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. 6381 _____ Фиалковский М.С.

Студент гр. 6381 _____ Афийчук И.И.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2019

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.

Сборка и компиляция программы осуществляется с помощью утилиты cmake и компилятора gcc из пакета MinGW. Графический интерфейс выполнен с помощью библиотеки FLTK. В качестве обертки над библиотекой OpenGL, используется GLEW.

При запуске программы создаётся окно AppWindow – базовое окно приложения, унаследованное от Fl_Window. Внутрь него методом композиции вложены окно GlSubWin (в котором будет отображаться вся выводимая графики), унаследованное от Fl_Gl_Window, и меню конфигурации. Это окно из библиотеки FLTK создано специально для работы с pure OpenGL calls.

Создание требуемого для работы контекста передаётся внутренним механизмам библиотеки. Для вызова команд OpenGL требуется переопределить виртуальный метод draw() в GlSubWin и делать это внутри него.

Передача конфигурационных параметров изображения реализовано с помощью вспомогательного класса State. Для каждого отдельного изображения (примитива) создаётся свой независимый от других класс наследник State,

который затем помещается в общий контейнер состояний. В созданном классе определяются передаваемые из меню параметры и логика их изменения. Также в нём определяются параметры и наполнение самого меню конфигурации, видимого пользователю.

Отрисовка текущего изображения (примитива) происходит с помощью вызова требуемой функции из вектора функторов. Вызовы этих функторов происходят внутри метода draw окна GlSubWin. Для добавления нового такого функтора следует пронаследоваться от абстрактного класса IPainter и переопределить оператор() с определённым набором передаваемых аргументов и затем добавить его в список функторов в правильном порядке.

Точки для дальнейшей отрисовки изображений не хранятся в отдельном месте, а создаются генератором случайных чисел каждый раз при обновлении какого-либо параметра изображения или при переключении этих самых изображений.

Во всех примитивах используется похожая схема рисования:

```
glBegin(GL_НАЗВАНИЕ_ПРИМИТИВА);
# цикл по количеству точек
	applyColor(…); // вызов функции glColor(r,g,b,alpha) для установки цвета
	… // генерация x и y координаты точки по определённому для примитива алгоритму
	glVertex2f(x, y); // определение точки
glEnd();
```

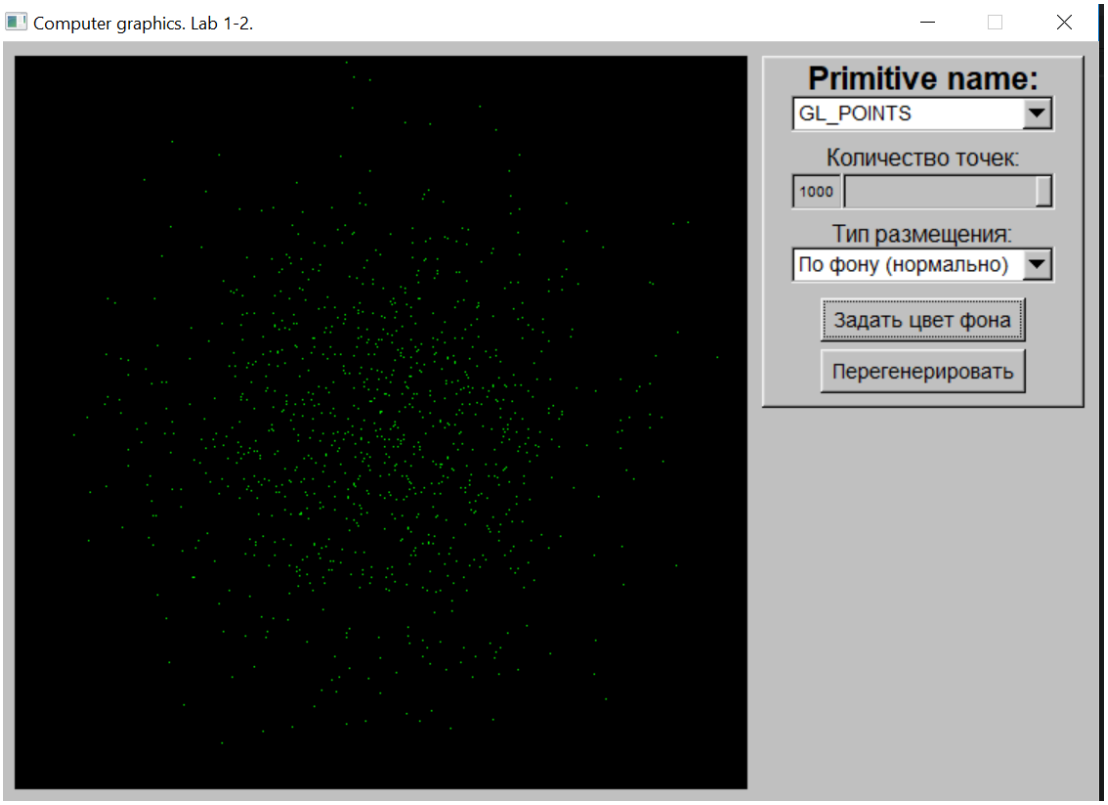
OpenGL распознает точки между «операторными скобками» и строит примитив в зависимости от переданного аргумента в glBegin(…).

Переключаться между примитивами можно с помощью букв.

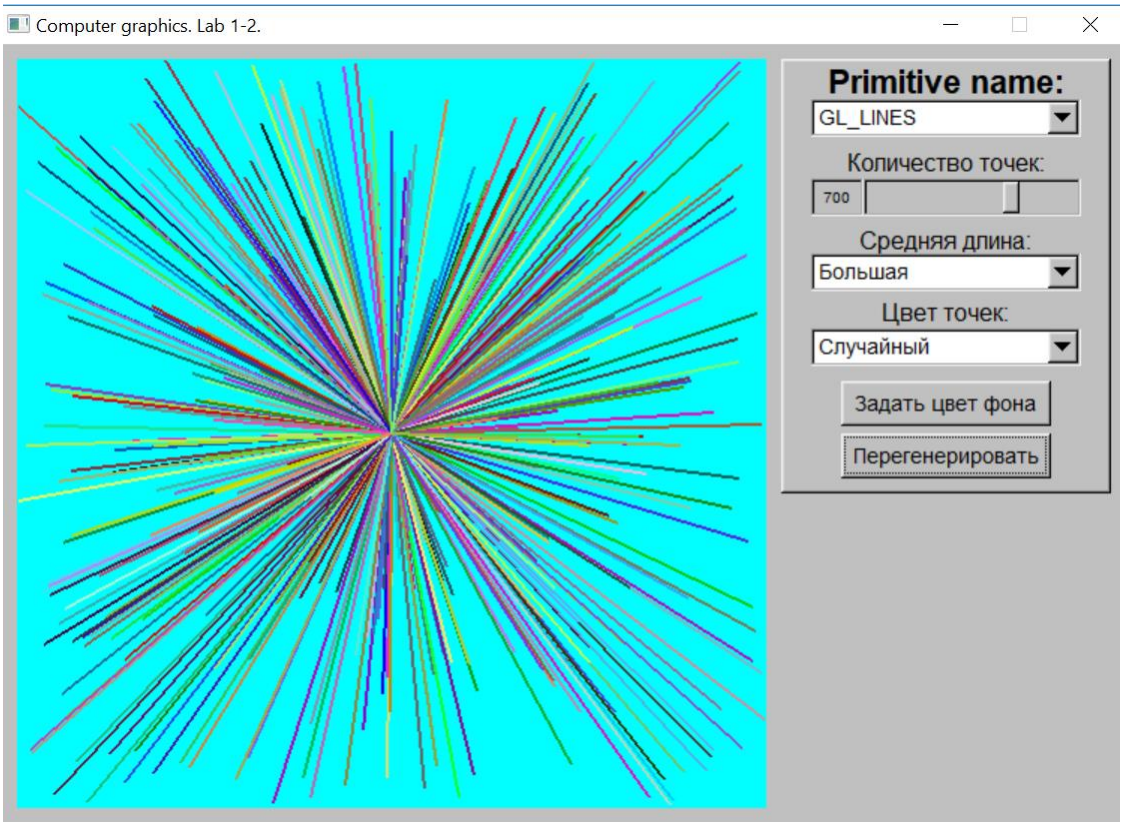
Тестирование.

Результаты тестирования представлены на изображениях:

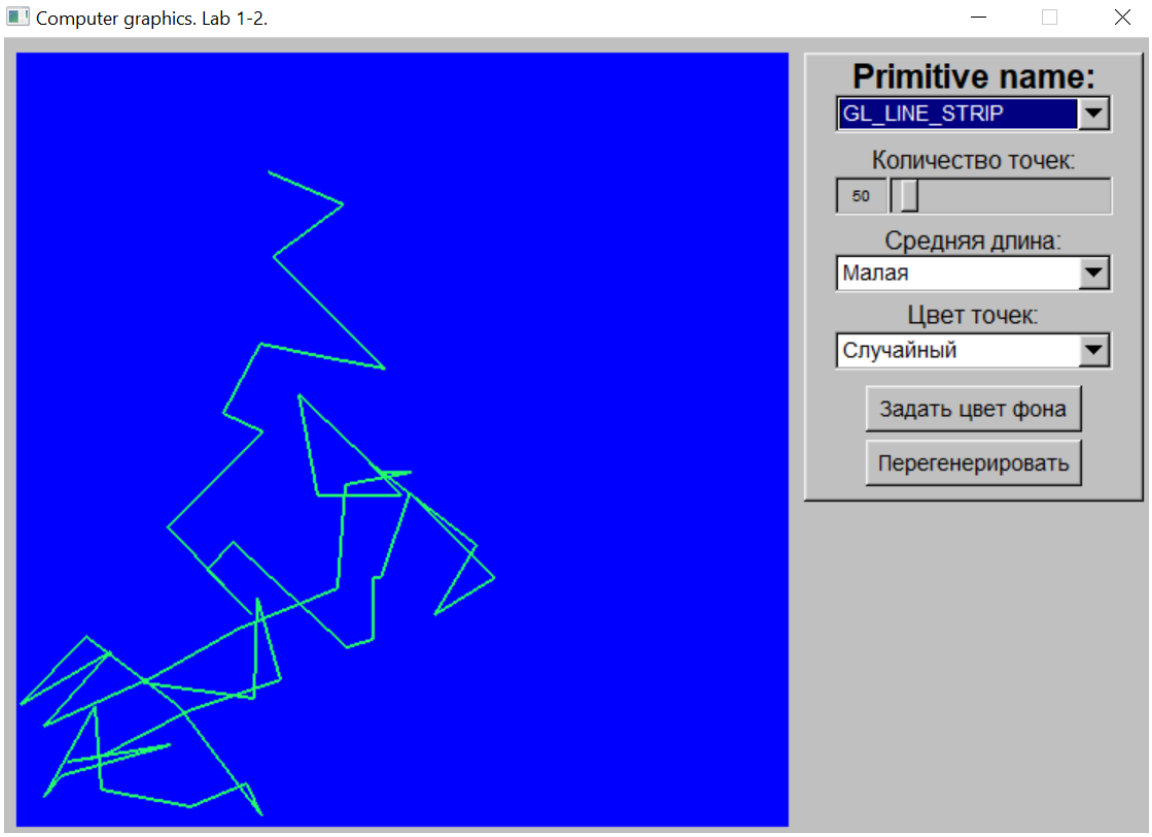
POINTS:



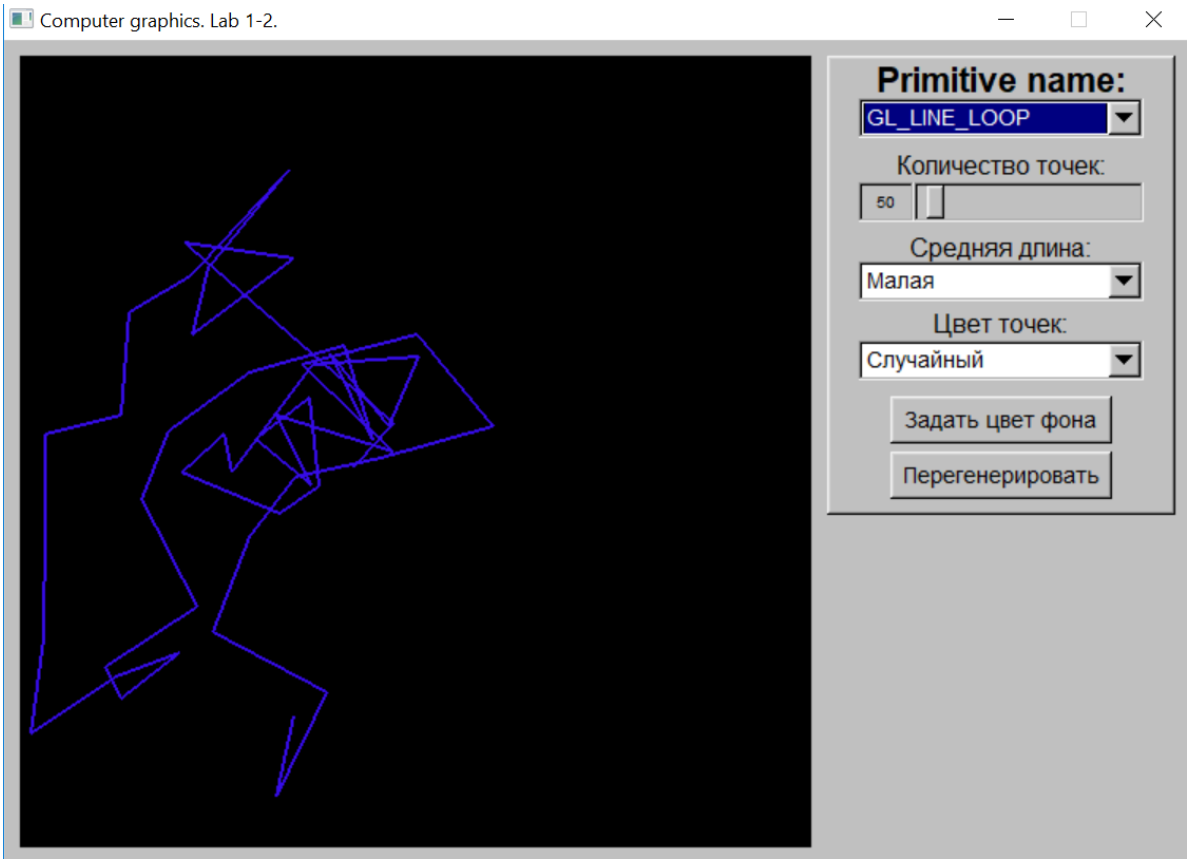
LINES:



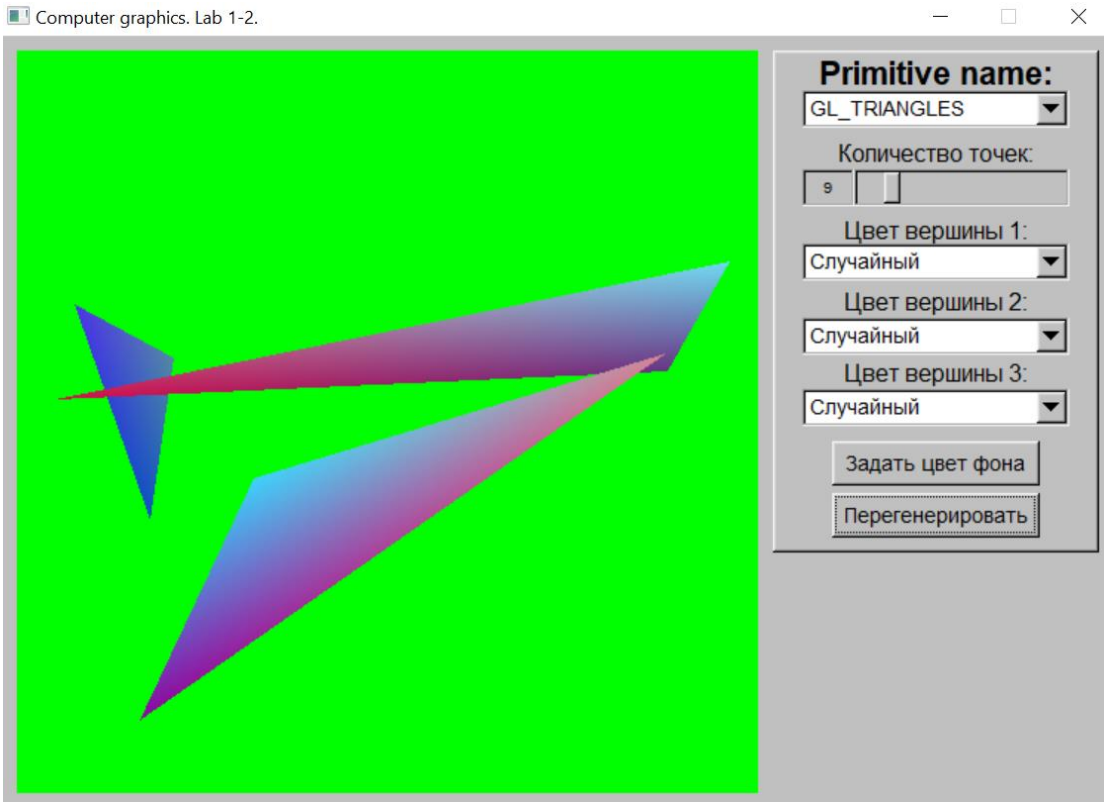
LINE STRIP:



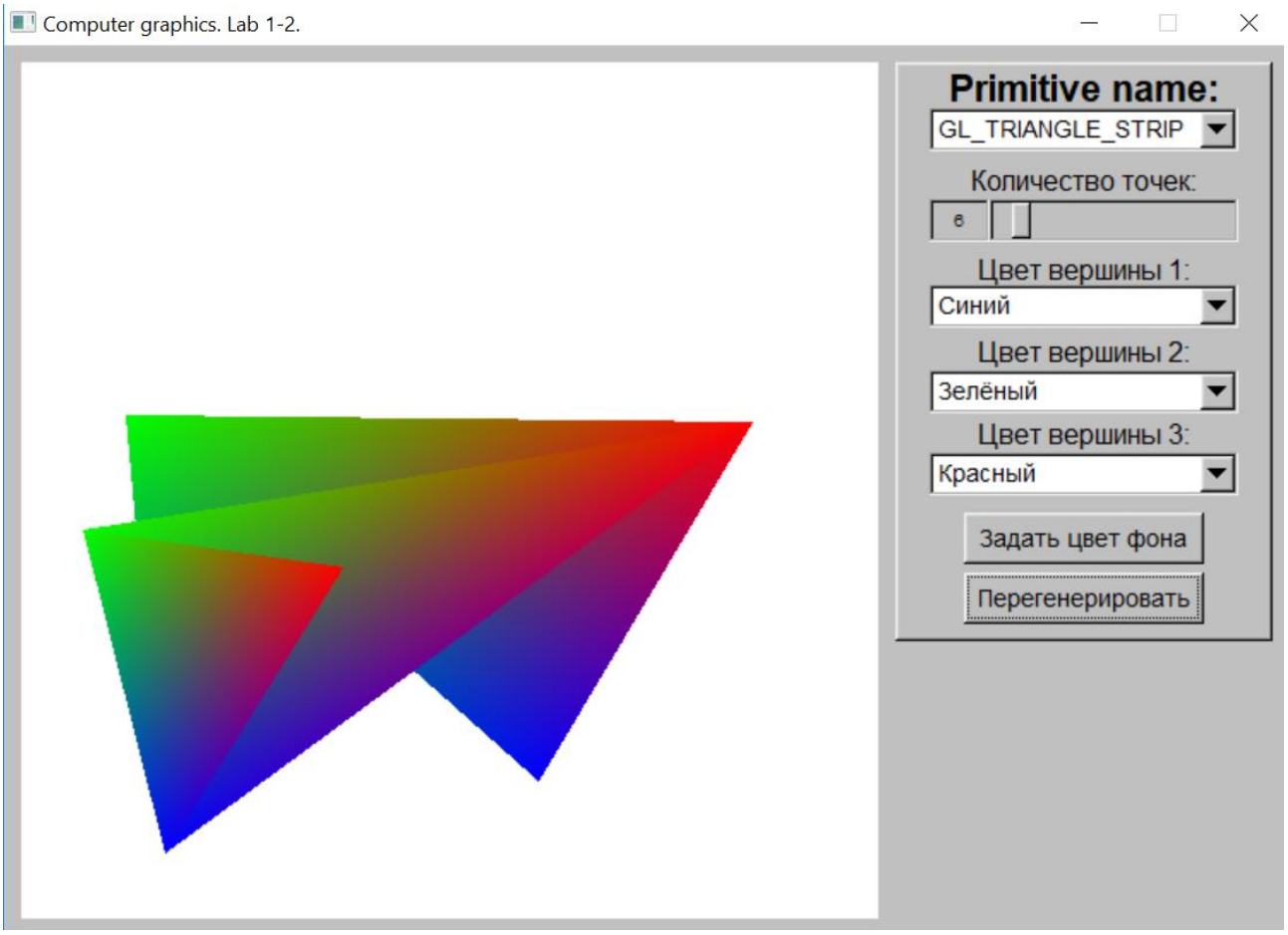
LINE LOOP:



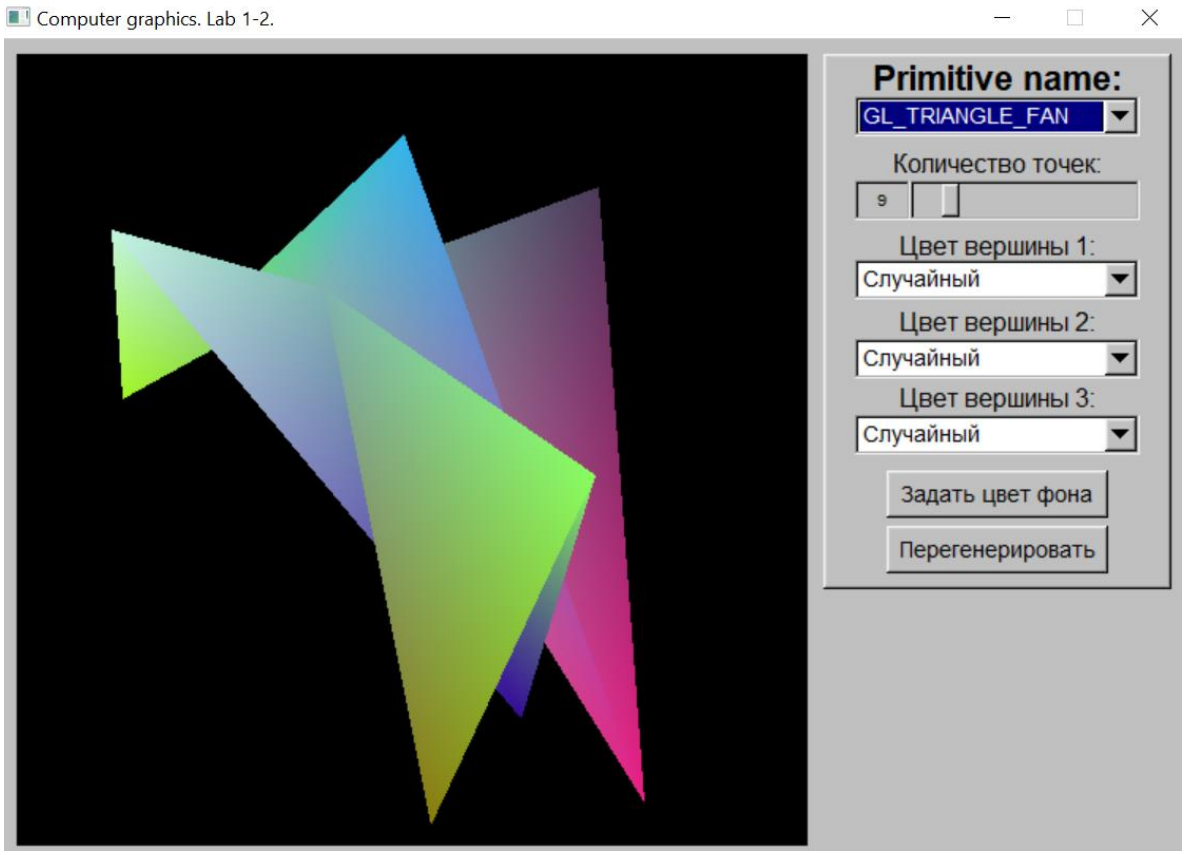
TRIANGLES:



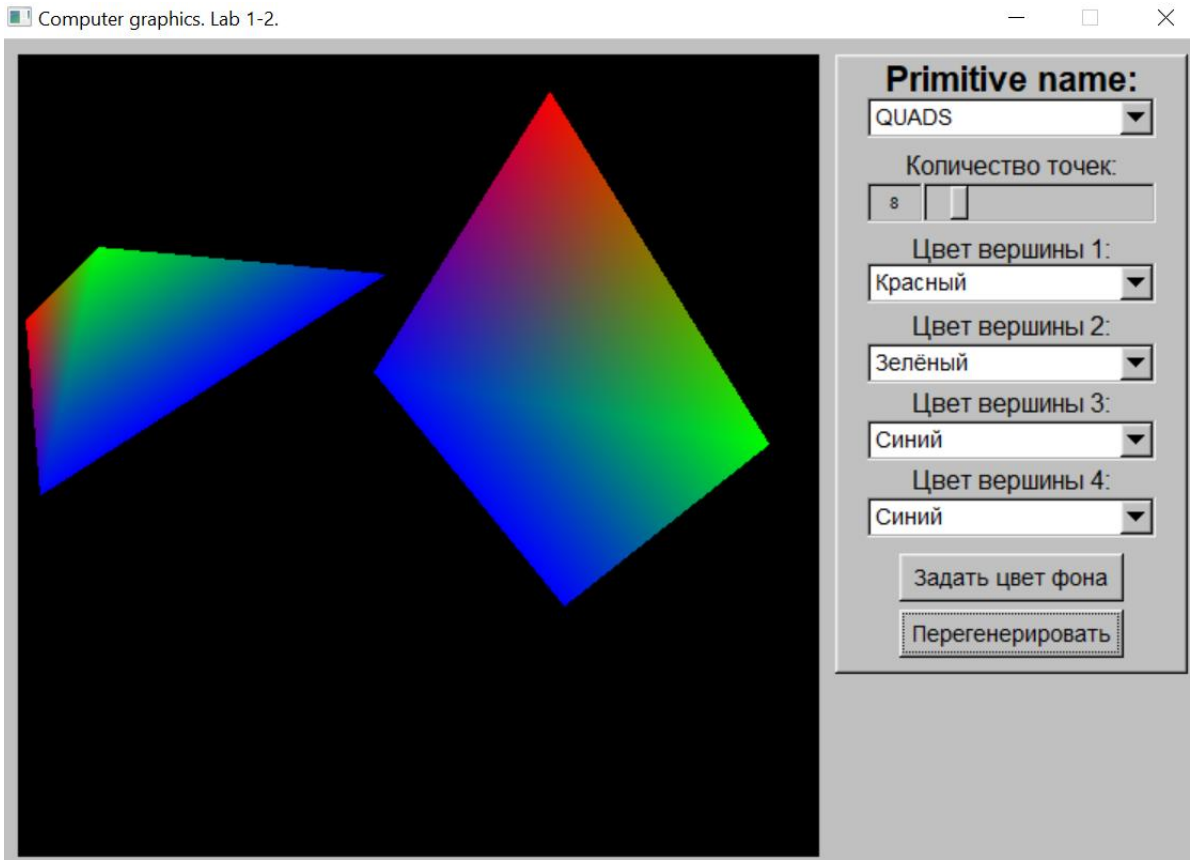
TRIANGLE STRIP:



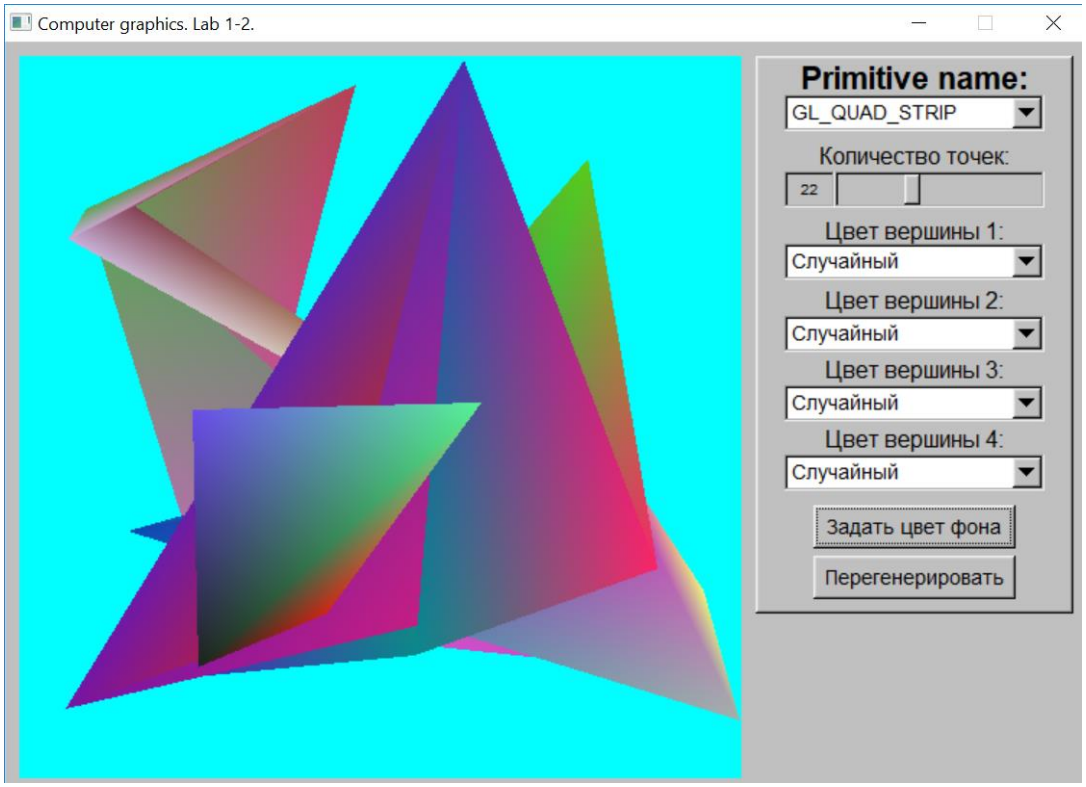
TRIANGLE FAN:



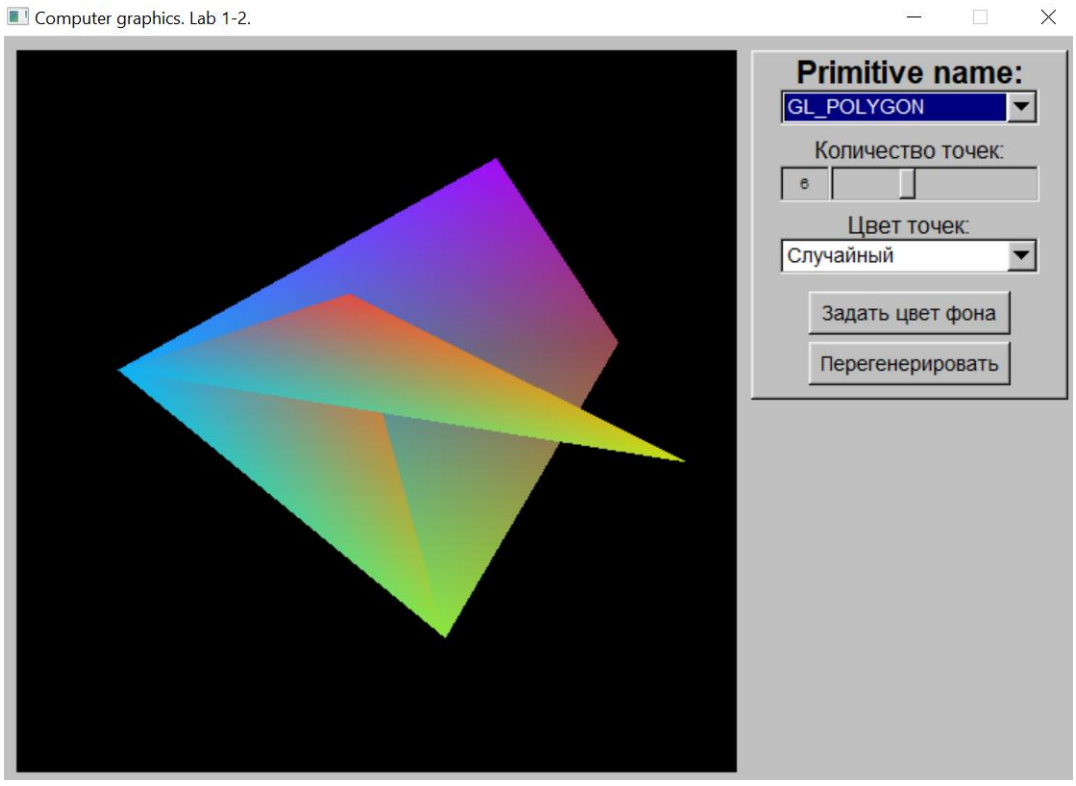
QUADS:



QUAD STRIP:



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.