Санкт-Петербургский Государственный
Электротехнический Университет

Кафедра МОЭВМ

Задание для лабораторной работы № 1
"Примитивы OpenGL"

Выполнили: Быков И. В.
Спас А. А.
Факультет: ФКТИ
Группа: 6383
Преподаватель: Герасимова Т.В.

Санкт-Петербург
2019 г.

Задание

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

Общие сведения

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.

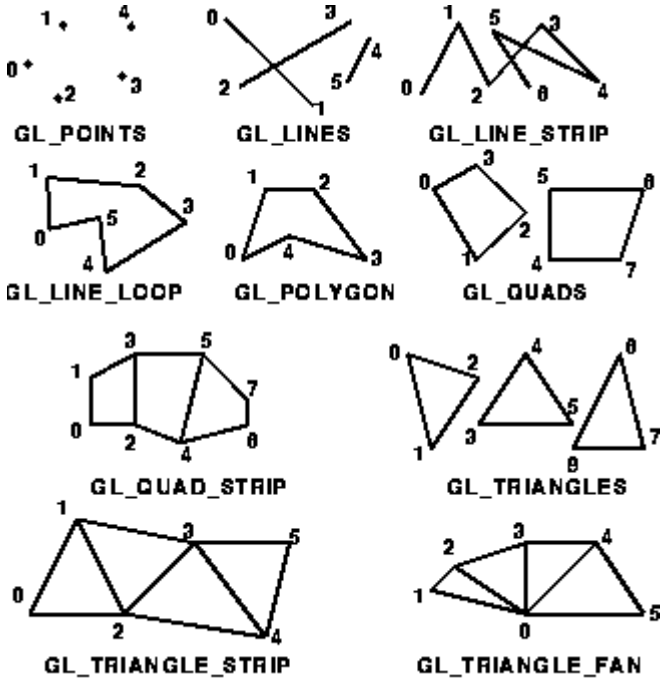
GL_TRIANGLE_FAN - в этом режиме рисуется группа связанных треугольников, имеющих общие грани и одну общую вершину. Первые три вершины определяют первый треугольник, первая, третья и четвертая – второй и т.д. Всего рисуется (N-2) треугольника.

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

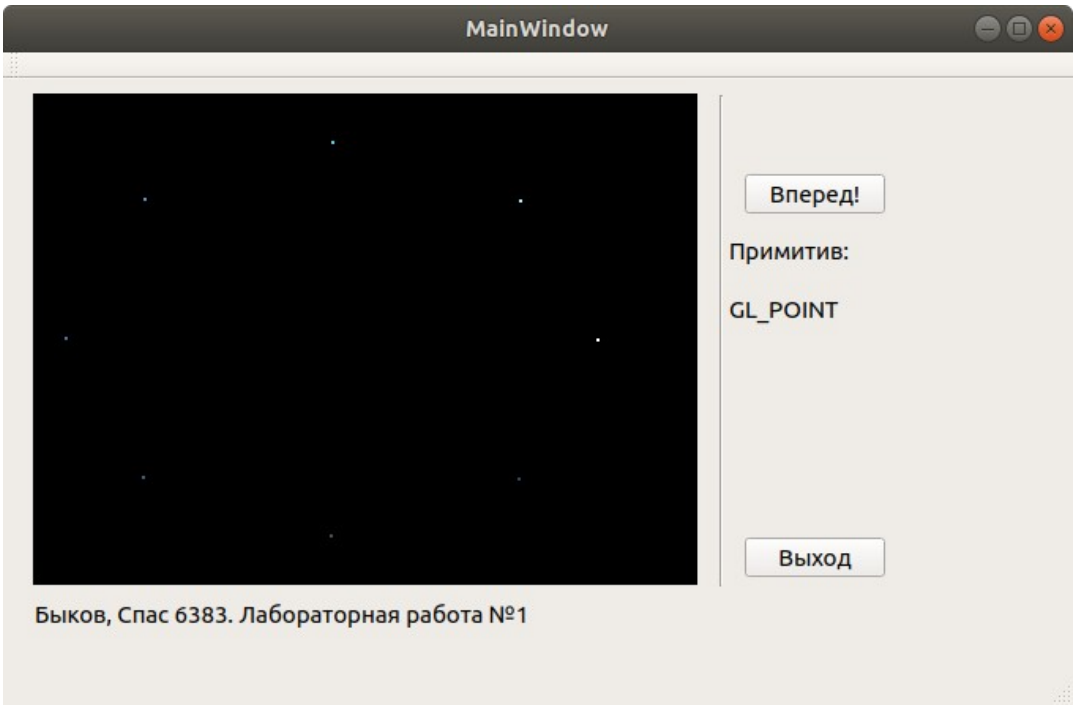
Работа выполнена в феймворке Qt 5.9.5. Создан проект Qt Widget. Примитивы рисуются в OpenGL Widget (виджет для отрисовки графики OpenGL - https://doc.qt.io/qt-5/qopenglwidget.html). Фигуры рисуются по очереди. Первая фигура — GL_POINTS, остальные рисуются в порядке, указанном в задании после нажатия кнопки «Вперед». Для прорисовки всех фигур, используем функцию void GLWidget::draw(float x, float y, GLenum type), которая получает на вход смещение точки по абциссе и ординате и тип фигуры.

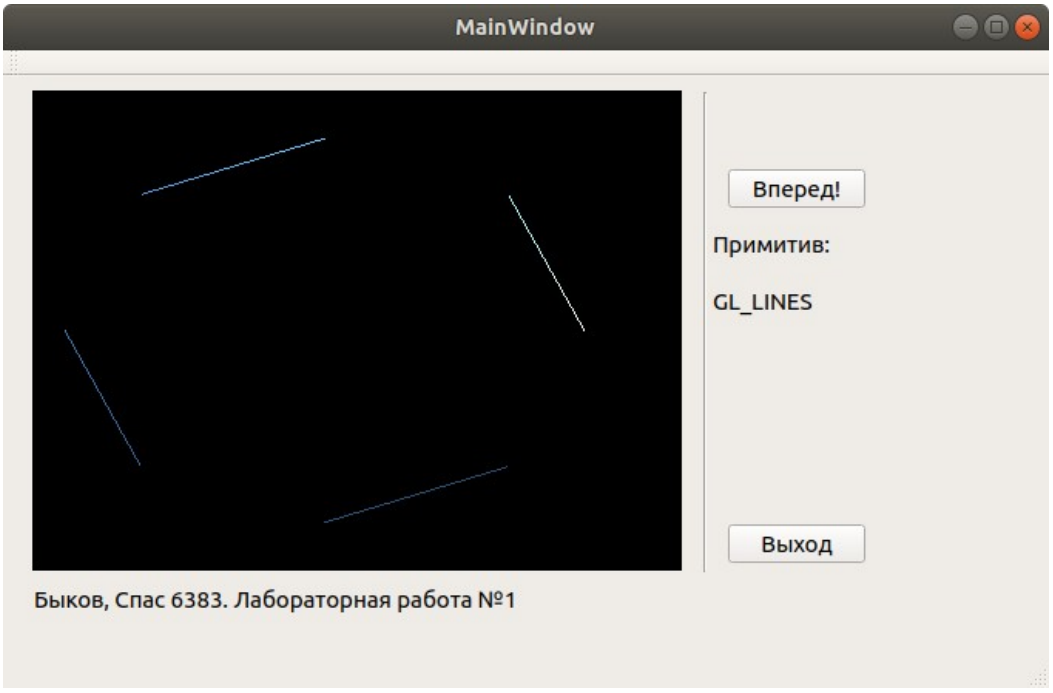
```
void GLWidget::draw(float x, float y, GLenum type) {
    int n = 8;
    glPointSize(2);
    glBegin(type);
        for (int i = 0; i < n; i++) {
            float angle = 2 * 3.14 * i / (n);
            float x_ = (( -0.2 + cos(angle) * 0.8 + x));
            float y_ = (( -0.1 + sin(angle) * 0.8 + y));
            glColor3f((float)1/(i+1),(float)2/(i+1),(float)3/(i+1));
            glVertex2f(x_, y_);
        }
    glEnd();
}
```

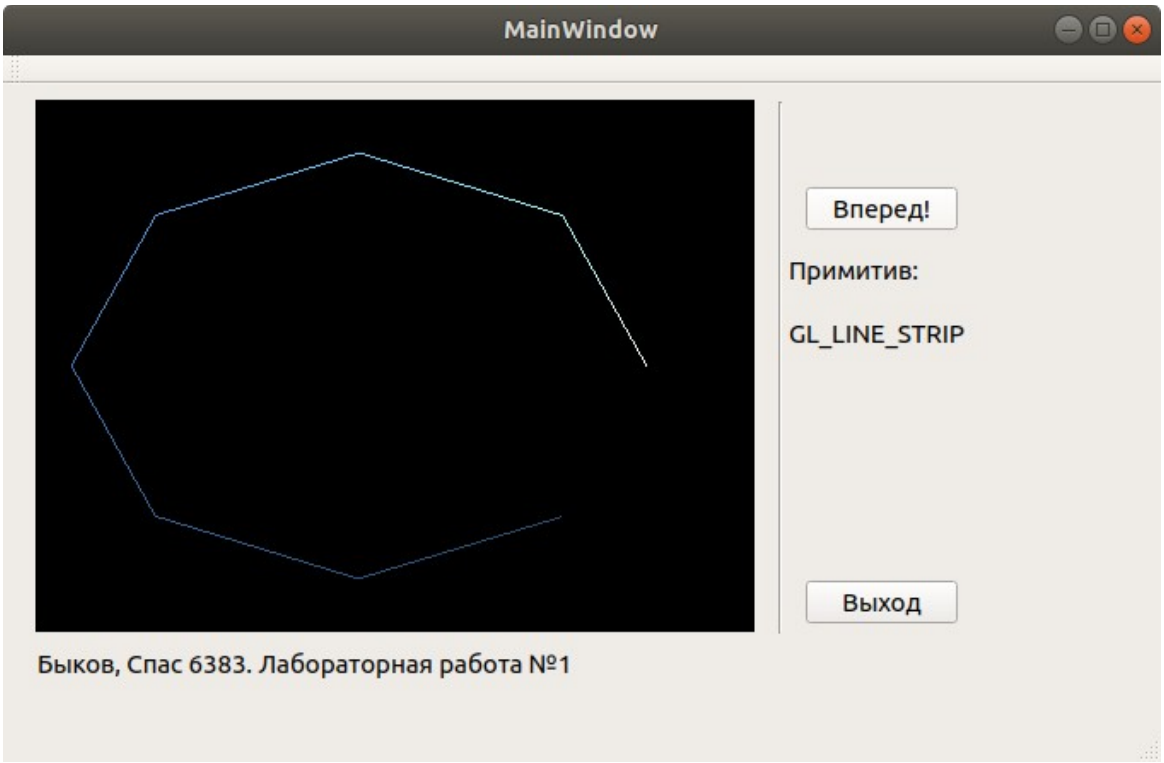
При каждом нажатии клавиши «Вперед» обновляется виджет и вызывается функция paintGL(), которая отвечает за отрисовку текущей фигуры.

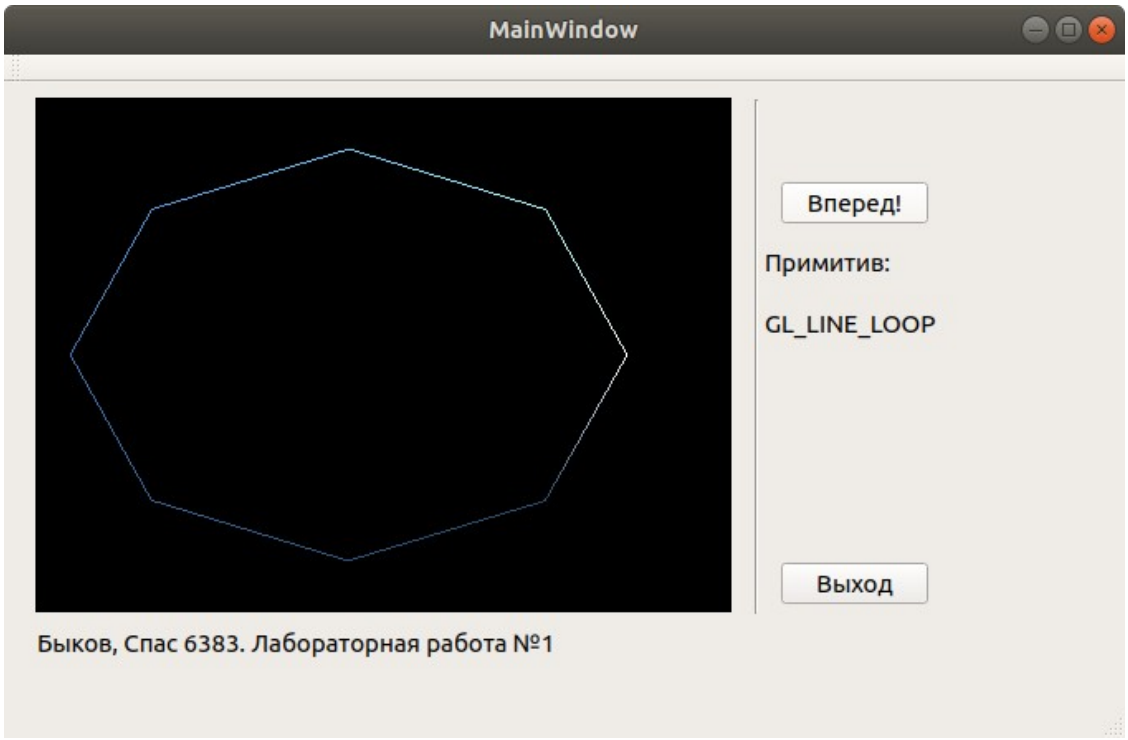
Тестирование

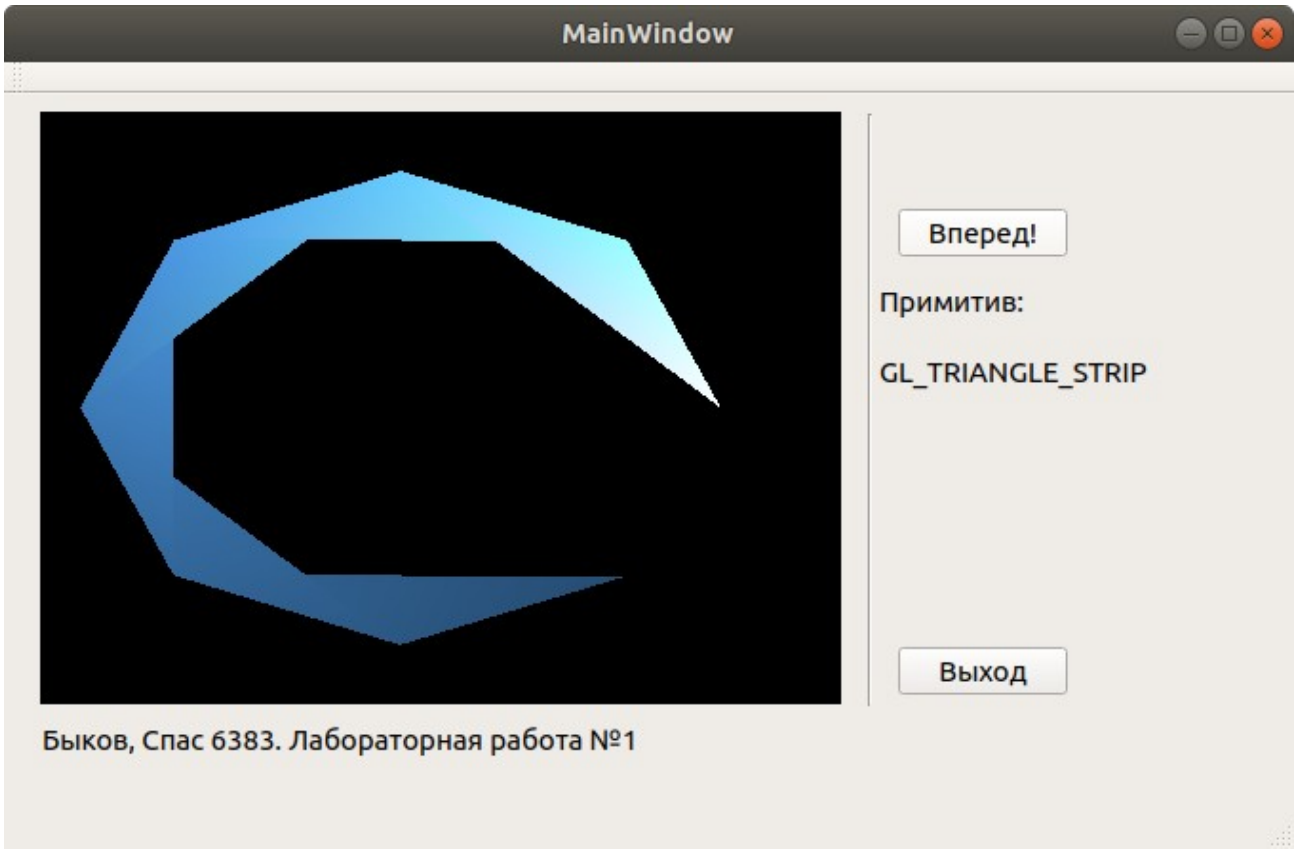
Результаты тестирования методом черного ящика приведены ниже:

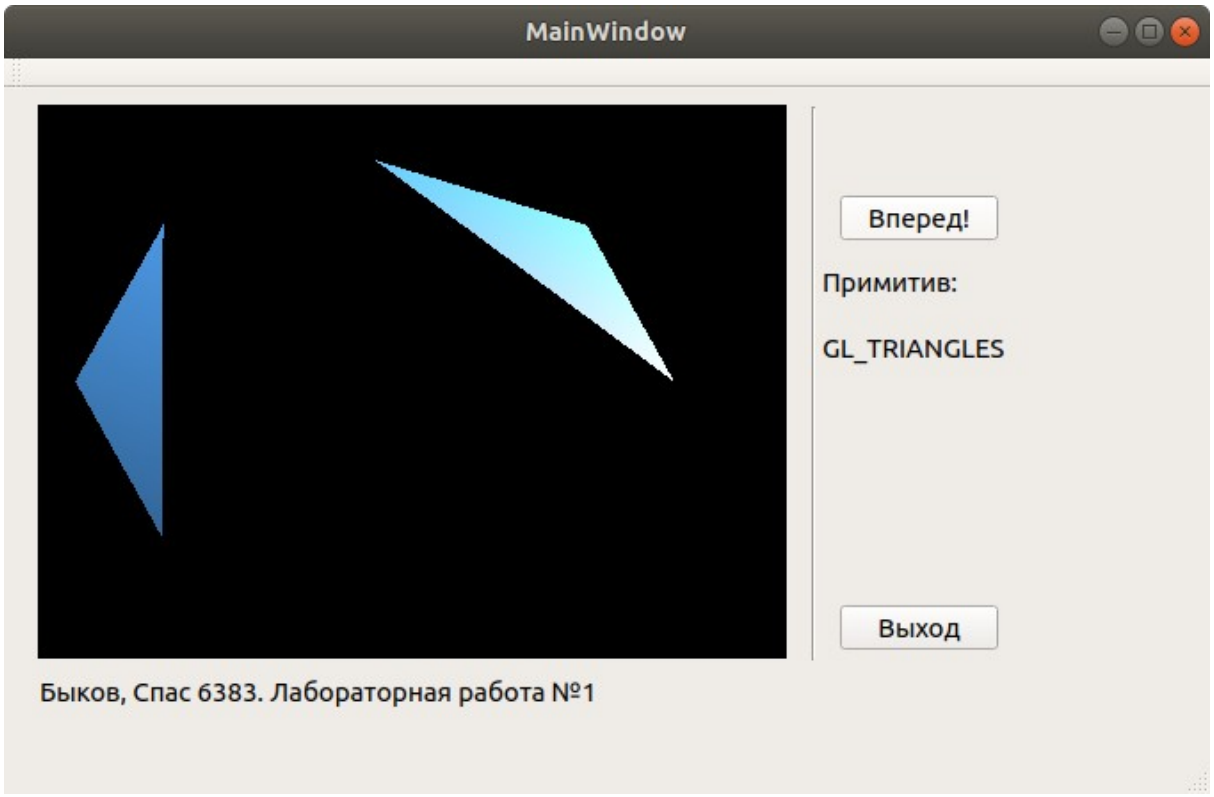


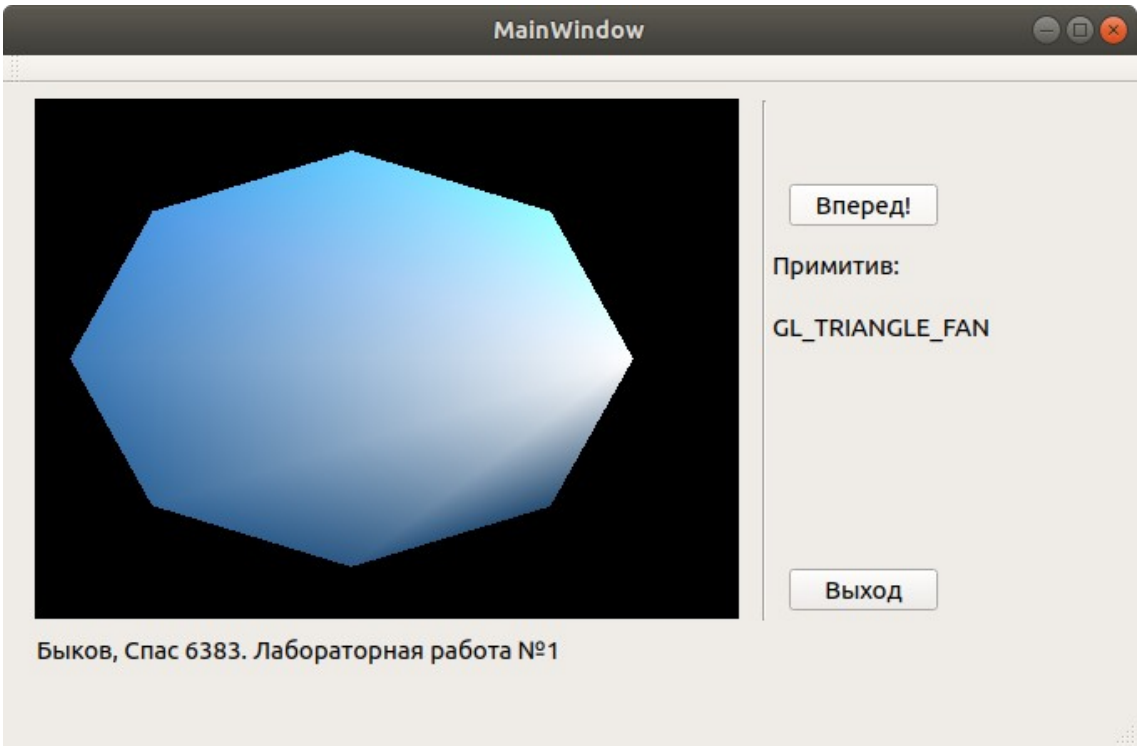


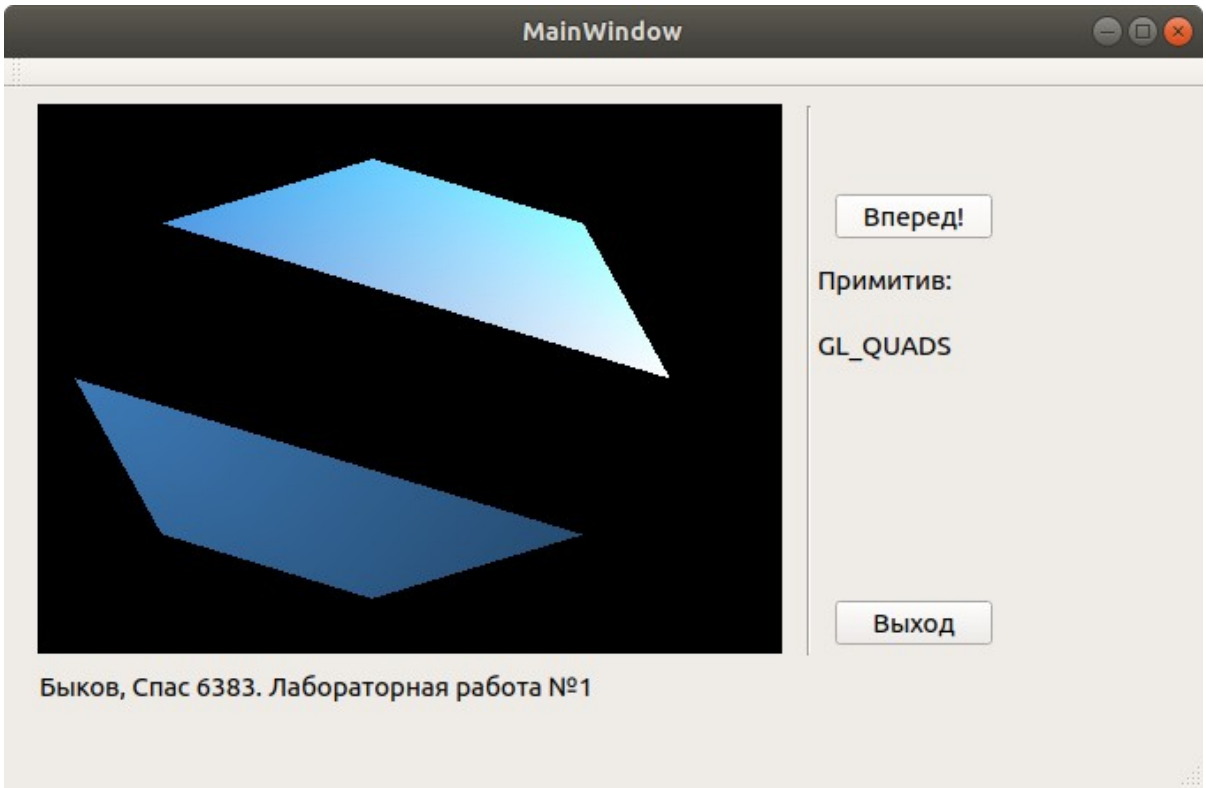

MainWindow
Вперед!
Примитив:
GL_LINE_STRIP
Выход
Быков, Спас 6383. Лабораторная работа №1

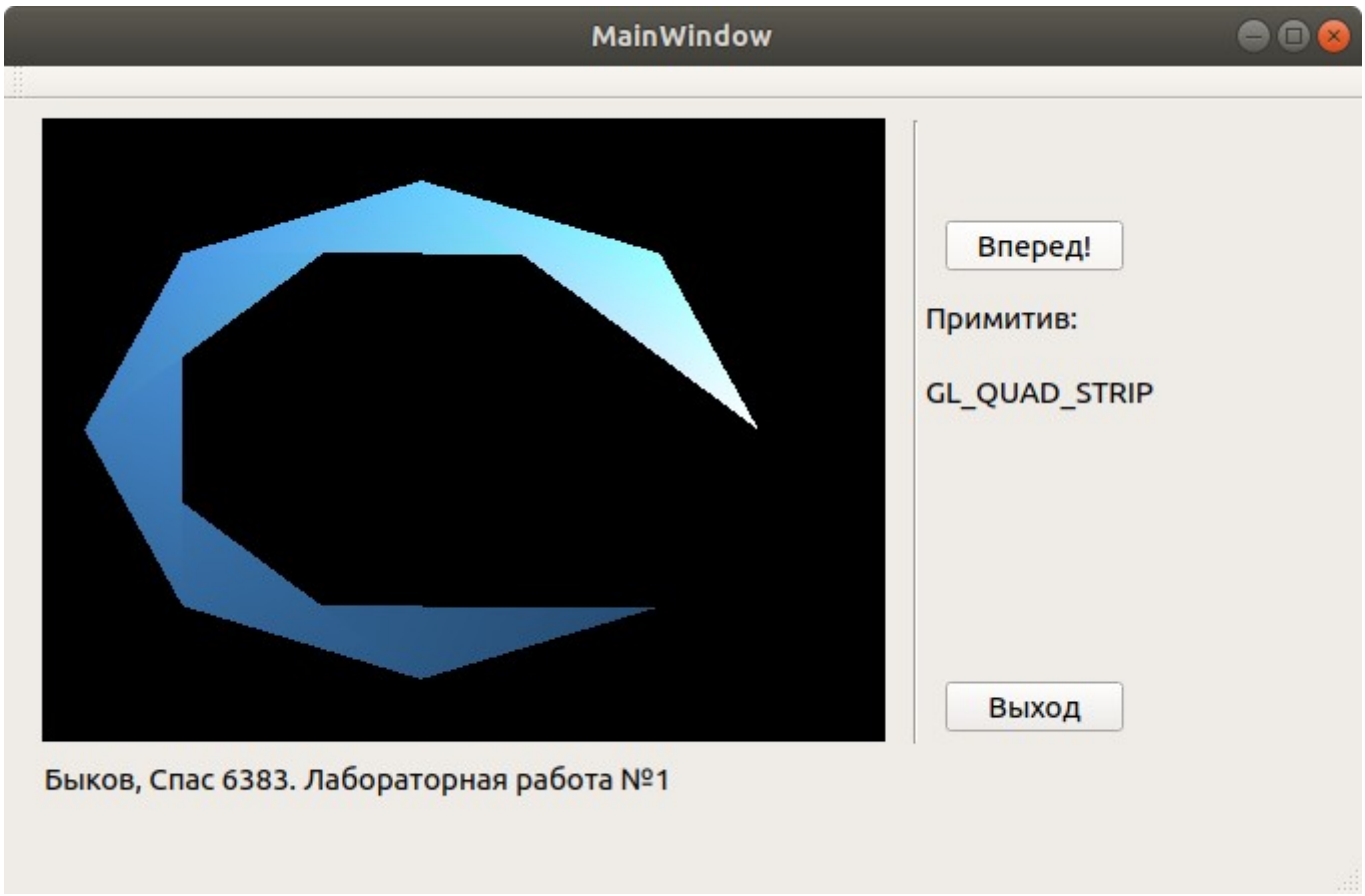

MainWindow
Вперед!
Примитив:
GL_LINE_LOOP
Выход
Быков, Спас 6383. Лабораторная работа №1

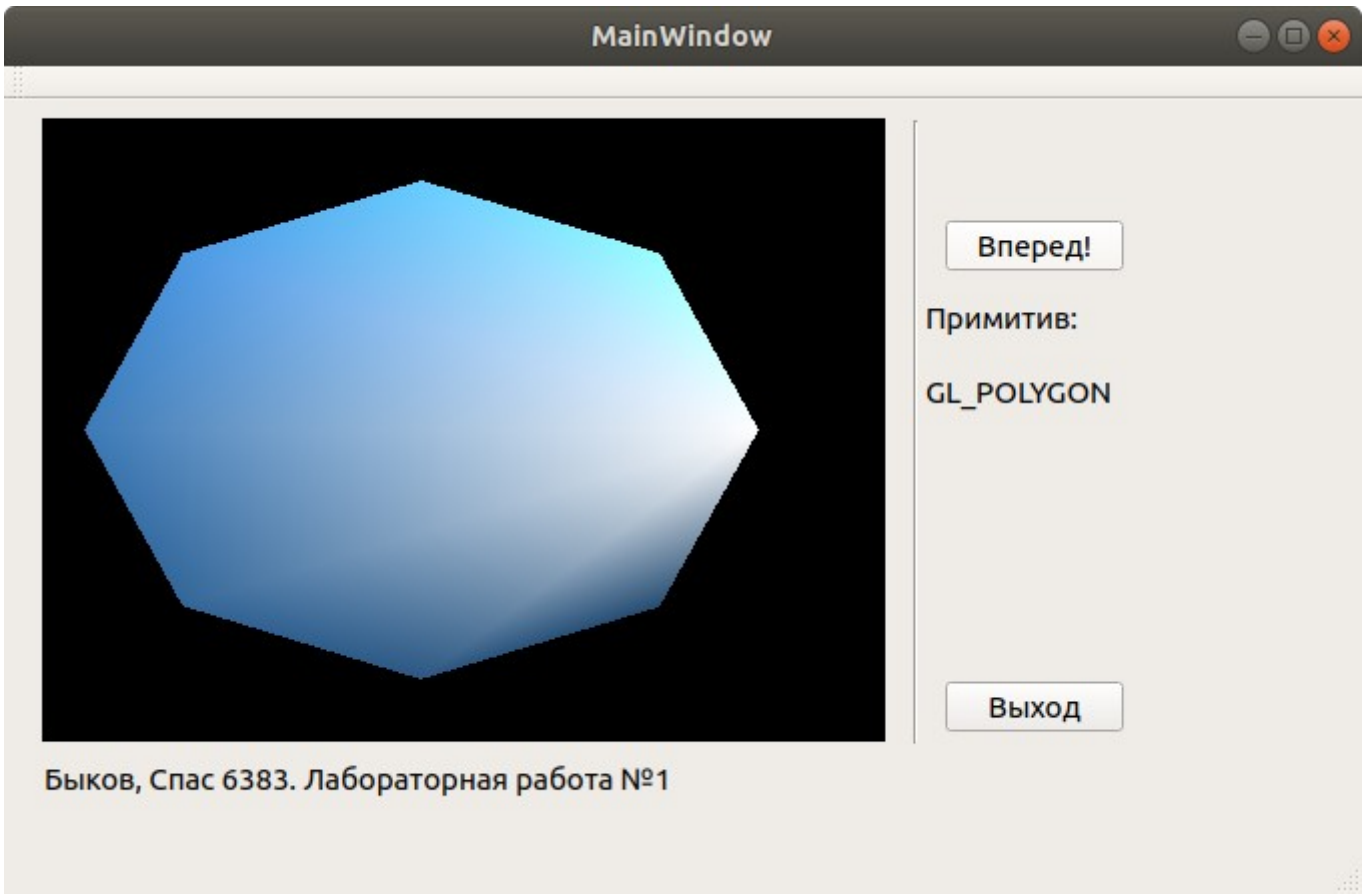

MainWindow
Вперед!
Примитив:
GL_TRIANGLE_STRIP
Выход
Быков, Спас 6383. Лабораторная работа №1


MainWindow
Вперед!
Примитив:
GL_TRIANGLES
Выход
Быков, Спас 6383. Лабораторная работа №1


MainWindow
Вперед!
Примитив:
GL_TRIANGLE_FAN
Выход
Быков, Спас 6383. Лабораторная работа №1


MainWindow
Вперед!
Примитив:
GL_QUADS
Выход
Быков, Спас 6383. Лабораторная работа №1


MainWindow
Вперед!
Примитив:
GL_QUAD_STRIP
Выход
Быков, Спас 6383. Лабораторная работа №1


MainWindow
Вперед!
Примитив:
GL_POLYGON
Выход
Быков, Спас 6383. Лабораторная работа №1

Вывод

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

Приложение. Код программы

main.cpp

#include "mainwindow.h"

```
#include <QApplication>
#include <QOpenGLWidget>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

mainwindow.cpp

#include "mainwindow.h"

```
#include "ui_mainwindow.h"
#include <QMessageBox>
#include <string>

/*
std::map< const QString, GLenum > MainWindow::mapOfFigures() {
    std::map< const QString, GLenum > map = {{"GL_POINT", GL_POINT},
                {"GL_LINES", GL_LINES},
                {"GL_LINE_STRIP", GL_LINE_STRIP},
                {"GL_LINE_LOOP", GL_LINE_LOOP},
                {"GL_TRIANGLES", GL_TRIANGLES},
                {"GL_TRIANGLE_STRIP", GL_TRIANGLE_STRIP},
                {"GL_TRIANGLE_FAN", GL_TRIANGLE_FAN},
                {"GL_QUADS", GL_QUADS},
                {"GL_QUAD_STRIP", GL_QUAD_STRIP},
                {"GL_POLYGON", GL_POLYGON}};
    return map;
}
*/
std::map< const QString, int > MainWindow::mapOfFigures() {
    std::map< const QString, int > map = {{"GL_POINT", 0},
                {"GL_LINES", 1},
                {"GL_LINE_STRIP", 2},
                {"GL_LINE_LOOP", 3},
```

```
                {"GL_TRIANGLES", 4},
                {"GL_TRIANGLE_STRIP", 5},
                {"GL_TRIANGLE_FAN", 6},
                {"GL_QUADS", 7},
                {"GL_QUAD_STRIP", 8},
                {"GL_POLYGON", 9}};
    return map;
}

std::vector<std::string> arrOfFigures =
{"GL_POINT","GL_LINES","GL_LINE_STRIP","GL_LINE_LOOP","GL_TRIANGLES","GL_TRIANGLE_STR
IP","GL_TRIANGLE_FAN","GL_QUADS","GL_QUAD_STRIP","GL_POLYGON"};

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    std::map< const QString, int > map = mapOfFigures();
    /*for (auto it = map.begin(); it != map.end(); ++it) {
        ui->comboBox->addItem(it->first);
    }*/
    /*for (std::string i : arrOfFigures) {
        ui->comboBox->addItem(QString::fromStdString(i));
    }*/
    ui->label_3->setText(QString::fromStdString(arrOfFigures[0]));
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
     QApplication::exit();
}

int num = 0;

void MainWindow::on_pushButton_2_clicked()
{
    ui->openGLWidget->update();
    ++num;
    if (num == 10) num = 0;
    ui->label_3->setText(QString::fromStdString(arrOfFigures[num]));
}
```

glwidget.cpp

#include "glwidget.h"

```
#include "mainwindow.h"
#include <QMessageBox>
#include <vector>
#include <cstdlib>

int flag = -1;

std::vector<GLenum> arrOfFigures =
{GL_POINTS,GL_LINES,GL_LINE_STRIP,GL_LINE_LOOP,GL_TRIANGLES,GL_TRIANGLE_STRIP,GL_TRIA
NGLE_FAN,GL_QUADS,GL_QUAD_STRIP,GL_POLYGON};
std::vector<QString> arr =
{"GL_POINT","GL_LINES","GL_LINE_STRIP","GL_LINE_LOOP","GL_TRIANGLES","GL_TRIANGLE_STR
IP","GL_TRIANGLE_FAN","GL_QUADS","GL_QUAD_STRIP","GL_POLYGON"};

GLWidget::GLWidget(QWidget *parent)
    : QOpenGLWidget(parent) {}

GLWidget::~GLWidget() {}

void GLWidget::initializeGL() {
    initializeOpenGLFunctions();
    glClearColor(0,0,0,0);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
    glEnable(GL_COLOR_MATERIAL);
}

void GLWidget::paintGL() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    draw(0.1, 0.1, arrOfFigures[flag]);
    flag++;
    if (flag == 10) flag = 0;

}

void GLWidget::draw(float x, float y, GLenum type) {
    int n = 8;
    glPointSize(2);
    glBegin(type);
        for (int i = 0; i < n; i++) {
            float angle = 2 * 3.14 * i / (n);
            float x_ = (( -0.2 + cos(angle) * 0.8 + x));
            float y_ = (( -0.1 + sin(angle) * 0.8 + y));
            glColor3f((float)1/(i+1),(float)2/(i+1),(float)3/(i+1));
            glVertex2f(x_, y_);
        }
    glEnd();
}
```

glwidget.h

#ifndef GLWIDGET_H

```
#define GLWIDGET_H
#include <QOpenGLWidget>
#include <QOpenGLFunctions>

class MainWindow;

class GLWidget : public QOpenGLWidget, protected QOpenGLFunctions
{
public:
    GLWidget(QWidget *parent = 0);
    ~GLWidget();
    void initializeGL();
    void paintGL() override;
    void draw(float, float, GLenum);
};

#endif // GLWIDGET_H
```

mainwindow.h

#ifndef MAINWINDOW_H

```
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtGui>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  explicit MainWindow(QWidget *parent = 0);
  std::map< const QString, int> mapOfFigures();
  ~MainWindow();
private slots:
```

```
    void on_pushButton_clicked();
  //  void on_comboBox_activated();

    void on_pushButton_2_clicked();

public:
  Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```