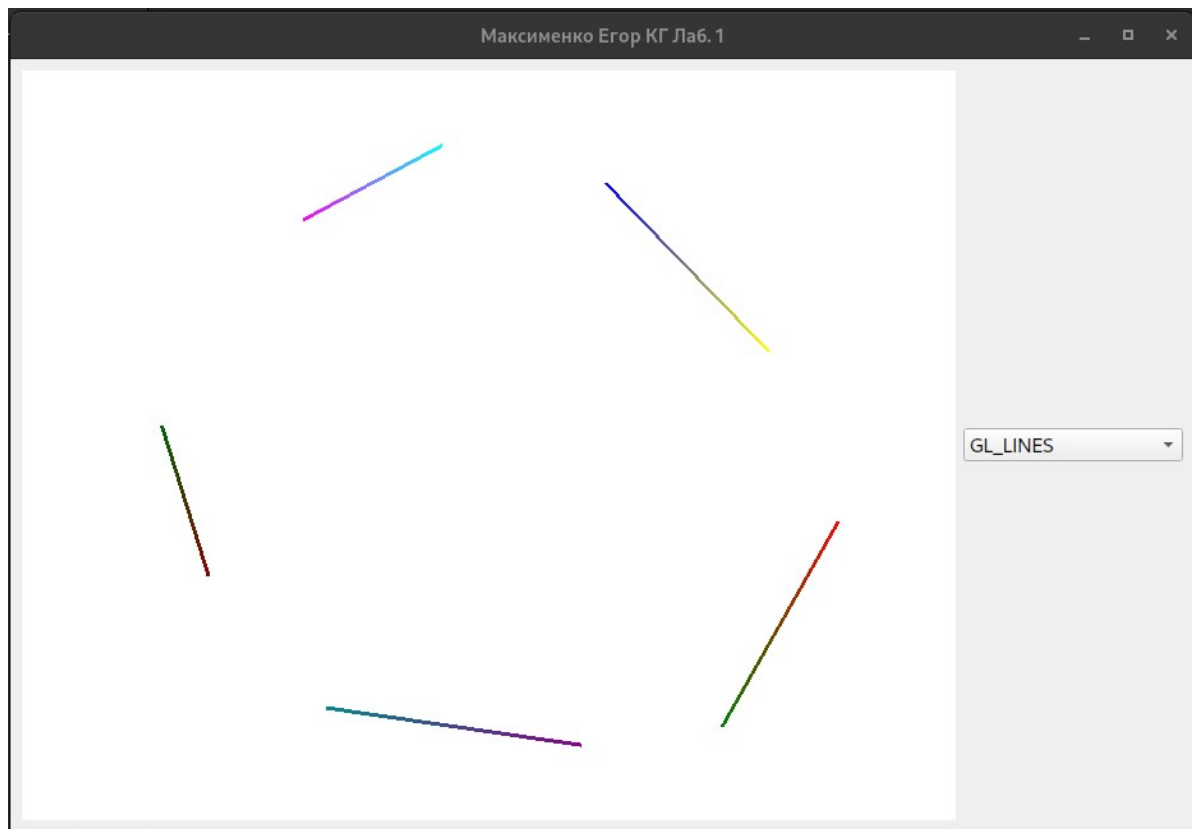


Рисунок 5. Результат запуска для примитива GL_LINES



Рисунок 6. Результат запуска для примитива GL_LINE_LOOP

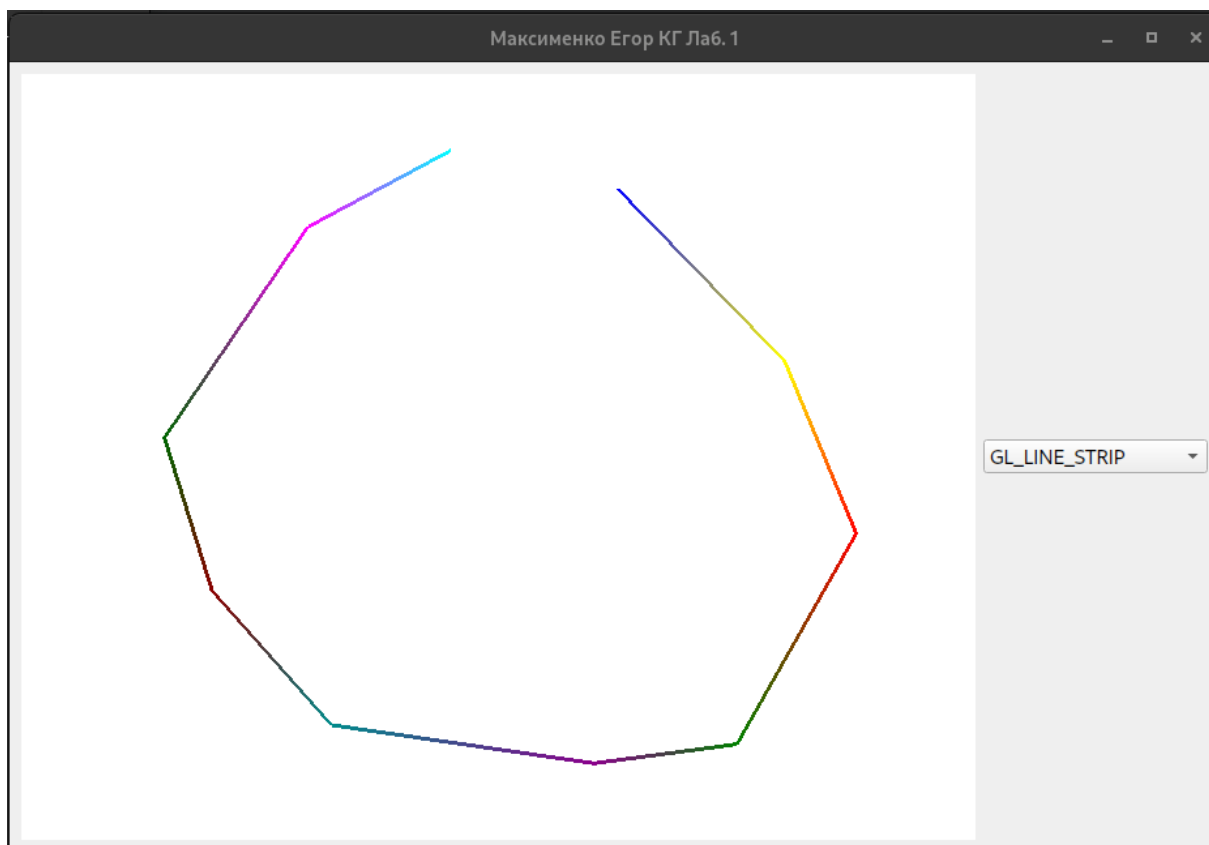


Рисунок 7. Результат запуска для примитива GL_LINE_STRIP

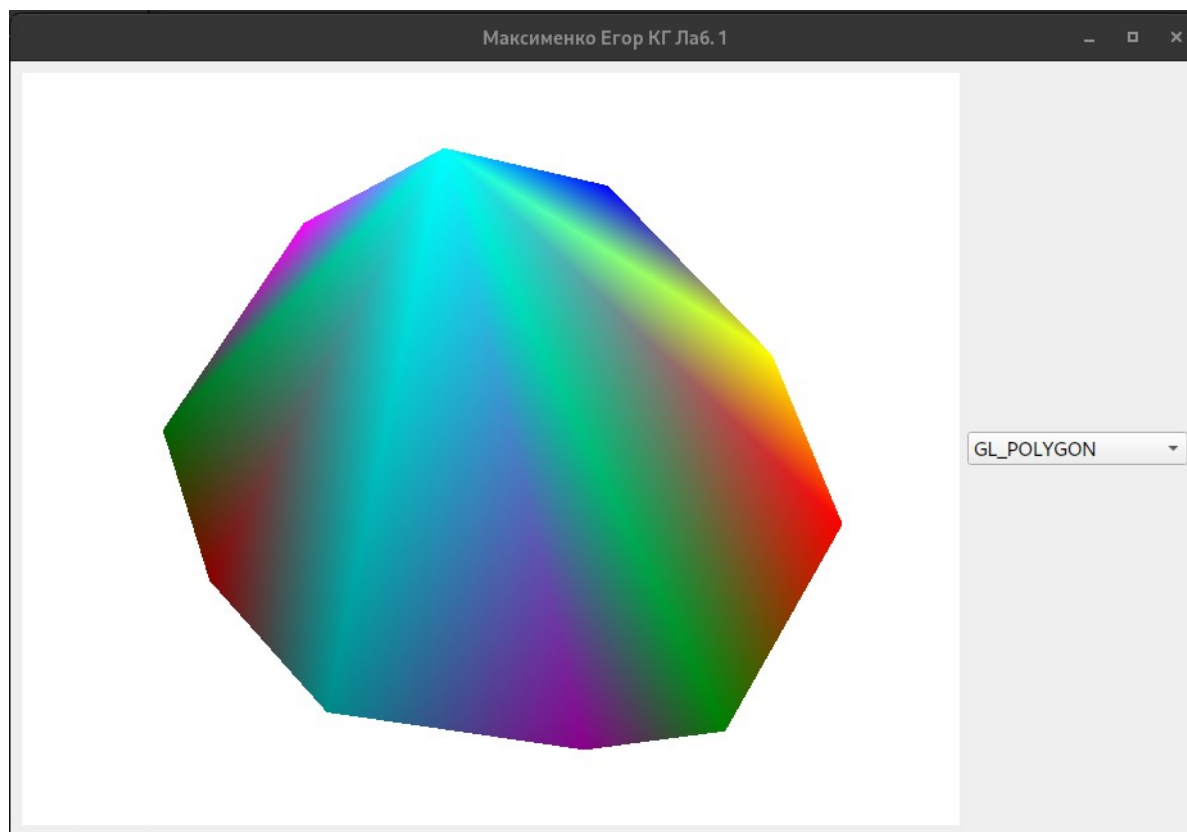


Рисунок 8. Результат запуска для примитива GL_POLYGON

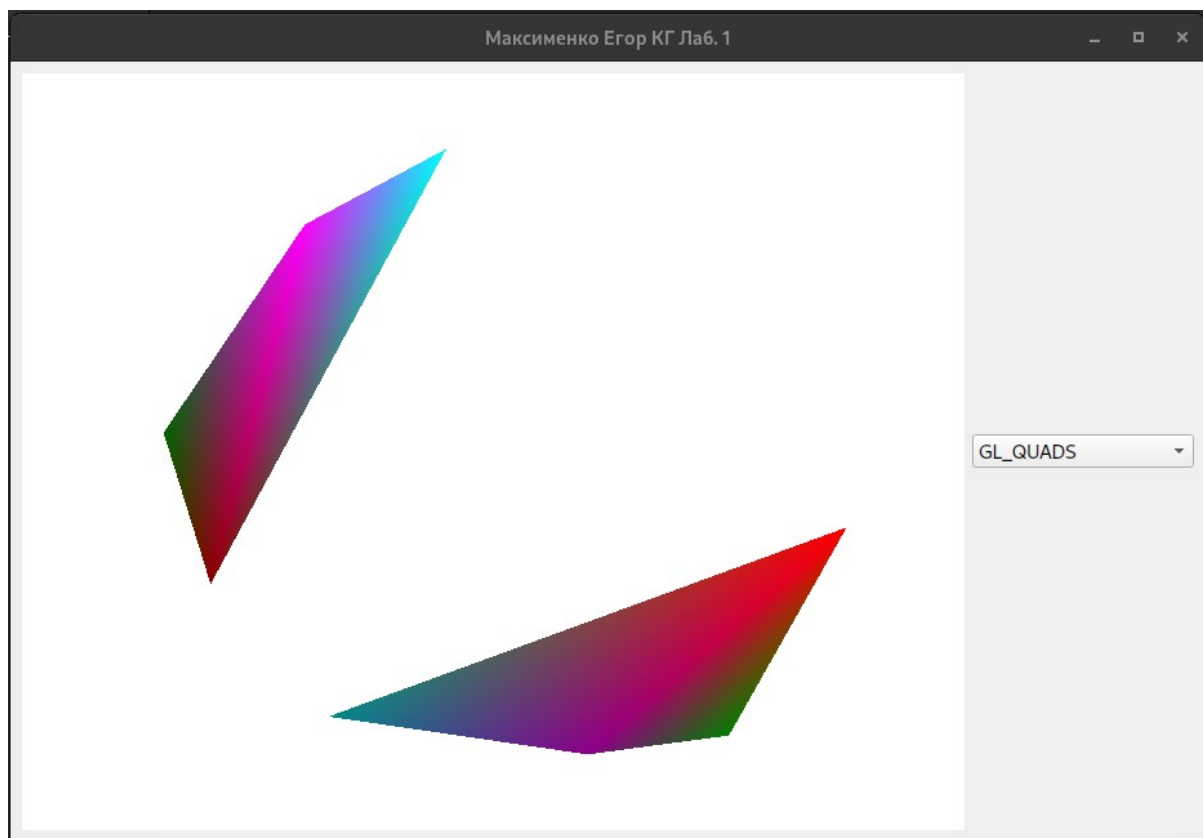


Рисунок 9. Результат запуска для примитива GL_QUADS

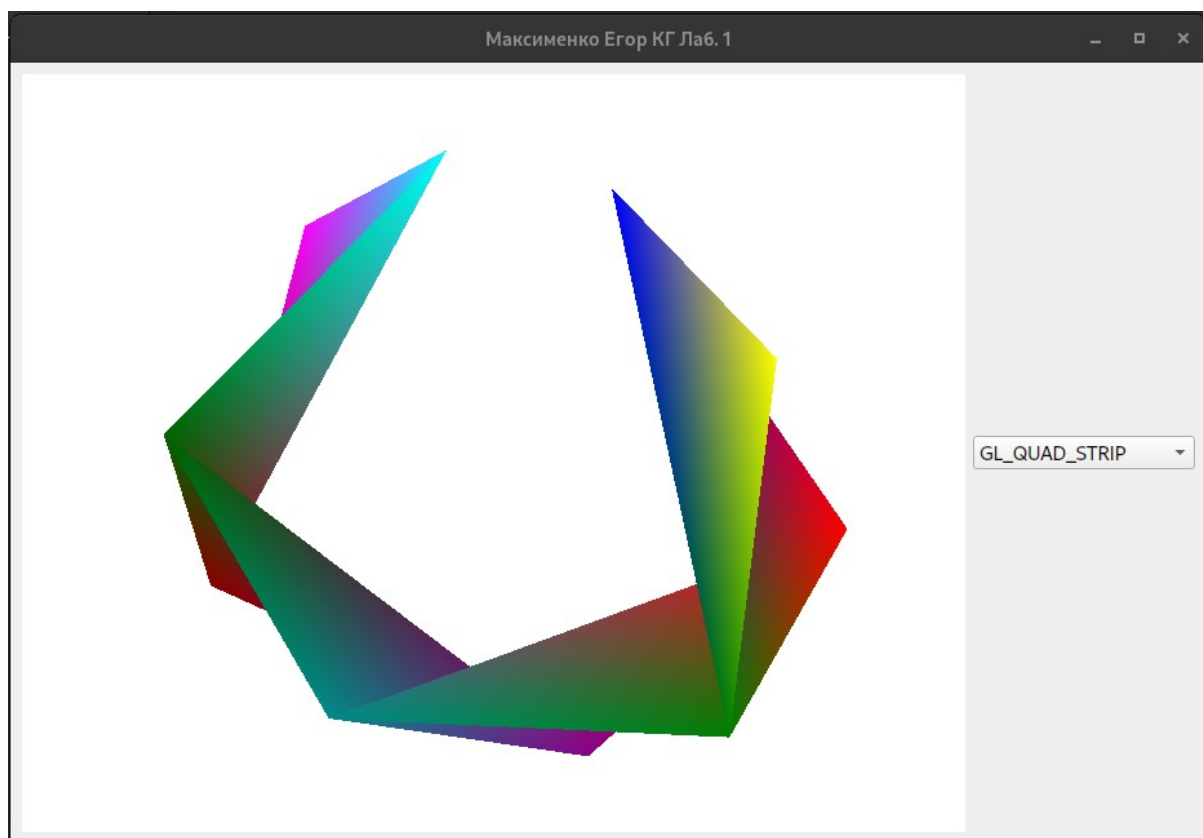


Рисунок 10. Результат запуска для примитива GL_QUAD_STRIP

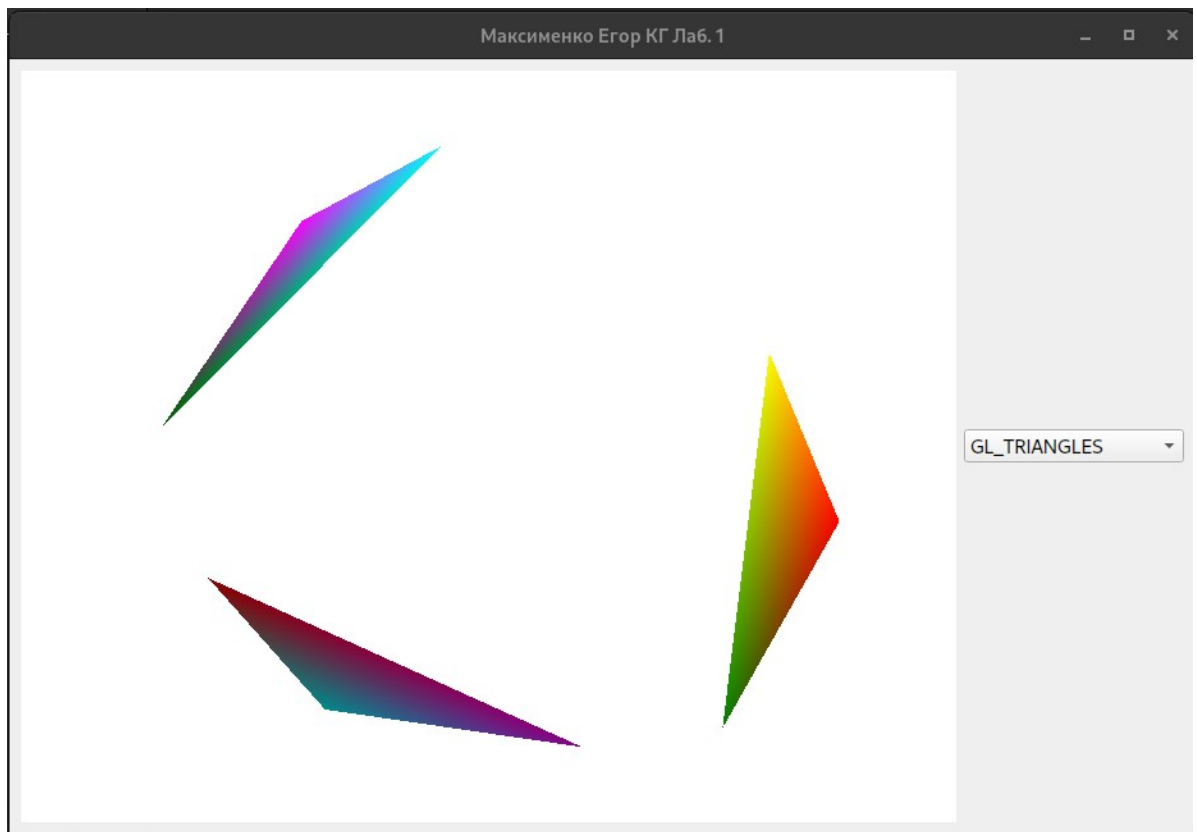


Рисунок 11. Результат запуска для примитива GL_TRIANGLES

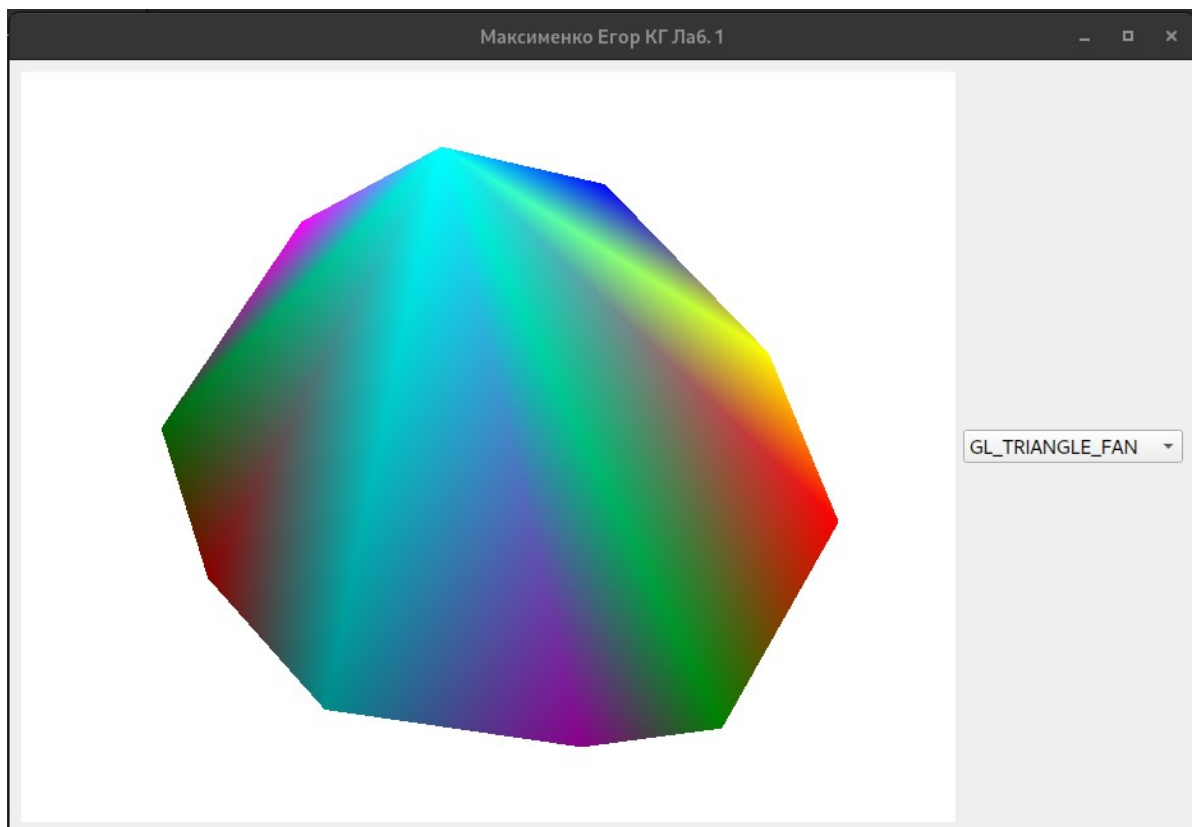


Рисунок 12. Результат запуска для примитива GL_TRIANGLE_FAN

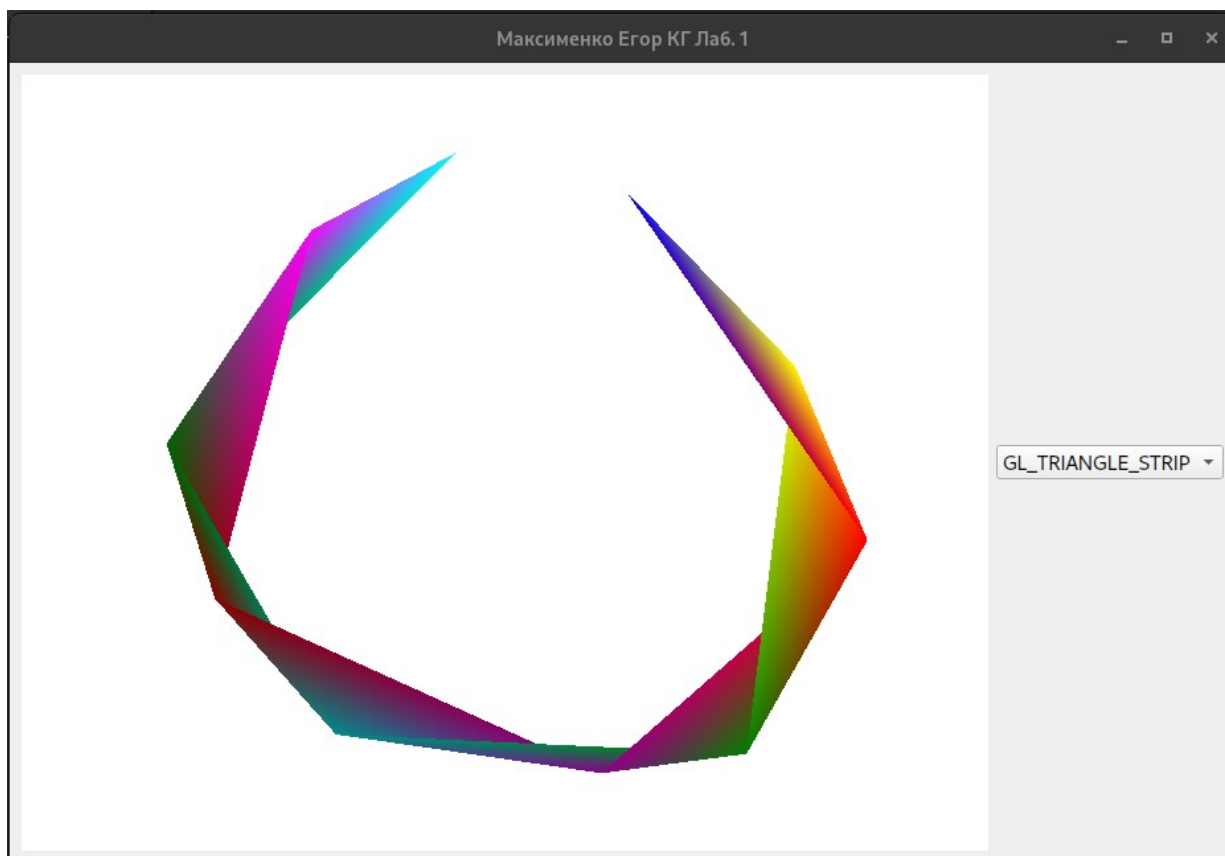


Рисунок 13. Результат запуска для примитива GL_TRIANGLE_STRIP

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.