Санкт-Петербургский
ГосударственныйЭлектротехническийУниве
рситет

Кафедра МОЭВМ

Задание для лабораторной работы № 1
"Примитивы OpenGL"

Выполнили:
Доброхвалов М. О. 6303
Черкасова Е. И. 6382
Факультет: КТИ
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) .

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 отрезокn. Всего рисуетсяN/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 ит.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 рисуетсяN/4четырехугольника.

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

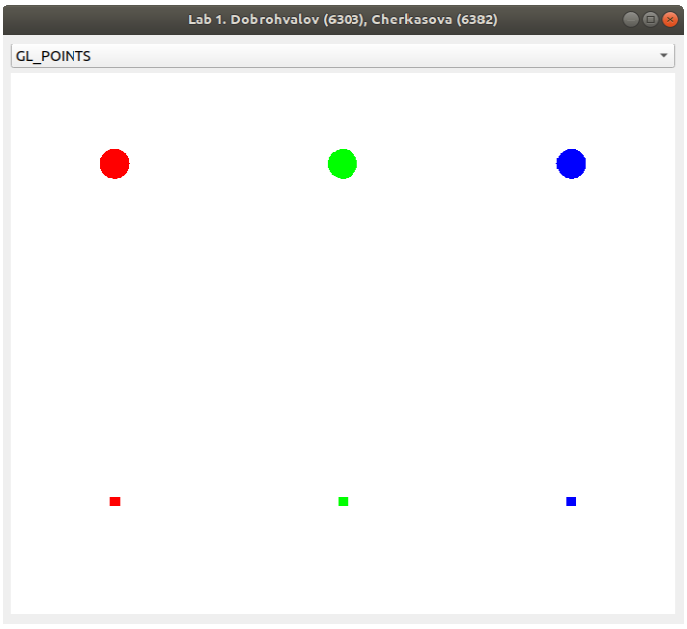
Работа выполнена на ОС Linux с помощью графической библиотеки OpenGL, а также библиотеки PyQt для интерфейса.

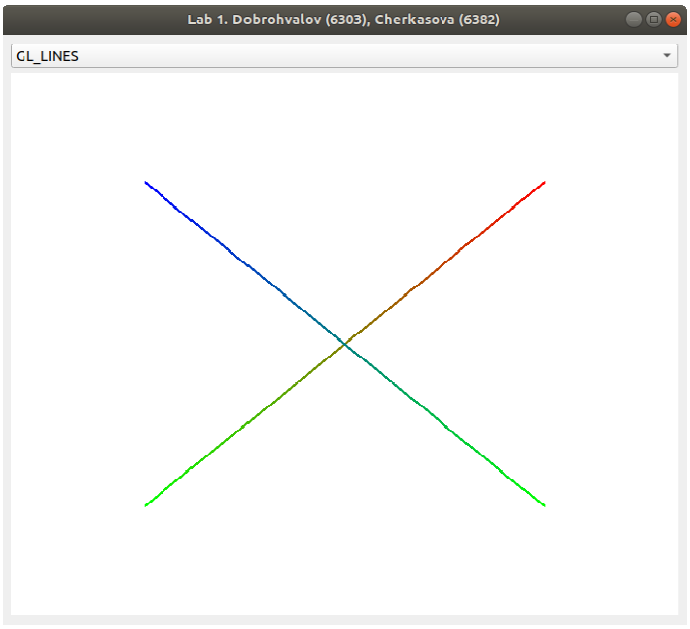
Ход работы

1. Для реализации программы, представляющей определенный набор 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), была использована командная строка на операционной системе Ubuntu 18.04 с подключением и настройкой следующих библиотек: OpenGL — это графический стандарт в област компьютерной графики; PyQt — набор «привязок» графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.
2. Был разработан интерфейс для взаимодействия с пользователем (Приложение 2)
3. Для инициализации приложения и виджета были написанны соответствующие коды(Приложение 2 и 3)
4. Тестирование. Результаты выполнения программы представлены на рисунках ниже.

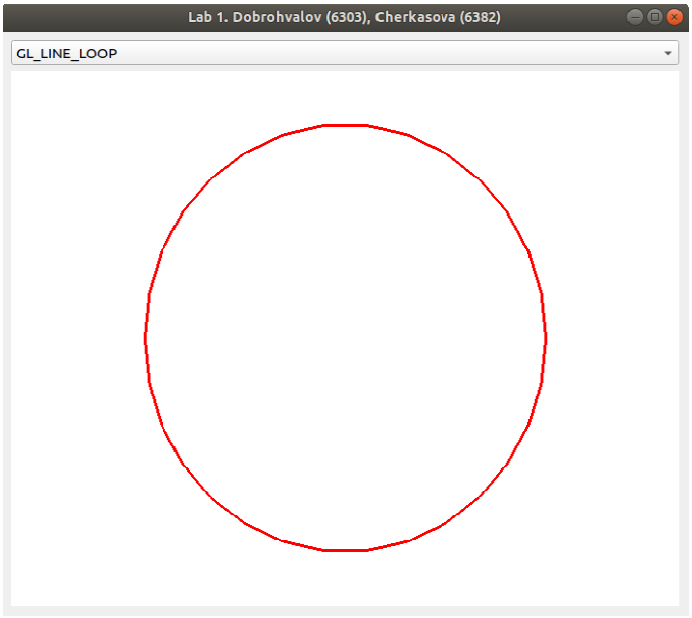
Вывод

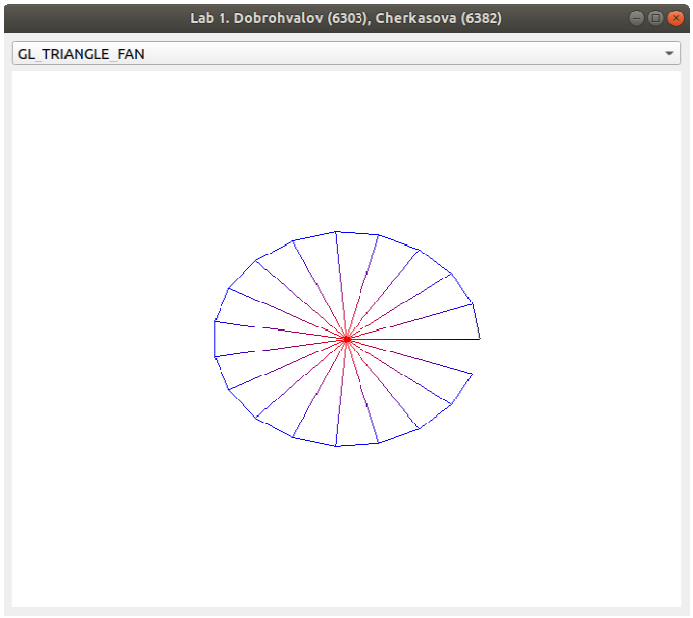
В результате выполнения лабораторной работы была разработана программа, создающая графические примитивы OpenGL. Программа работает корректно. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.

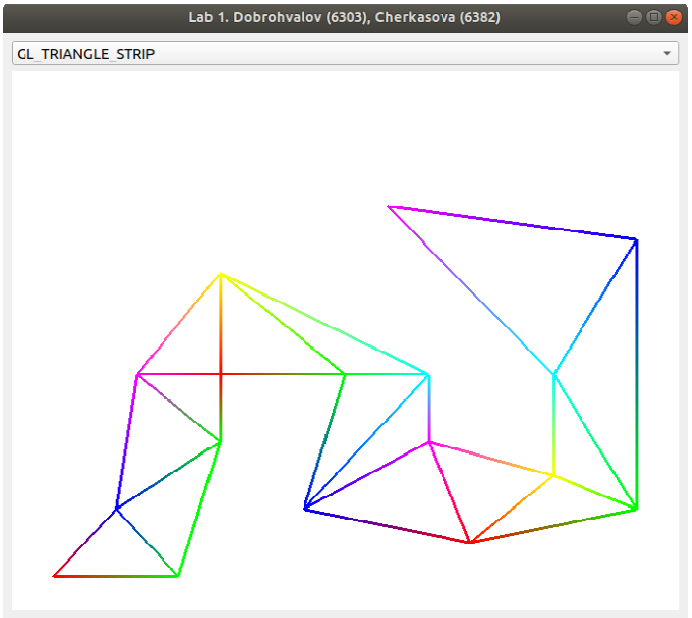


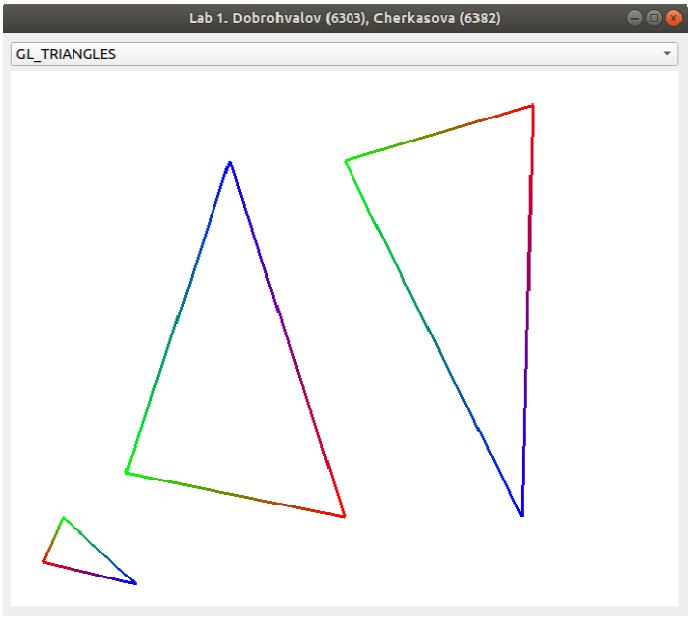


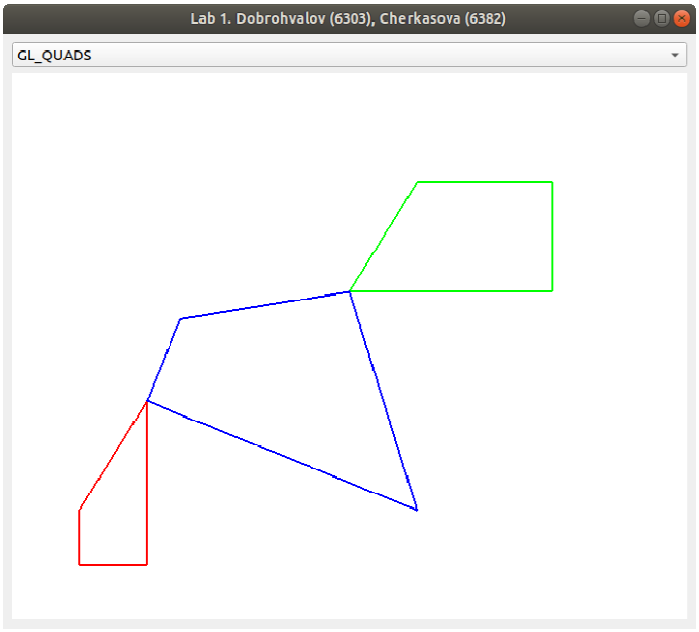

Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_LINE_STRIP

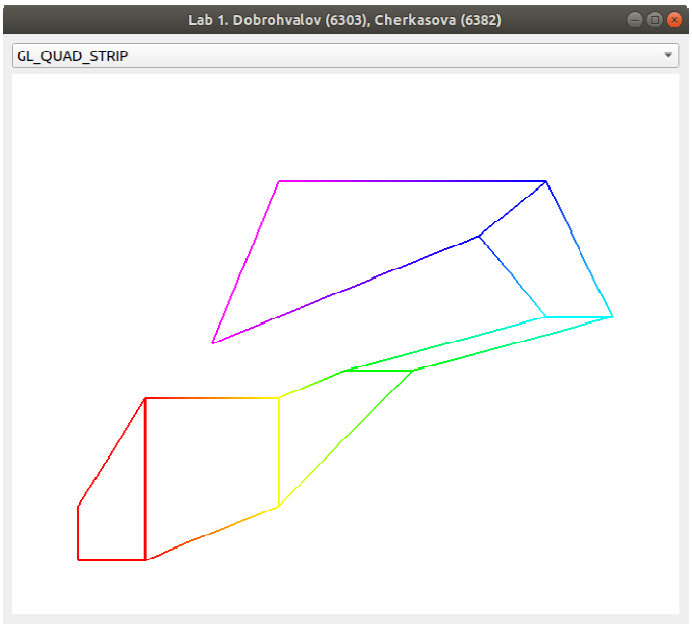

Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_LINE_LOOP


Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_TRIANGLE_FAN


Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_TRIANGLE_STRIP


Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_TRIANGLES


Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_QUADS


Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_QUAD_STRIP


Lab 1. Dobrohvalov (6303), Cherkasova (6382)
GL_POLYGON

Приложение 1.

```
# -*- coding: utf-8 -*-

from PyQt5 import QtCore, QtGui, QtWidgets
from controller import GLWidget

class Ui_GLInterface(object):
    def setupUi(self, GLInterface):
        GLInterface.setObjectName("GLInterface")
        GLInterface.resize(700, 600)

        self.centralwidget = QtWidgets.QWidget(GLInterface)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName("verticalLayout")
        self.comboBox = QtWidgets.QComboBox(self.centralwidget)
        self.comboBox.setObjectName("comboBox")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.verticalLayout.addWidget(self.comboBox)
        self.openGLWidget = GLWidget(self)
        self.openGLWidget.setObjectName("openGLWidget")
        self.verticalLayout.addWidget(self.openGLWidget)
        GLInterface.setCentralWidget(self.centralwidget)

        self.retranslateUi(GLInterface)
        QtCore.QMetaObject.connectSlotsByName(GLInterface)

    def retranslateUi(self, GLInterface):
        _translate = QtCore.QCoreApplication.translate
        GLInterface.setWindowTitle(_translate("GLInterface", "Lab 1. Dobrohvalov (6303), Cherkasova (6382)"))

        self.comboBox.setItemText(0, _translate("GLInterface", "GL_POINTS"))
        self.comboBox.setItemText(1, _translate("GLInterface", "GL_LINES"))
        self.comboBox.setItemText(2, _translate("GLInterface", "GL_LINE_STRIP"))
        self.comboBox.setItemText(3, _translate("GLInterface", "GL_LINE_LOOP"))
        self.comboBox.setItemText(4, _translate("GLInterface", "GL_TRIANGLES"))
        self.comboBox.setItemText(5, _translate("GLInterface", "GL_TRIANGLE_STRIP"))
        self.comboBox.setItemText(6, _translate("GLInterface", "GL_TRIANGLE_FAN"))
        self.comboBox.setItemText(7, _translate("GLInterface", "GL_QUADS"))
        self.comboBox.setItemText(8, _translate("GLInterface", "GL_QUAD_STRIP"))
        self.comboBox.setItemText(9, _translate("GLInterface", "GL_POLYGON"))
```

Приложение 2.

```
# -*- coding: utf-8 -*-

import sys  # sys нужен для передачи argv в QApplication
from PyQt5 import QtWidgets, QtGui

import design

class GLInterface(QtWidgets.QMainWindow, design.Ui_GLInterface):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.comboBox.activated[str].connect(self.onActivated)

    def onActivated(self, text):
        self.openGLWidget.changeFigure(text)
        self.openGLWidget.updateGL()

def main():
    app = QtWidgets.QApplication(sys.argv)  # Новый экземпляр QApplication
    window = GLInterface()  # Создаём объект класса GLInterface
    window.show()  # Показываем окно
    app.exec_()  # и запускаем приложение


if __name__ == '__main__':  # Если мы запускаем файл напрямую, а не импортируем
    main()  # то запускаем функцию main()
```

Приложение 3.

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
from PyQt5.QtOpenGL import *
from random import randint, random
from numpy import sin, cos, pi, power
from lab1 import PRIMITIVES


class GLWidget(QGLWidget):
    def __init__(self, parent):
        super(GLWidget, self).__init__(parent)
        self.width = 640
        self.height = 480
        self.current_mode = 'GL_POINTS'

    def initializeGL(self):
        """It is called once before the first call to paintGL() or resizeGL(),
        and then once whenever the widget has been assigned a new QGLContext """
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) # очистка буферов
        glViewport(0, 0, self.width, self.height)
        glMatrixMode(GL_PROJECTION) # загрузка матрицы проекции
        gluOrtho2D(0, self.width, 0, self.height)
        glMatrixMode(GL_MODELVIEW)
        glClearDepth(1.0)
        glShadeModel(GL_SMOOTH)
        glLoadIdentity()

    def changeFigure(self, text):
        self.current_mode = text

    def paintGL(self):
        """It is called whenever the widget needs to be painted"""
        PRIMITIVES[self.current_mode]()

    def resizeGL(self, w, h):
        self.width = w
        self.height = h

        glViewport(0, 0, w, h)
        glOrtho(0, w, 0, h, -1.0, 1.0)
        glLoadIdentity()
```