**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**По лабораторной работе № 1**

**по дисциплине «Компьютерная графика»**

Тема: **Примитивы OpenGL**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Архипов В.А. |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

2023

**Цель работы.**

-ознакомление с основными примитивами OpenGL.

-освоение возможности подключения графической библиотеки в среду разработки.

Требования и рекомендации к выполнению задания:

-проанализировать полученное задание, выделить информационные объекты и действия;

- разработать программу с использованием требуемых примитивов и атрибутов.

**Задания.**

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 вами шаблона программа должна быть пополнена возможностями остановки интерактивно различных атрибутов примитивов рисования через вызов соответствующих элементов интерфейса пользователя.

**Выполнение работы.**

Работа была выполнена в среде разработки PyCharm на языке программирования Python 3.10. Для реализации пользовательского интерфейса были использованы библиотека PyQt6 и программа Qt Designer, создающая пользовательский интерфейс по построенному в ней макету. Для работы с графикой была использована библиотека PyOpenGL. Подключение графической библиотеки к пользовательскому интерфейсу было осуществлено с помощью виджета QOpenGLWidget из библиотеки PyQt6.

Для того, чтобы настроить среду разработки, необходимо сначала скачать обозначенные выше библиотеки. Сделать это можно с помощью команд pip install <имя библиотеки> (см. рис. 1–2).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – установка библиотеки PyQt6 в среде разработки PyCharm

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – установка библиотеки PyOpenGL в среде разработки PyCharm

Далее в программе Qt Designer был создан макет приложения, который был сохранен в разрешении .ui (см. рис. 3).

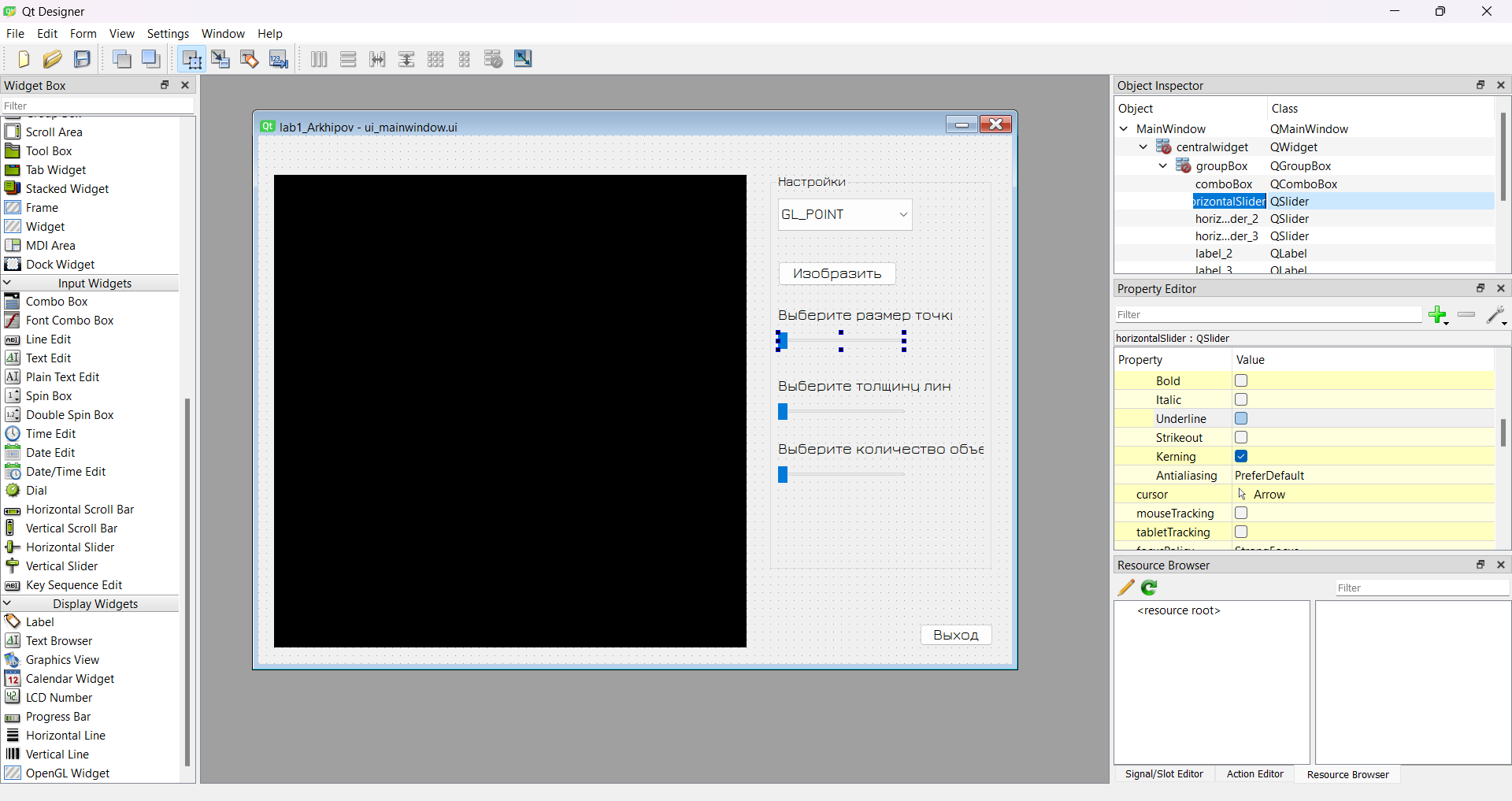


Рисунок 3 – разработанный в Qt Designer макет приложения

Полученный файл нужно сохранить в рабочей директории и на его основе создать файл с расширением .py. Сделать это можно набрав в терминале команду pyuic6 ui\_mainwindow.ui —output ui\_mainwindow.py.

В результате будет создан файл ui\_mainwindow.py, который описывает, как нужно строить разработанный макет.

Далее на основе класса QMainWindow был создан класс MainWindow главного окна приложения. При инициализации этого окна происходит чтение файла ui\_mainwindow.py и создание на его основе пользовательского интерфейса. Здесь же происходит создание виджета для отображения графики glWidget, созданного на основе класса QOpenGLWidget. Помимо создания виджетов внутри MainWindow происходит привязка слотов к сигналам.

Так, например, при нажатии на кнопку «Выход» срабатывает функция leave , и осуществляется закрытие приложения.

При выборе определенного примитива и нажатии кнопки «Изобразить» вызовется функция draw, которая вызовет метод update у glWidget и тем самым осуществит отрисовку выбранного примитива с установленными параметрами.

При перемещении горизонтальных ползунков на изображении будут изменяться размеры точек, толщина линии и количество объектов. Информация о том, что изображение необходимо изменить, передается с помощью функций pointSizeChanged(), lineWidthChanged(), objectCounterChanged() соответственно. Стоит отметить, что изменение размера точек или толщины линии не приведет к генерации новых точек; работа будет производиться с уже имеющимся списком точек в двумерном пространстве.

В самом glWidget для удобства были добавлены следующие поля:

-primitive – содержит примитив, который нужно отобразить в данный момент

-minPoints – минимальное число точек, необходимое для построения текущего примитива

-makeNewPoints – флаг, указывающий, нужно ли генерировать новые точки, или нужно работать со старыми

-mw – ссылка на родительское окно (с ее помощью внутри класса glWidget можно получить параметры, заданные пользователем на ползунках)

-points – список точек, для каждого примитива и числа объектов генерируется случайным образом

-colors – список цветов, генерируется случайным образом.

Также класс обладает следующими перегруженными методами:

-initializeGL – настраивает состояние виджета. Вызывается один раз, после инициализации объекта glWidget.

-paintGL – непосредственно метод, отвечающий за отрисовку объектов. Сперва производится проверка на выбранный примитив, если он не задан, то отрисовка не выполняется. Далее проверяется флаг makeNewPoints – если он истинен, то происходит случайная генерация точек. Поскольку для отображения различных объектов требуется разное количество точек, то по умолчанию генерируется их теоретически максимально необходимое число (минимальное число точек, необходимое для построения одного примитива, умноженное на заданное пользователем число объектов), далее в зависимости от выбранного примитива сгенерированный список точек может быть обрезан до определенного числа точек в нем. После этого для каждой точки случайным образом генерируется свой цвет, который сохраняется в списке colors. Затем происходит установка значений размера точки и толщины линии (значения берутся из ползунков на главном окне) и начинается процесс рисования. В открывающую скобку glBegin передаем примитив, который мы хотим изобразить, а далее для каждой точки задаем ее цвет и рисуем эту точку с помощью функции glVertex2fv. Как только все точки будут отображены, процесс рисования заканчивается с помощью команды glEnd. Разработанный программный код см. в приложении А.

**Тестирование.**

Примеры получаемых примитивов изображены на рис. 4–13.

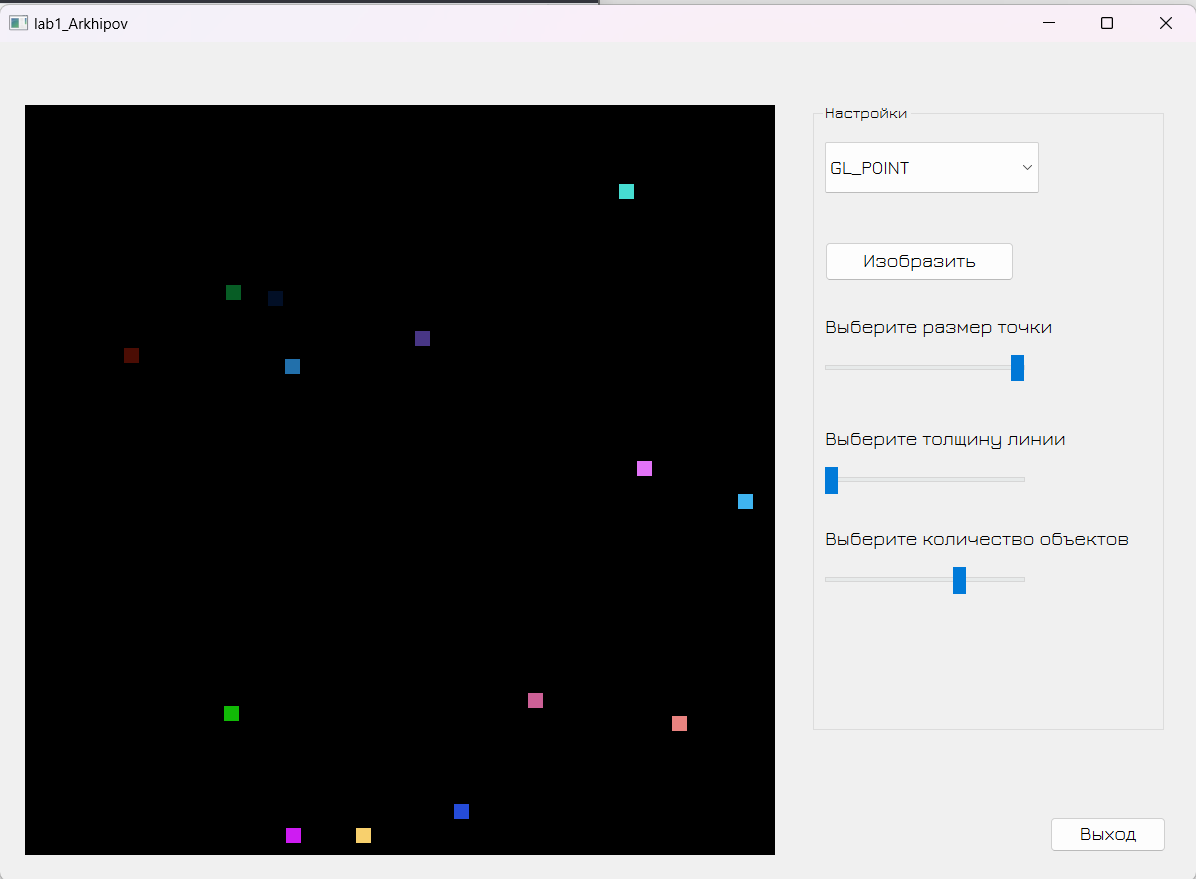


Рисунок 4 – примитив GL\_POINT

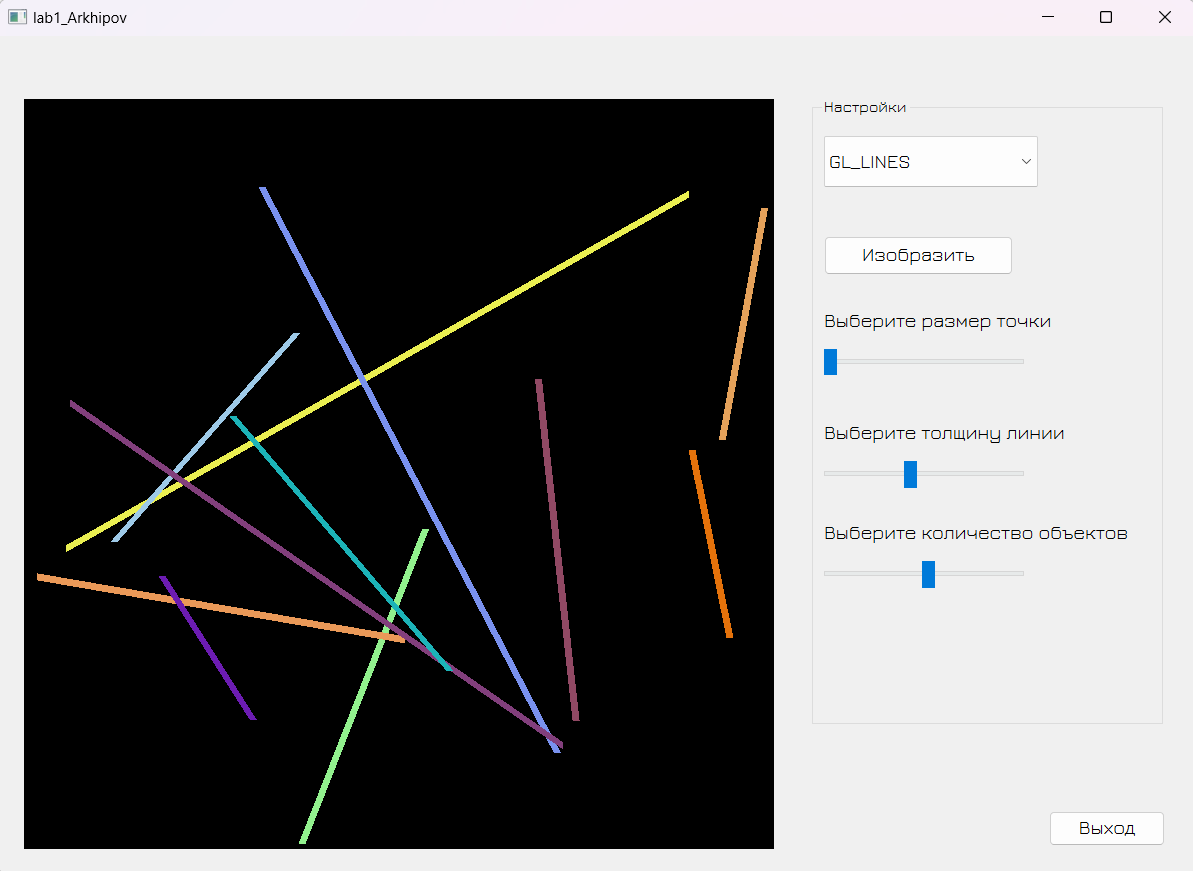


Рисунок 5 – примитив GL\_LINES

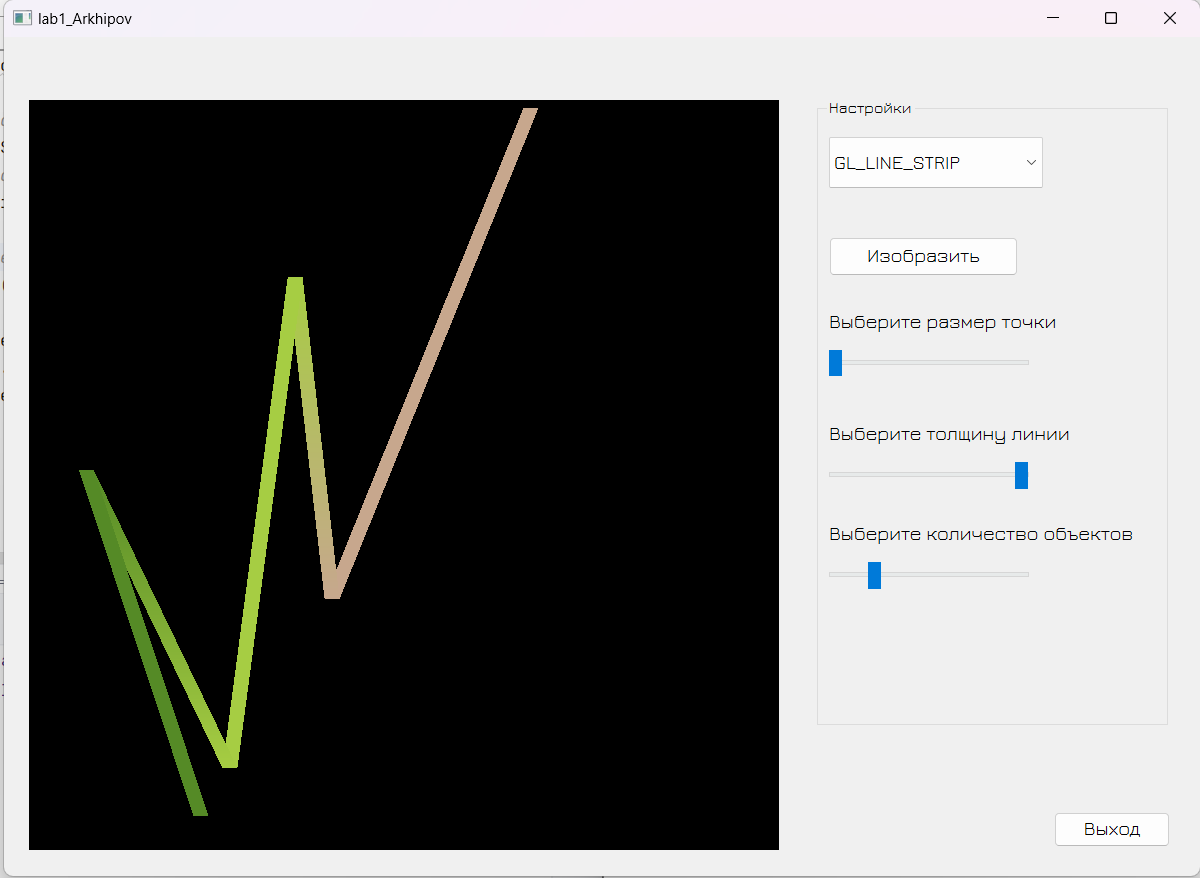


Рисунок 6 – примитив GL\_LINE\_STRIP

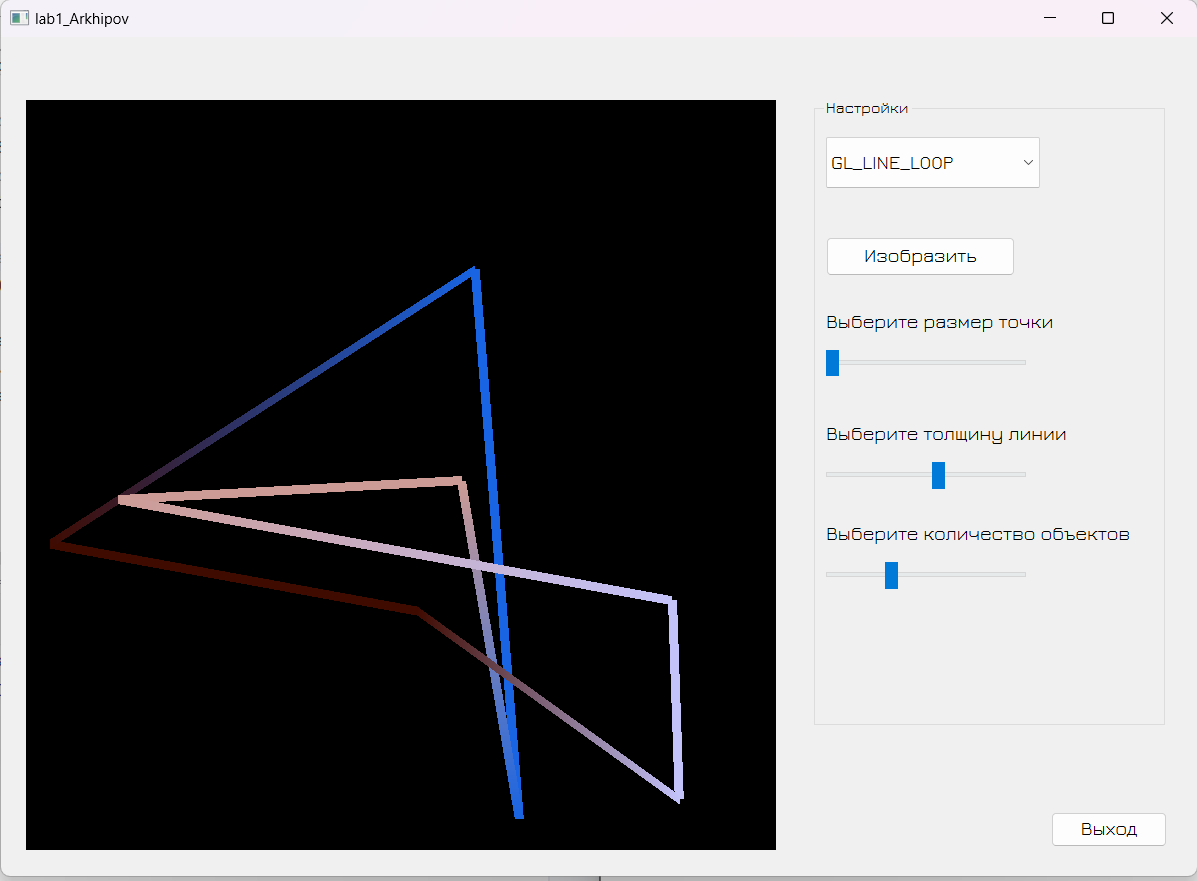


Рисунок 7 – примитив GL\_LINE\_LOOP

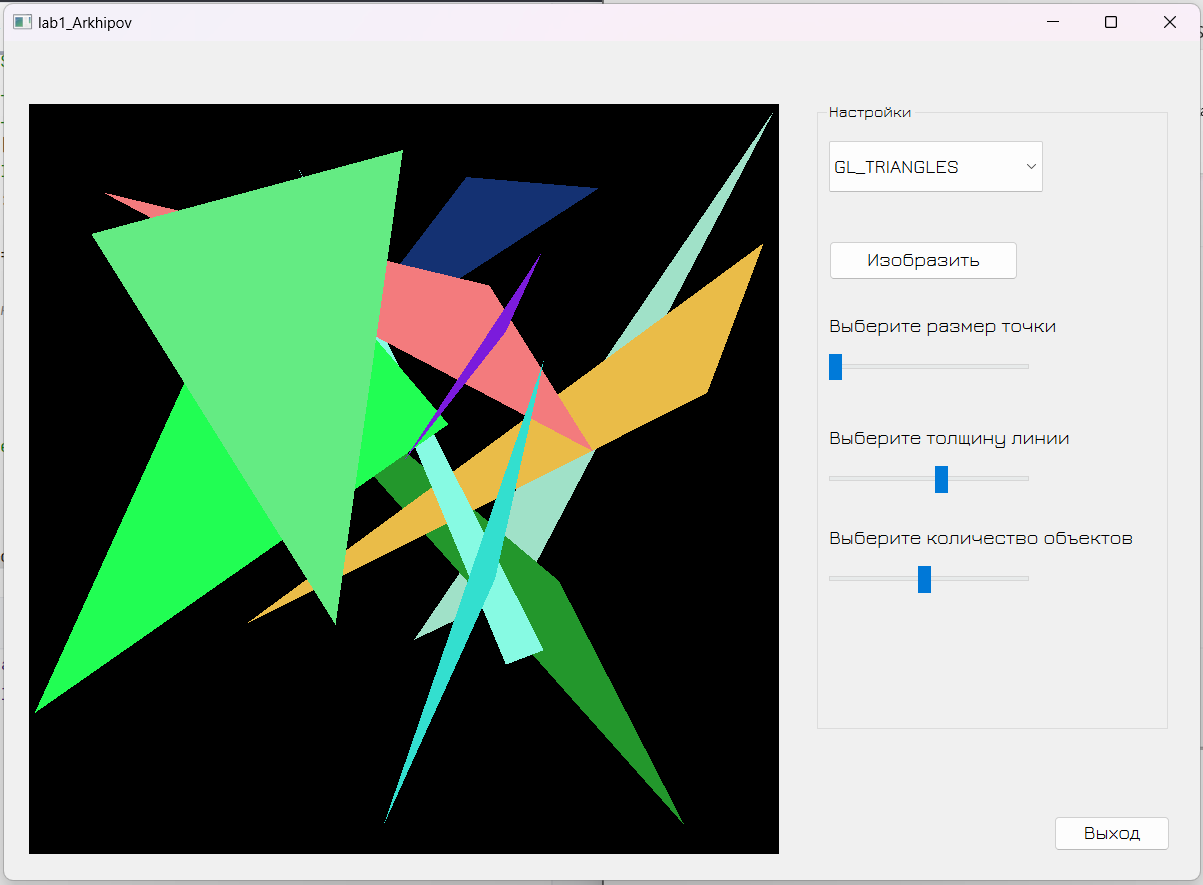


Рисунок 8 – примитив GL\_TRIANGLES

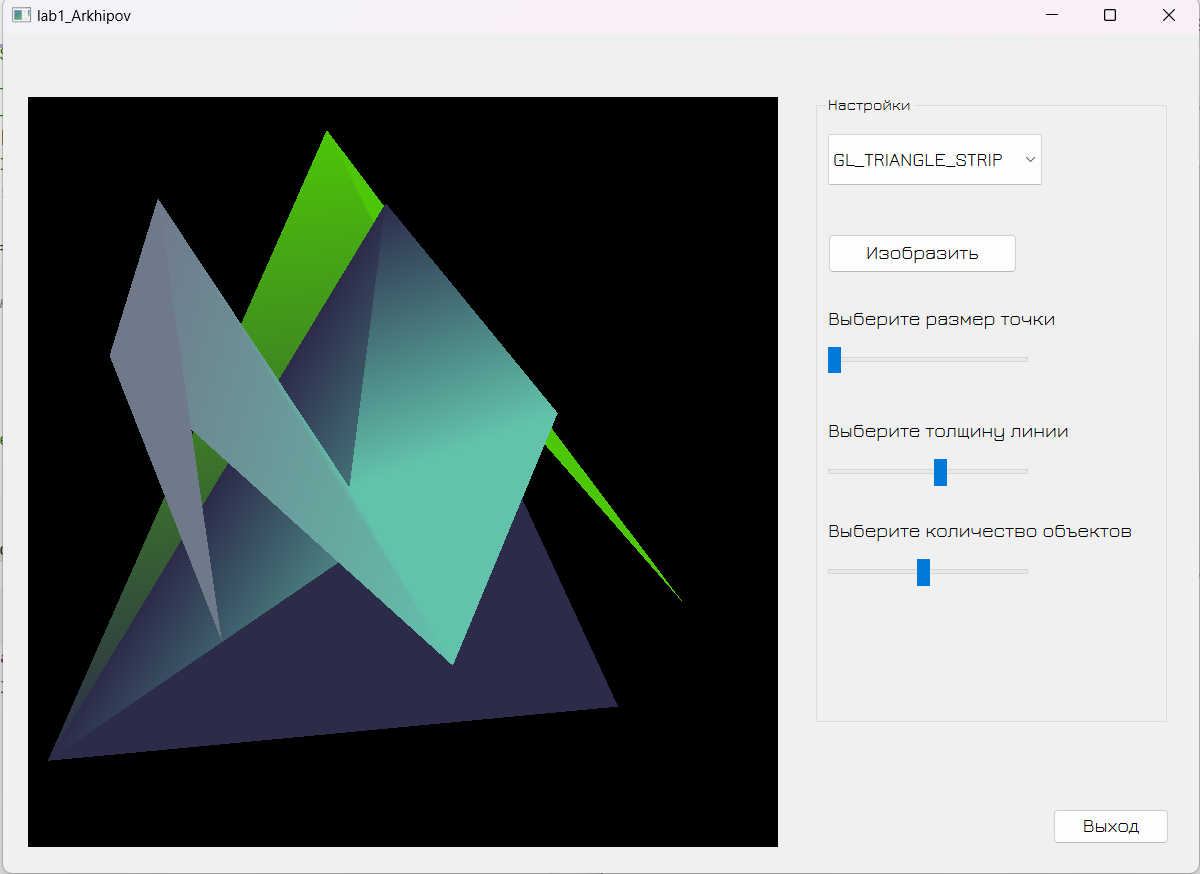


Рисунок 9 – примитив GL\_TRIANGLE\_STRIP

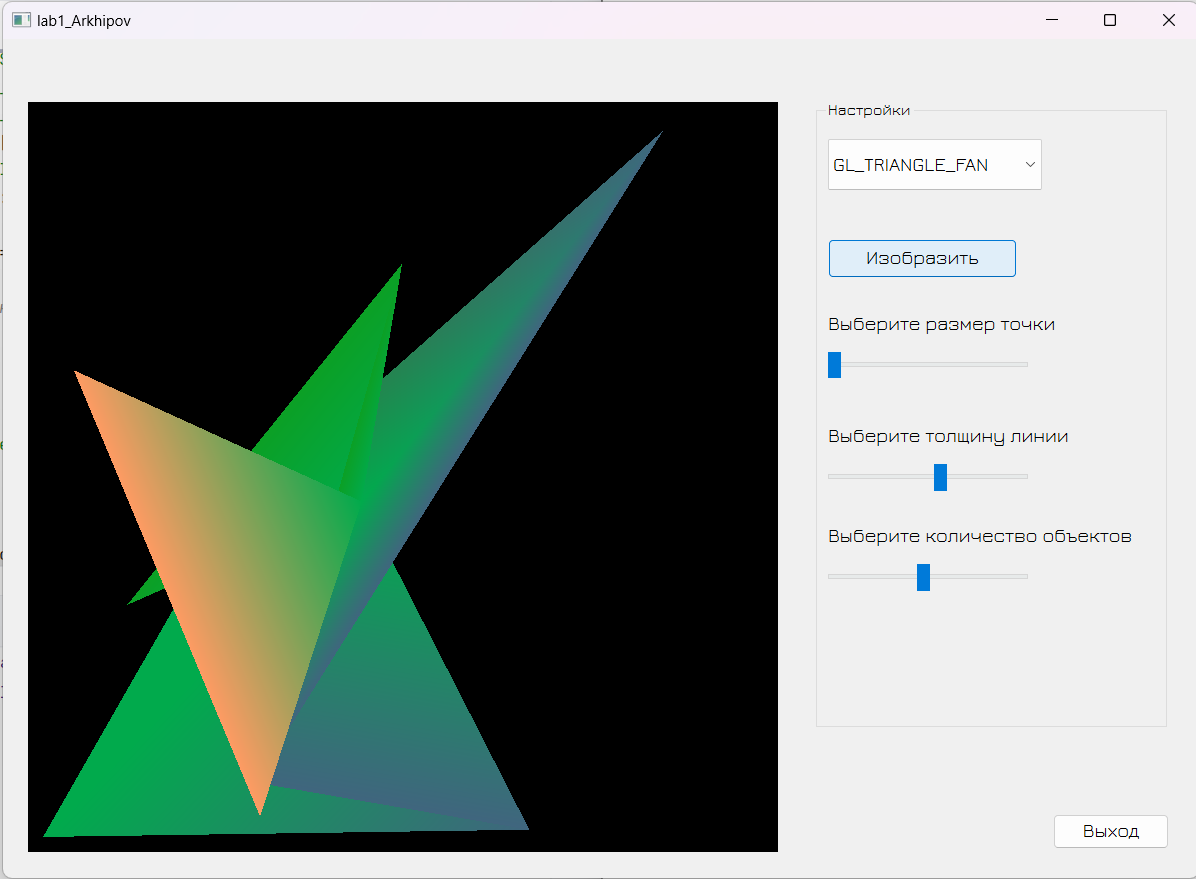


Рисунок 10 – примитив GL\_TRIANGLE\_FAN

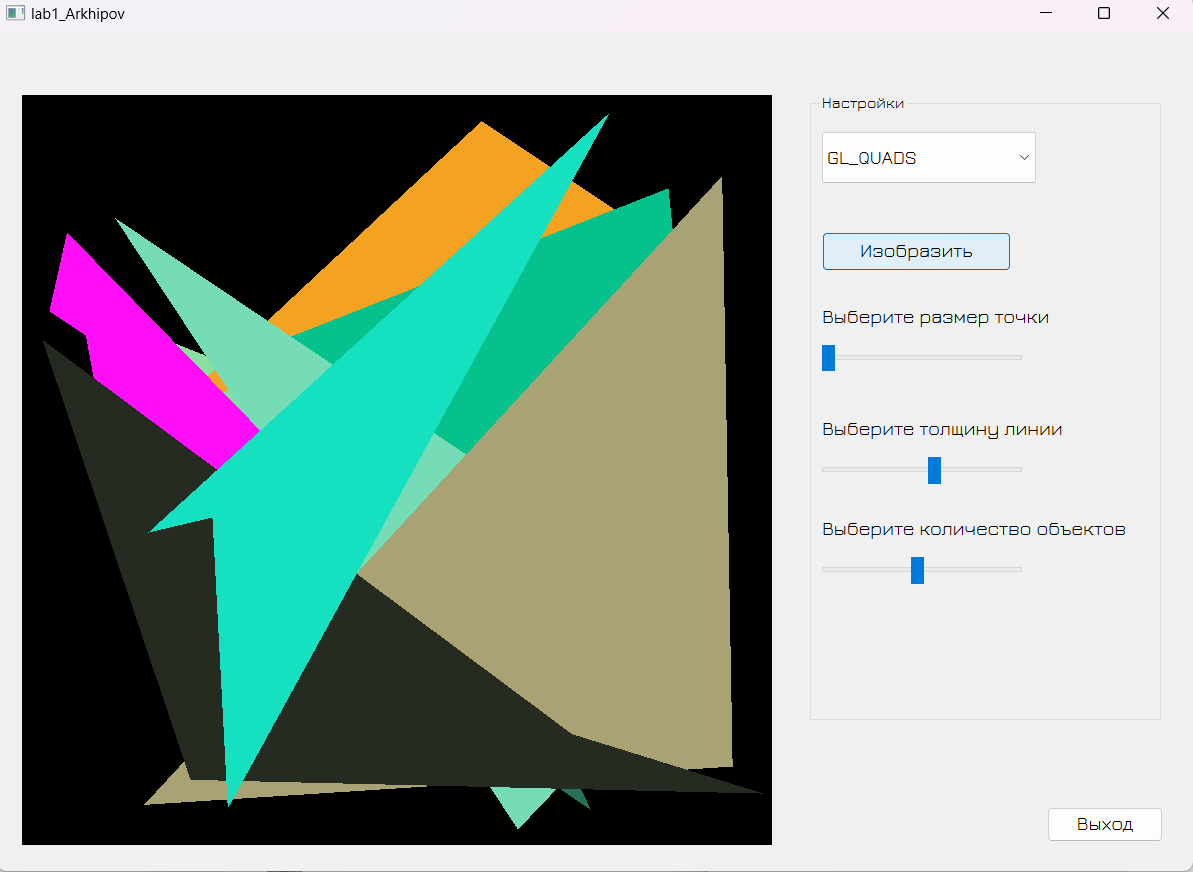


Рисунок 11 – примитив GL\_QUADS

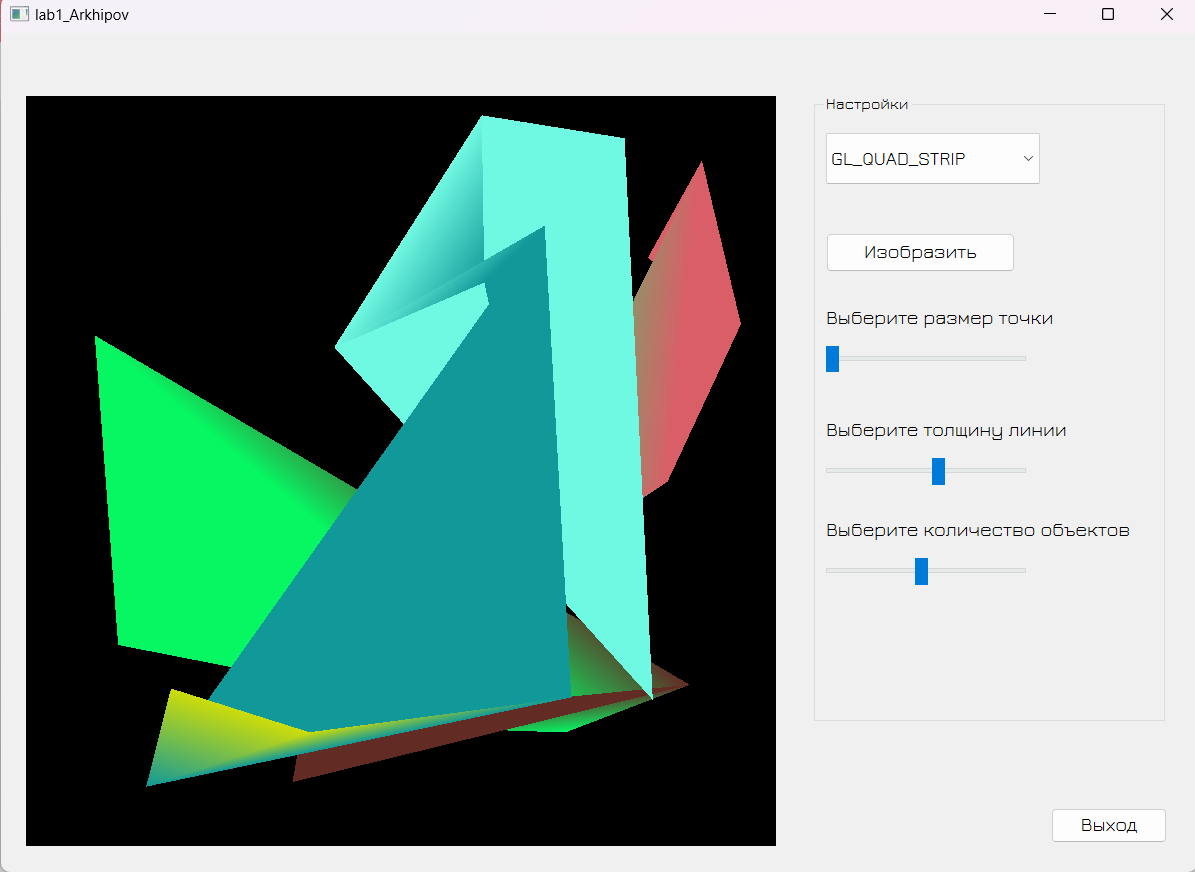


Рисунок 12 – примитив GL\_QUAD\_STRIP

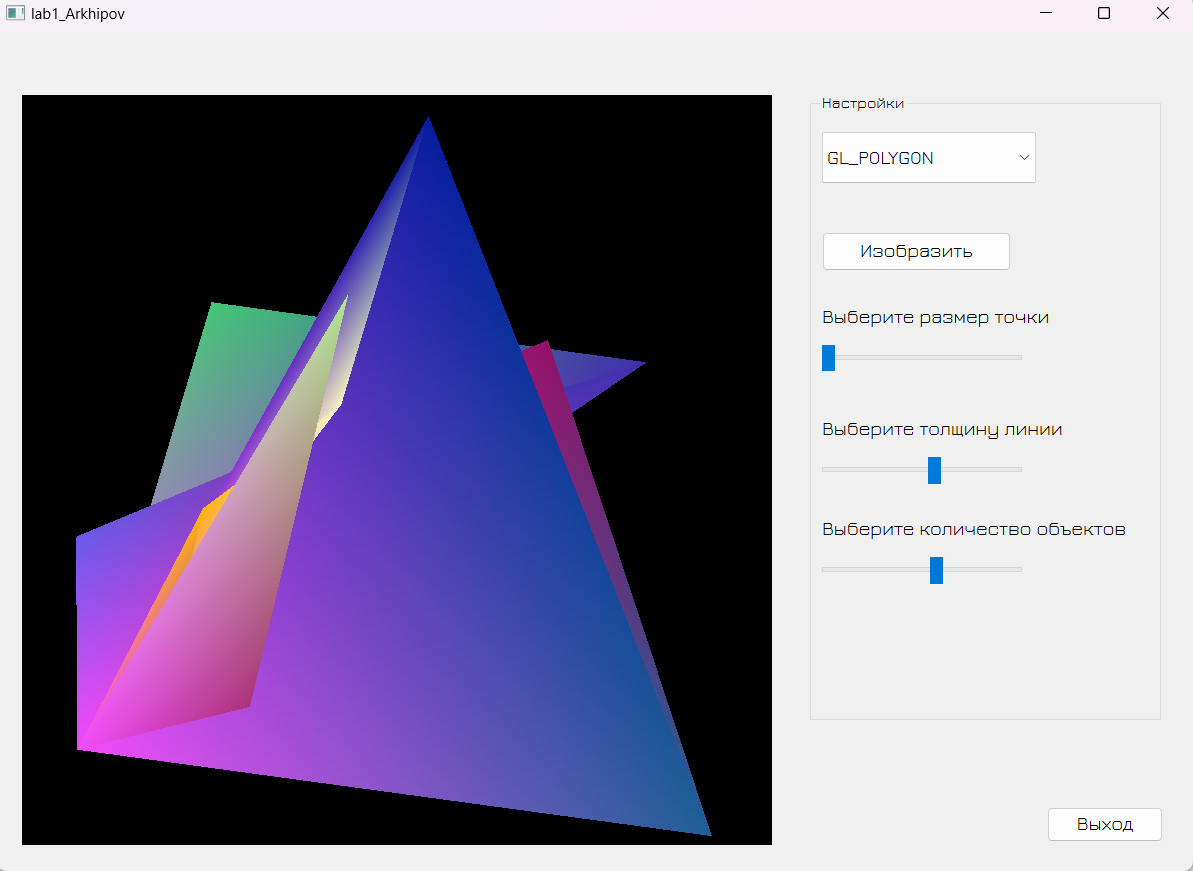


Рисунок 13 – примитив GL\_POLYGON

**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.

**ПРИЛОЖЕНИЕ А**

**РАЗРАБОТАННЫЙ ПРОГРАММНЫЙ КОД**

Название файла: glWidget.py

from PyQt6.QtOpenGLWidgets import QOpenGLWidget  
from OpenGL import GL as gl  
from PyQt6 import QtCore  
from random import uniform  
  
class glWidget(QOpenGLWidget):  
 def \_\_init\_\_(self, main\_window):  
 super().\_\_init\_\_(parent=main\_window.ui.centralwidget)  
 *# Примитив, который будет нарисован* self.primitive = None  
 *# Минимальное количество точек, которое может быть в примитиве* self.minPoints = None  
 *# Флаг, содержащий информацию о том, нужно ли вычислять точки заново* self.makeNewPoints = False  
 *# Ссылка на родительское окно* self.mw = main\_window  
  
 main\_window.ui.openGLWidget = self  
 main\_window.ui.openGLWidget.setGeometry(QtCore.QRect(20, 50, 600, 600))  
 main\_window.ui.openGLWidget.setObjectName("openGLWidget")  
  
 *# Настройка состояния. Вызывается один раз в самом начале* def initializeGL(self):  
 gl.glClearColor(0, 0, 0, 1)  
 gl.glClear(gl.GL\_COLOR\_BUFFER\_BIT | gl.GL\_DEPTH\_BUFFER\_BIT)  
  
 def paintGL(self):  
 if self.primitive != None:  
 *# Генерация новых точек, если это необходимо* if self.makeNewPoints:  
 objectsCounter = self.mw.ui.horizontalSlider\_3.value()  
 self.points = [[uniform(-0.99,0.99), uniform(-0.99, 0.99)] for \_ in range(objectsCounter \* self.minPoints)]  
  
 *# Обрезание массива точек* if self.primitive in [gl.GL\_LINE\_STRIP, gl.GL\_LINE\_LOOP]:  
 self.points = self.points[:objectsCounter + 1]  
 elif self.primitive in [gl.GL\_TRIANGLE\_STRIP, gl.GL\_TRIANGLE\_FAN]:  
 self.points = self.points[:objectsCounter + 2]  
 elif self.primitive in [gl.GL\_QUAD\_STRIP]:  
 self.points = self.points[: 2 \* (objectsCounter + 1)]  
  
 *# Генерация цветов точек* self.colors = [[uniform(0, 1) for i in range(3)] for \_ in range(len(self.points))]  
  
 *# Установка размера точки* gl.glPointSize(self.mw.ui.horizontalSlider.value())  
 *# Установка толщины линии* gl.glLineWidth(self.mw.ui.horizontalSlider\_2.value())  
  
 *# Рисование* gl.glBegin(self.primitive)  
 for index, point in enumerate(self.points):  
 if index % self.minPoints == 0:  
 gl.glColor3fv(self.colors[index])  
 gl.glVertex2fv(point)  
 gl.glEnd()

Название файла: MainWindow.py

import sys  
from PyQt6.QtWidgets import QMainWindow  
from glWidget import glWidget  
from ui\_mainwindow import Ui\_MainWindow  
from OpenGL import GL as gl  
  
  
class MainWindow(QMainWindow):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 *# Создание виджета для отображения графики* self.ui.openGLWidget = glWidget(self)  
  
 *# Подписка элементов интерфейса на события* self.ui.pushButton.clicked.connect(self.leave)  
 self.ui.pushButton\_2.clicked.connect(self.draw)  
 self.ui.horizontalSlider.valueChanged.connect(self.pointSizeChanged)  
 self.ui.horizontalSlider\_2.valueChanged.connect(self.lineWidthChanged)  
 self.ui.horizontalSlider\_3.valueChanged.connect(self.objectCounterChanged)  
  
 *# Создание словаря примитивов и минимального числа точек, необходимого для их построения* self.functions = {  
 "GL\_POINT": [gl.GL\_POINTS, 1],  
 "GL\_LINES": [gl.GL\_LINES, 2],  
 "GL\_LINE\_STRIP": [gl.GL\_LINE\_STRIP, 2],  
 "GL\_LINE\_LOOP": [gl.GL\_LINE\_LOOP, 2],  
 "GL\_TRIANGLES": [gl.GL\_TRIANGLES, 3],  
 "GL\_TRIANGLE\_STRIP": [gl.GL\_TRIANGLE\_STRIP, 3],  
 "GL\_TRIANGLE\_FAN": [gl.GL\_TRIANGLE\_FAN, 3],  
 "GL\_QUADS": [gl.GL\_QUADS, 4],  
 "GL\_QUAD\_STRIP": [gl.GL\_QUAD\_STRIP, 4],  
 "GL\_POLYGON": [gl.GL\_POLYGON, 1]  
 }  
 self.dictionary = {count: elem for count, elem in enumerate(self.functions.keys())}  
  
 *# Демонстрация окна* self.show()  
  
  
 def leave(self):  
 print('You entered "Quit" button')  
 sys.exit()  
  
 def draw(self):  
 option = self.dictionary[self.ui.comboBox.currentIndex()]  
 self.ui.openGLWidget.primitive, self.ui.openGLWidget.minPoints = self.functions[option]  
 self.ui.openGLWidget.makeNewPoints = True  
 self.ui.openGLWidget.update()  
  
 def pointSizeChanged(self):  
 self.ui.openGLWidget.makeNewPoints = False  
 self.ui.openGLWidget.update()  
  
  
 def lineWidthChanged(self):  
 self.ui.openGLWidget.makeNewPoints = False  
 self.ui.openGLWidget.update()  
  
 def objectCounterChanged(self):  
 self.ui.openGLWidget.makeNewPoints = True  
 self.ui.openGLWidget.update()

Название файла: main.py

import sys  
from PyQt6.QtWidgets import QApplication  
from MainWindow import MainWindow  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 main\_window = MainWindow()  
 main\_window.setFixedSize(957, 671)  
 sys.exit(app.exec())

Название файла: ui\_mainwindow.py

*# Form implementation generated from reading ui file 'ui\_mainwindow.ui'  
#  
# Created by: PyQt6 UI code generator 6.4.2  
#  
# WARNING: Any manual changes made to this file will be lost when pyuic6 is  
# run again. Do not edit this file unless you know what you are doing.*from PyQt6 import QtCore, QtGui, QtWidgets  
from PyQt6.QtOpenGLWidgets import QOpenGLWidget  
from glWidget import glWidget  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 self.openGLWidget = None  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(957, 671)  
 self.centralwidget = QtWidgets.QWidget(parent=MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.pushButton = QtWidgets.QPushButton(parent=self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(840, 620, 93, 28))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.pushButton.setFont(font)  
 self.pushButton.setObjectName("pushButton")  
 self.groupBox = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox.setGeometry(QtCore.QRect(650, 50, 281, 501))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(10)  
 self.groupBox.setFont(font)  
 self.groupBox.setObjectName("groupBox")  
 self.comboBox = QtWidgets.QComboBox(parent=self.groupBox)  
 self.comboBox.setGeometry(QtCore.QRect(10, 30, 171, 41))  
 font = QtGui.QFont()  
 font.setPointSize(12)  
 self.comboBox.setFont(font)  
 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.pushButton\_2 = QtWidgets.QPushButton(parent=self.groupBox)  
 self.pushButton\_2.setGeometry(QtCore.QRect(10, 110, 151, 31))  
 font = QtGui.QFont()  
 font.setPointSize(12)  
 self.pushButton\_2.setFont(font)  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.horizontalSlider = QtWidgets.QSlider(parent=self.groupBox)  
 self.horizontalSlider.setGeometry(QtCore.QRect(10, 200, 160, 22))  
 self.horizontalSlider.setMinimum(1)  
 self.horizontalSlider.setMaximum(15)  
 self.horizontalSlider.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.horizontalSlider.setObjectName("horizontalSlider")  
 self.horizontalSlider\_2 = QtWidgets.QSlider(parent=self.groupBox)  
 self.horizontalSlider\_2.setGeometry(QtCore.QRect(10, 290, 160, 22))  
 self.horizontalSlider\_2.setMinimum(1)  
 self.horizontalSlider\_2.setMaximum(15)  
 self.horizontalSlider\_2.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.horizontalSlider\_2.setObjectName("horizontalSlider\_2")  
 self.horizontalSlider\_3 = QtWidgets.QSlider(parent=self.groupBox)  
 self.horizontalSlider\_3.setGeometry(QtCore.QRect(10, 370, 160, 22))  
 self.horizontalSlider\_3.setMinimum(1)  
 self.horizontalSlider\_3.setMaximum(20)  
 self.horizontalSlider\_3.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.horizontalSlider\_3.setObjectName("horizontalSlider\_3")  
 self.label\_2 = QtWidgets.QLabel(parent=self.groupBox)  
 self.label\_2.setGeometry(QtCore.QRect(10, 170, 221, 16))  
 font = QtGui.QFont()  
 font.setPointSize(12)  
 self.label\_2.setFont(font)  
 self.label\_2.setObjectName("label\_2")  
 self.label\_3 = QtWidgets.QLabel(parent=self.groupBox)  
 self.label\_3.setGeometry(QtCore.QRect(10, 260, 221, 16))  
 font = QtGui.QFont()  
 font.setPointSize(12)  
 self.label\_3.setFont(font)  
 self.label\_3.setObjectName("label\_3")  
 self.label\_4 = QtWidgets.QLabel(parent=self.groupBox)  
 self.label\_4.setGeometry(QtCore.QRect(10, 340, 261, 16))  
 font = QtGui.QFont()  
 font.setPointSize(12)  
 self.label\_4.setFont(font)  
 self.label\_4.setObjectName("label\_4")  
 MainWindow.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "lab1\_Arkhipov"))  
 self.pushButton.setText(\_translate("MainWindow", "Выход"))  
 self.groupBox.setTitle(\_translate("MainWindow", "Настройки"))  
 self.comboBox.setItemText(0, \_translate("MainWindow", "GL\_POINT"))  
 self.comboBox.setItemText(1, \_translate("MainWindow", "GL\_LINES"))  
 self.comboBox.setItemText(2, \_translate("MainWindow", "GL\_LINE\_STRIP"))  
 self.comboBox.setItemText(3, \_translate("MainWindow", "GL\_LINE\_LOOP"))  
 self.comboBox.setItemText(4, \_translate("MainWindow", "GL\_TRIANGLES"))  
 self.comboBox.setItemText(5, \_translate("MainWindow", "GL\_TRIANGLE\_STRIP"))  
 self.comboBox.setItemText(6, \_translate("MainWindow", "GL\_TRIANGLE\_FAN"))  
 self.comboBox.setItemText(7, \_translate("MainWindow", "GL\_QUADS"))  
 self.comboBox.setItemText(8, \_translate("MainWindow", "GL\_QUAD\_STRIP"))  
 self.comboBox.setItemText(9, \_translate("MainWindow", "GL\_POLYGON"))  
 self.pushButton\_2.setText(\_translate("MainWindow", "Изобразить"))  
 self.label\_2.setText(\_translate("MainWindow", "Выберите размер точки"))  
 self.label\_3.setText(\_translate("MainWindow", "Выберите толщину линии"))  
 self.label\_4.setText(\_translate("MainWindow", "Выберите количество объектов"))