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. 7383 _____ Власов Р.А.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2020

Задание.

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

Общие сведения.

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).

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, GL_QUAD_STRIP, GL_POLYGON – устаревшие примитивы и в версиях OpenGL выше 3.3 отсутствуют.

Ход работы.

Программа разработана на языке программирования Python. Графический интерфейс реализован с помощью библиотеки Qt. Установка библиотеки PyOpenGL (Python OpenGL) осуществляется с помощью команды “pip install PyOpenGL”. Подключение библиотеки в проект осуществляется при помощи следующего кода:

```
from OpenGL.GL import *
from OpenGL.GLU import *
```

Для успешного отображения примитивов реализован класс OpenGLView. В качестве родительского класса взят QGLWidget. Перепределены методы initializeGL, resizeGL и paintGL, отвечающие за инциализацию, изменение размера виджета и рисование изображения.

За рисование изображений в различных режимах отвечает код:

```
modes = {                                   # Список доступных режимов
    0: GL_POINTS,
    1: GL_LINES,
```

```
    2: GL_LINE_STRIP,
    3: GL_LINE_LOOP,
    4: GL_TRIANGLES,
    5: GL_TRIANGLE_STRIP,
    6: GL_TRIANGLE_FAN,
    7: GL_QUADS,
    8: GL_QUAD_STRIP,
    9: GL_POLYGON
}
colors = {                                  # Набор цветов
    0: (0, 0, 1, 0.2),
    1: (0, 1, 1, 0.3),
    2: (1, 0, 1, 0.4),
    3: (1, 0, 0, 0.5),
    4: (0, 1, 0, 0.6),
    5: (0.5, 0.5, 0, 0.7),
    6: (0, 0, 0.5, 0.8),
    7: (0, 0.5, 0.5, 1.0)
}
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
glLoadIdentity()
glTranslatef(0.0, 0.0, -4.0)
glBegin(modes.get(self.mode, GL_POINTS)) # Выбор нужного режима из
                                         # доступных
posx = 0
posy = 0
radius = 1
for i in range(self.sides):
    glColor4dv(colors.get(i % 7))
    x = radius * cos(i * 2 * pi / self.sides) + posx
    y = radius * sin(i * 2 * pi / self.sides) + posy
    glVertex3f(x, y, 1)
glEnd()
glFlush()
```

Тестирование.

Программа протестирована в операционной системе Ubuntu 19.04.

Результаты тестирования представлены на рис. 1-10.

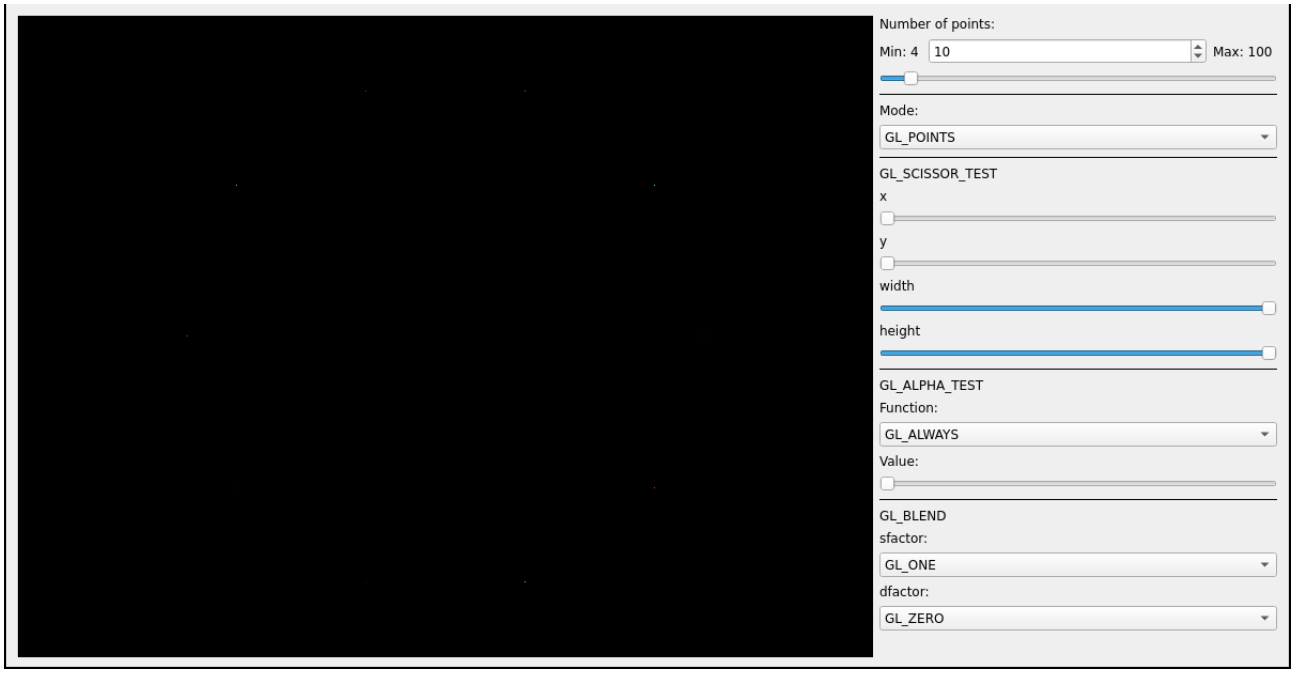

Рисунок 1 – Режим GL_POINTS

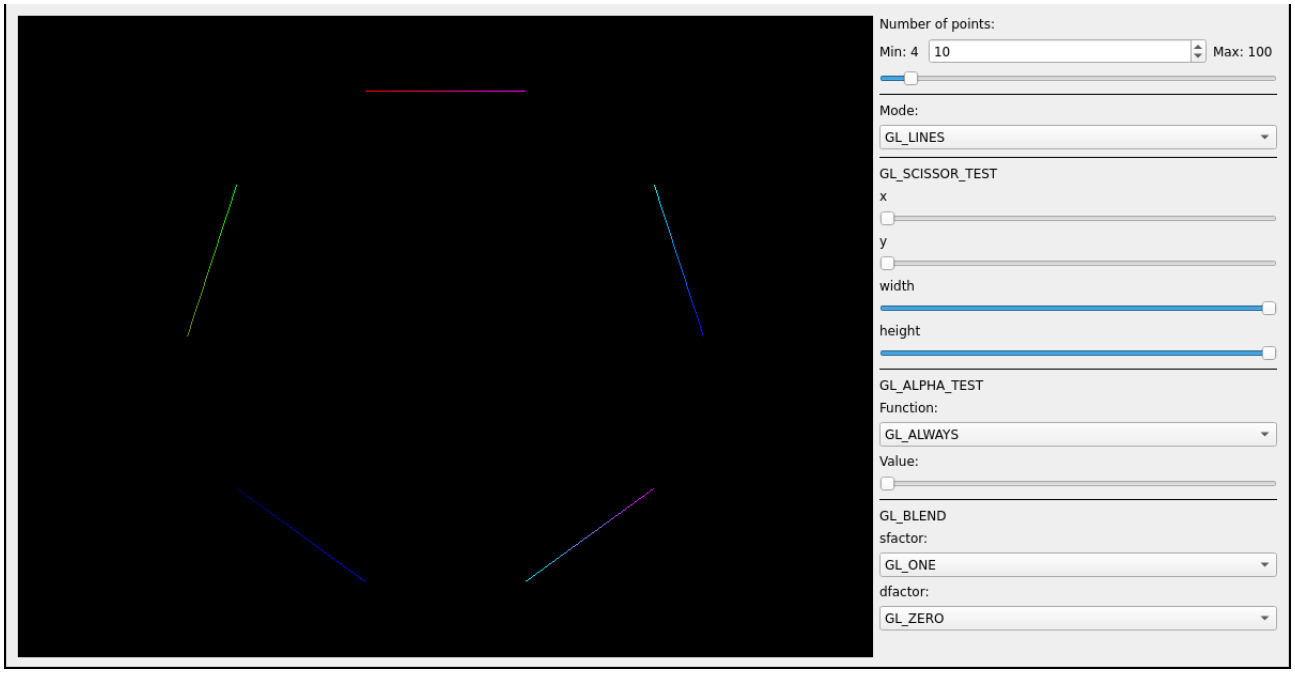

Рисунок 2 – Режим GL_LINES

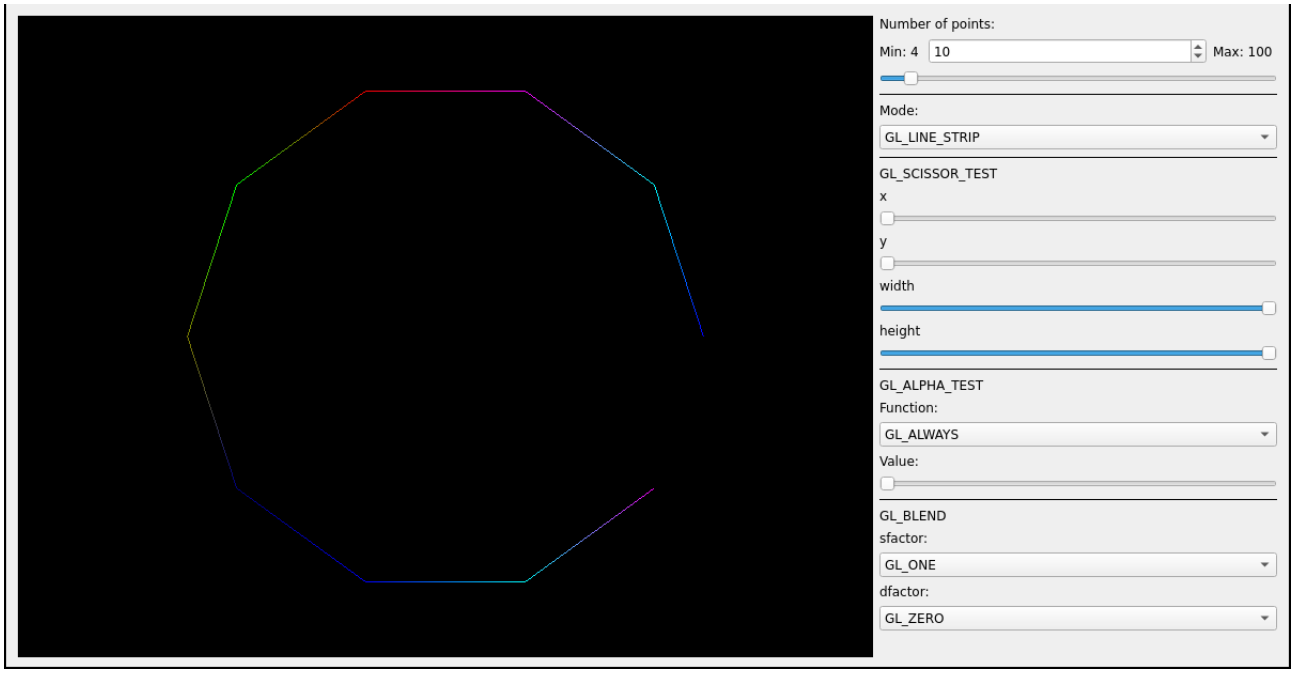

Рисунок 3 – Режим GL_LINE_STRIP

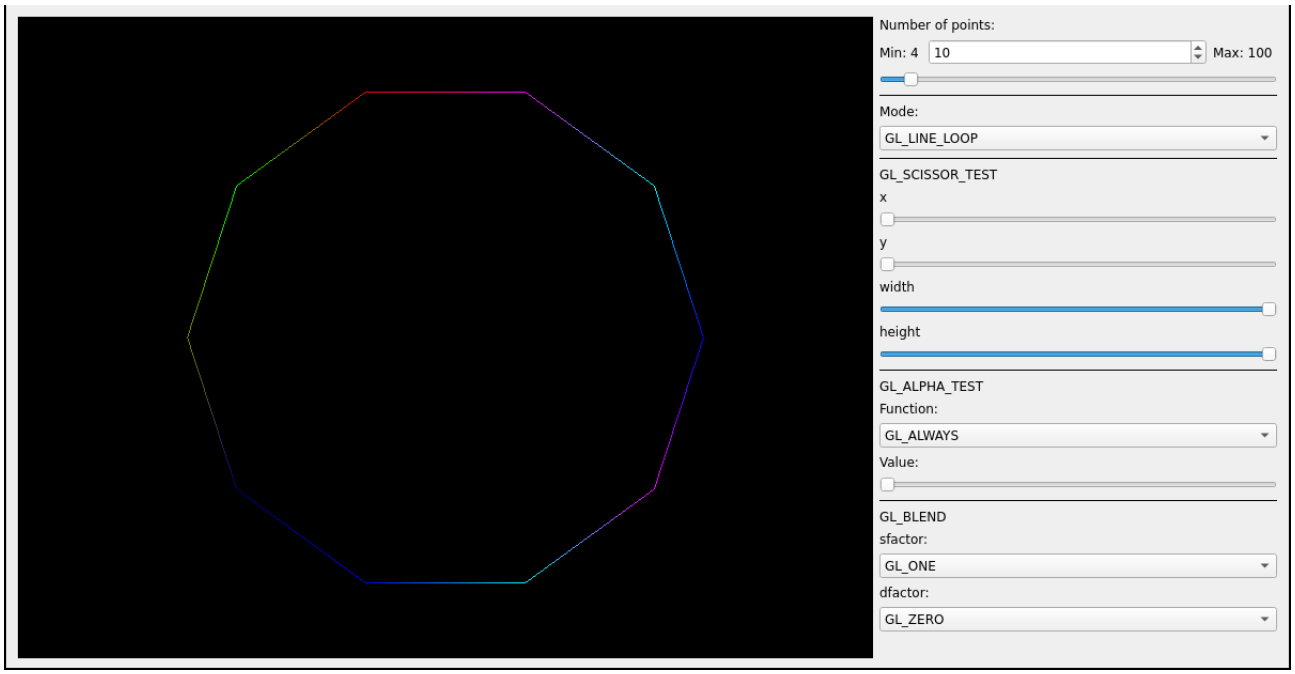

Рисунок 4 – Режим GL_LINE_LOOP

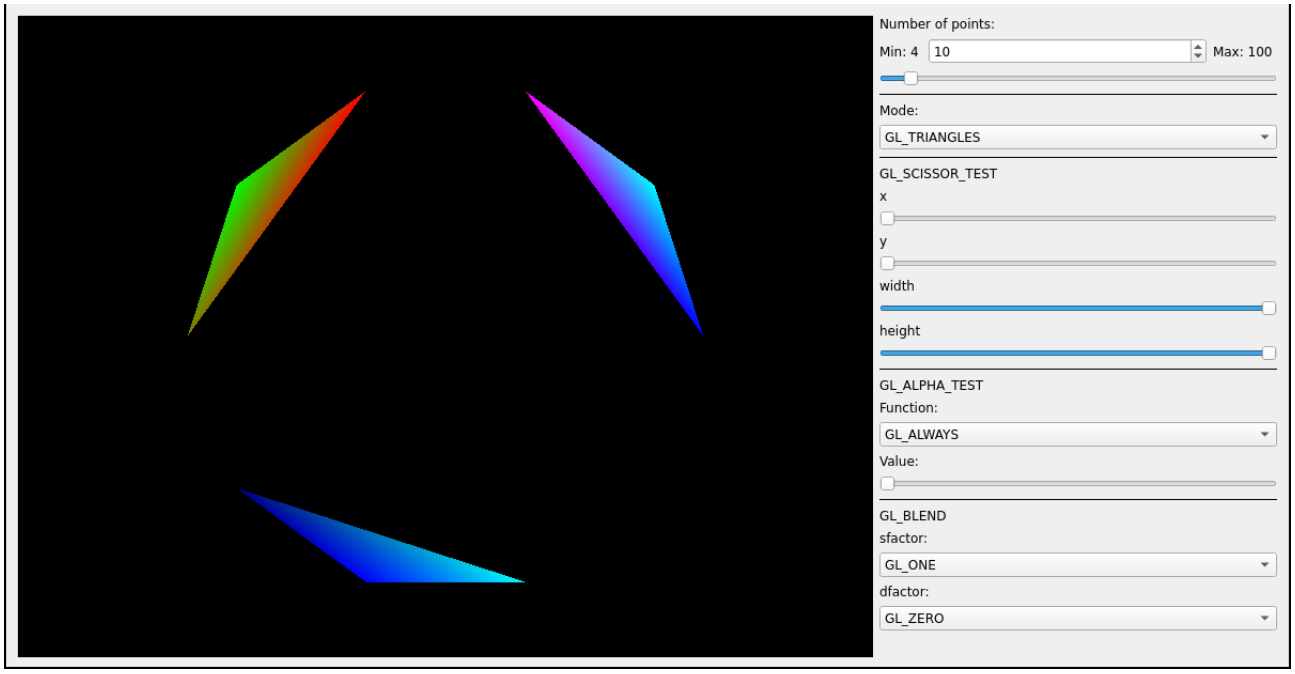

Рисунок 5 – Режим GL_TRIANGLES

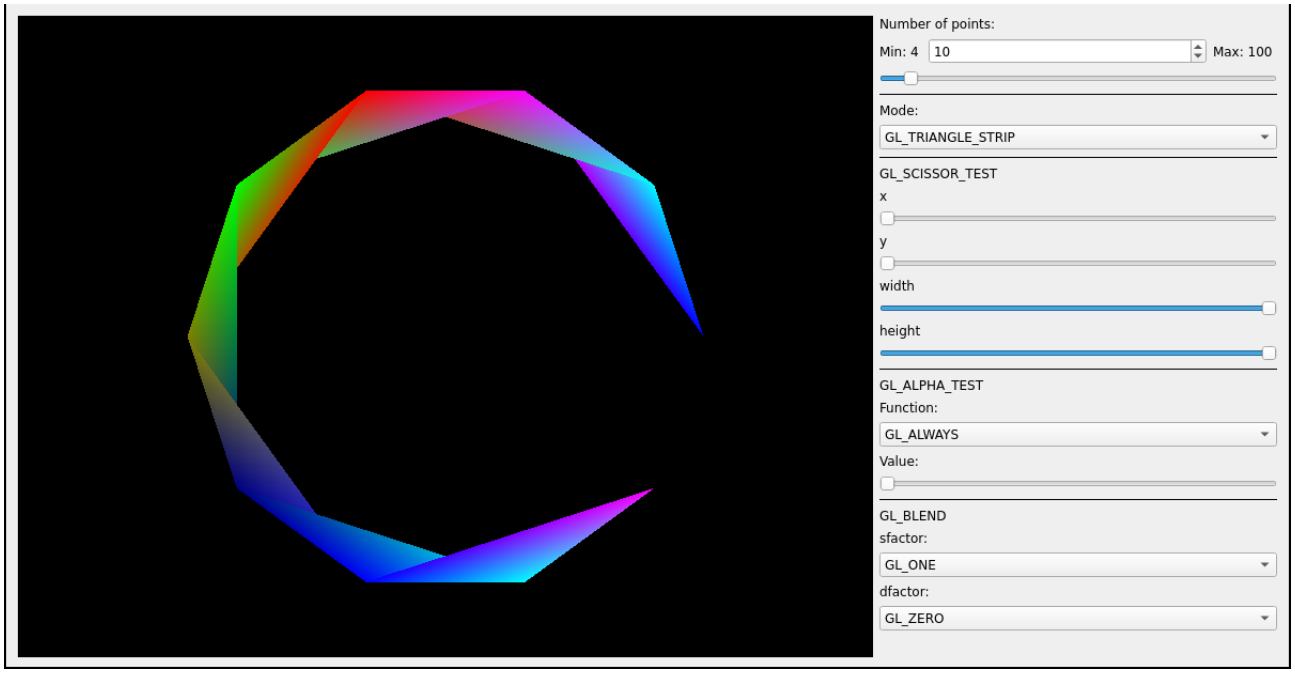

Рисунок 6 – Режим GL_TRIANGLE_STRIP

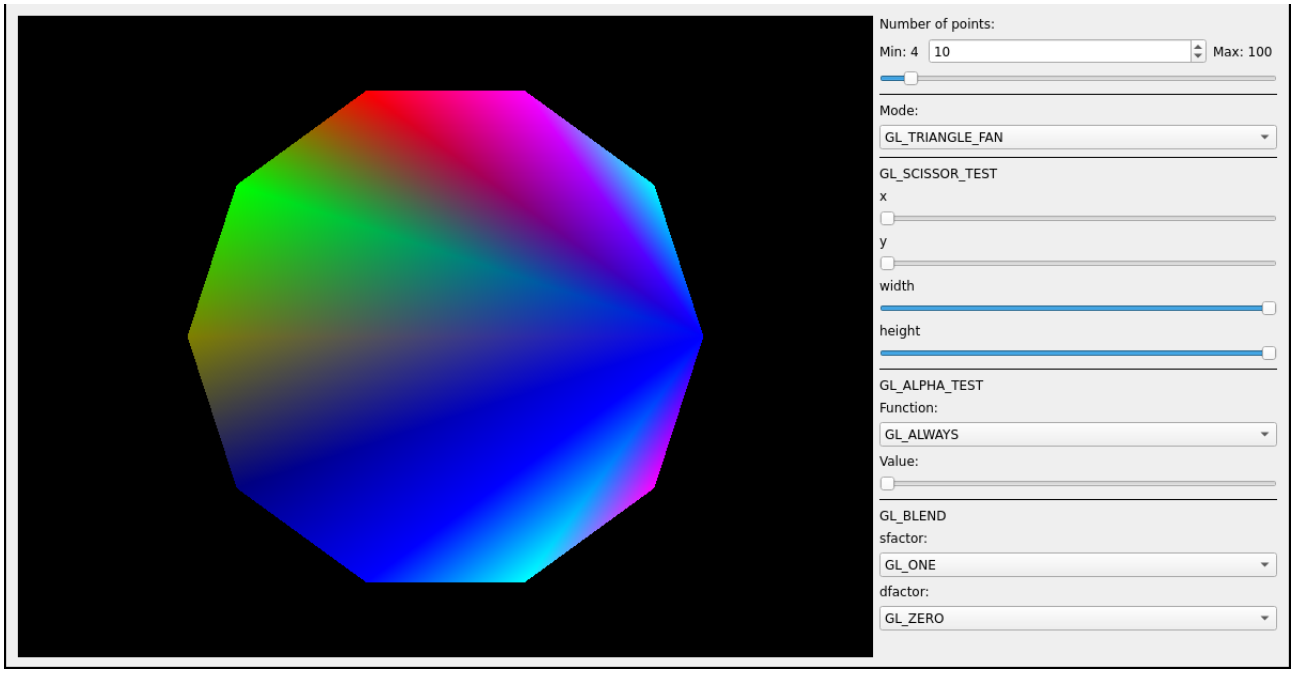

Рисунок 7 – Режим GL_TRIANGLE_FAN

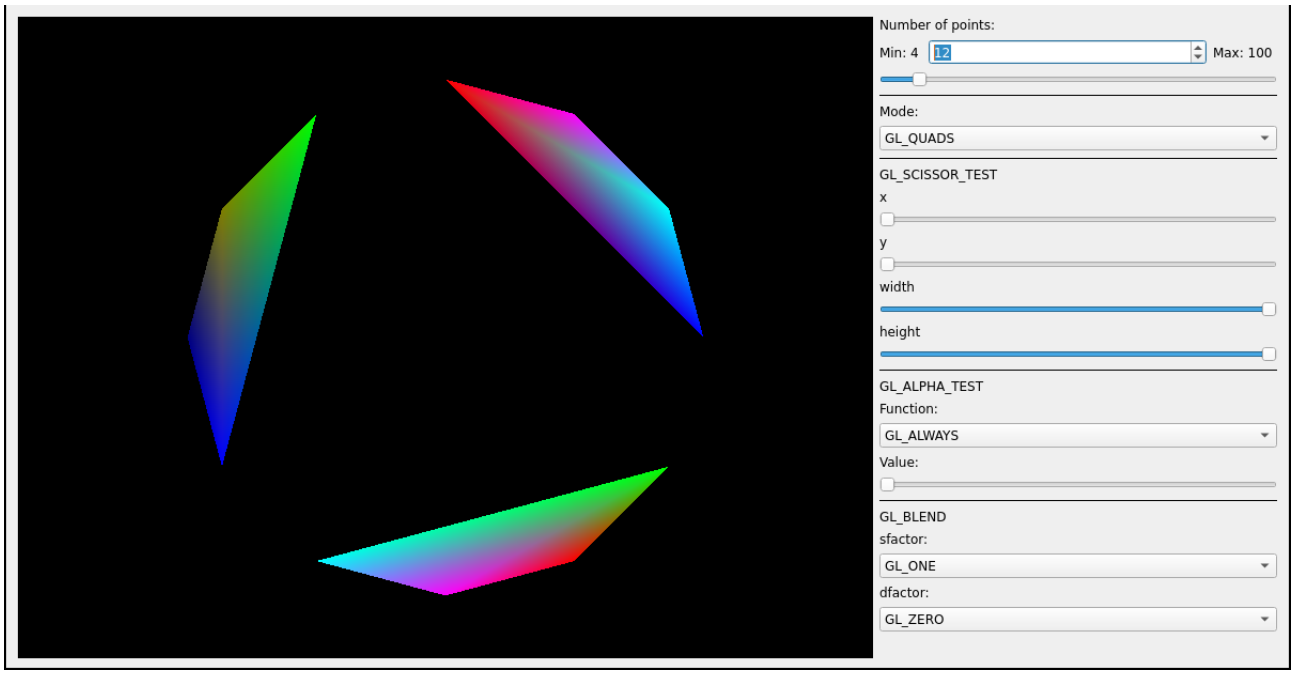

Рисунок 8 – Режим GL_QUADS

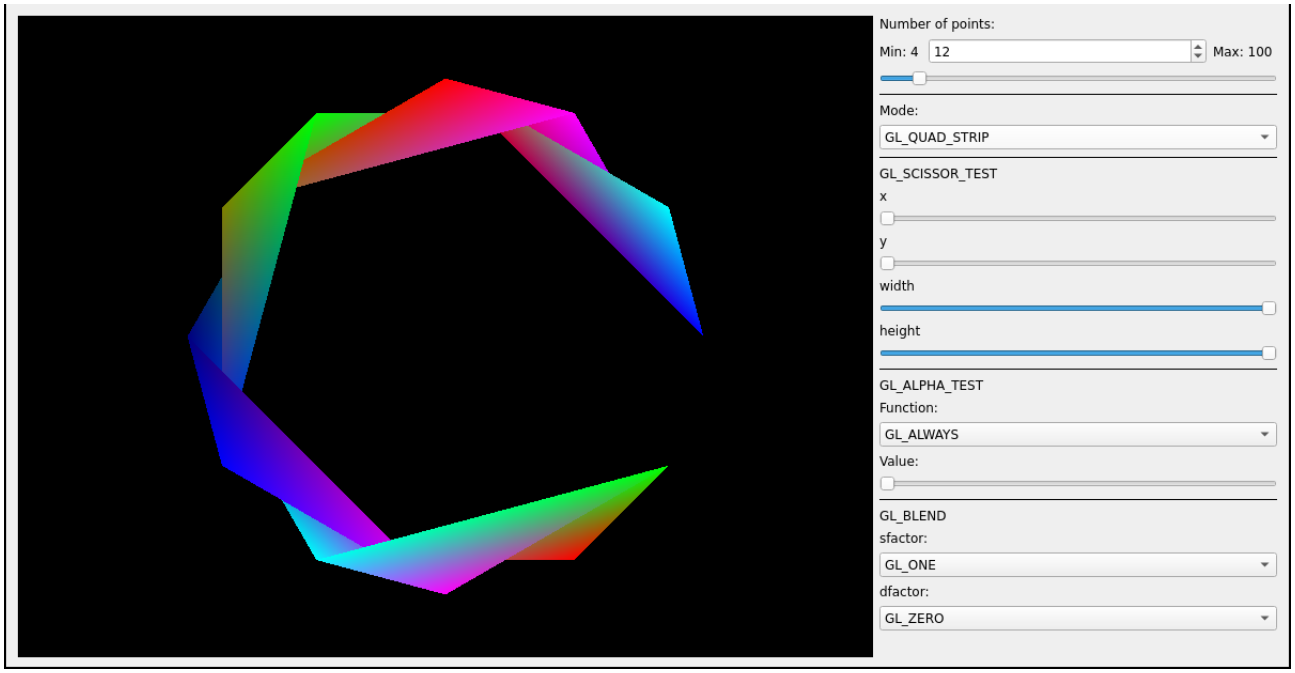

Рисунок 9 – Режим GL_QUAD_STRIP

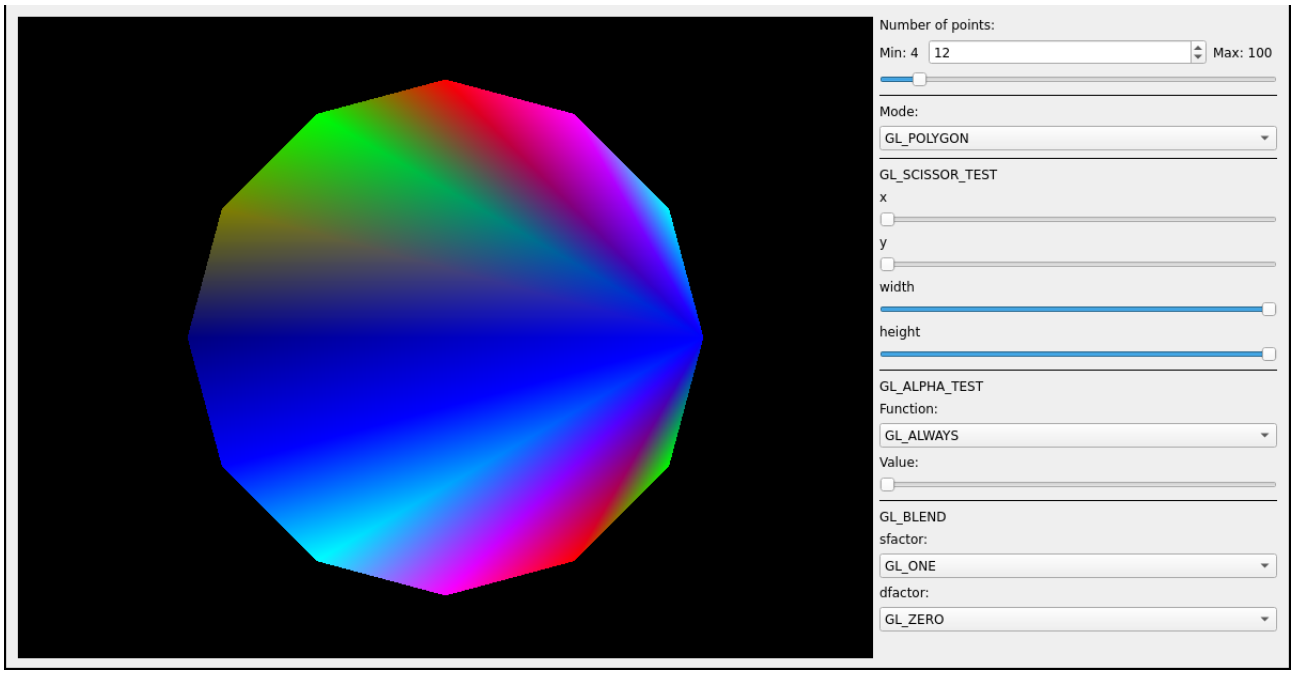

Рисунок 10 – Режим GL_POLYGON

Вывод.

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