# 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

Выполнила: Сергеев А.Д.

Факультет: ФКТИ

Группа: 8304

Преподаватель: Герасимова Т.В.

Санкт-Петербург 2021

# Задание.

Разработать программу, реализующую представление определенного набора примитивов 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 шаблона программа должна быть пополнена возможностями остановки интерактивно различных атрибутов примитивов рисования через вызов соответствующих элементов интерфейса пользователя.

# 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).

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 среде разработки PyCharm при помощи языка программирования Python. Была использована библиотека PyOpenGl, а также PyQt6 для создания пользовательского интерфейса.

Изначально область, на которой производится отрисовка вершин, содержит 6 вершин, расположенных в форме правильного шестиугольника. При нажатии на ней появляется новая вершина в точке нажатия. Помимо этой области окно программы содержит выпадающий список, состоящий из названий графических примитивов, которые можно применить. Выбранный в списке примитив сразу же отображается. Также была создана кнопка *Reset*, удаляющая все вершины с полотна.

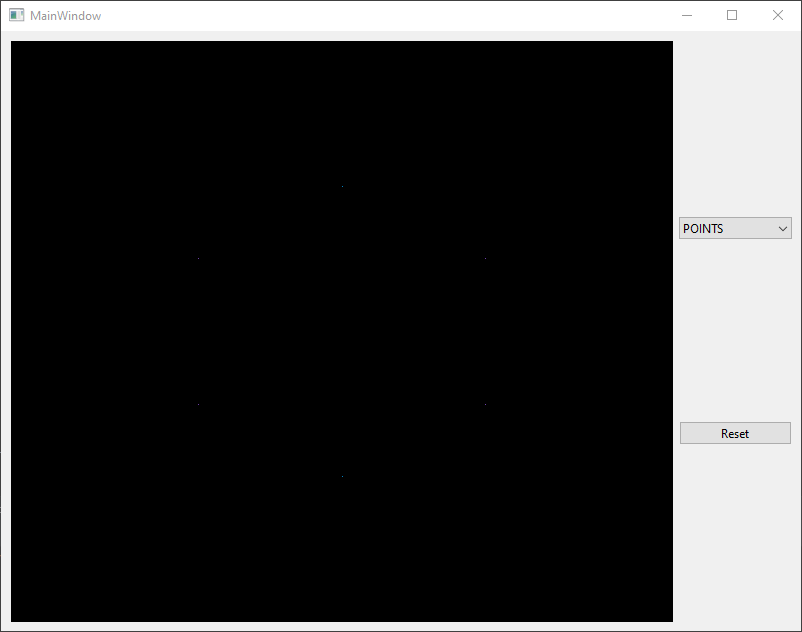


Рисунок 1 - GL\_POINTS

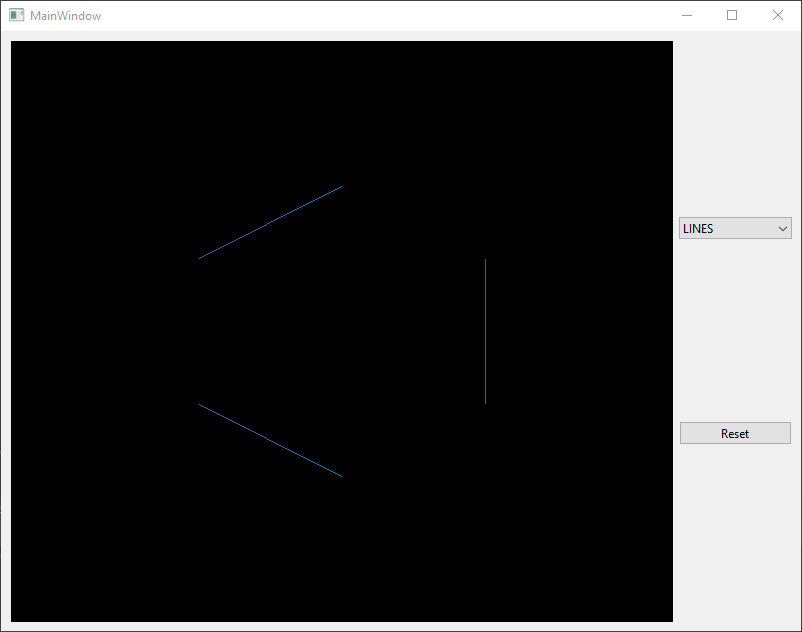


Рисунок 2 - GL\_LINES



Рисунок 3 - GL\_LINE\_STRIP

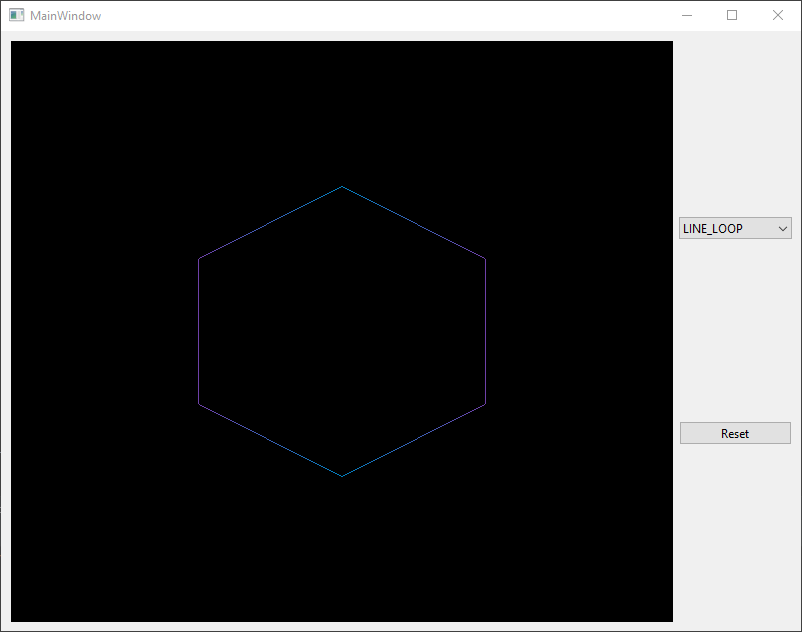


Рисунок 4 - GL\_LINE\_LOOP

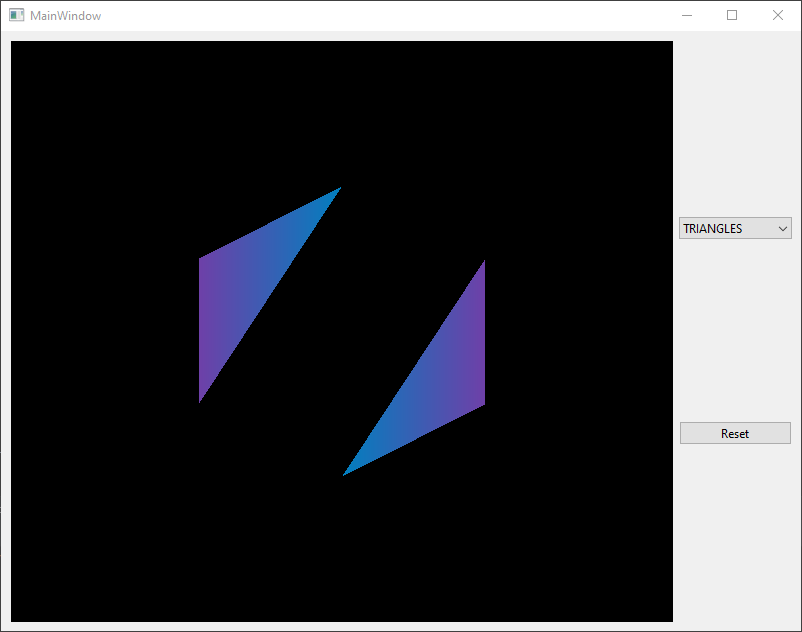


Рисунок 5 - GL\_TRIANGLES

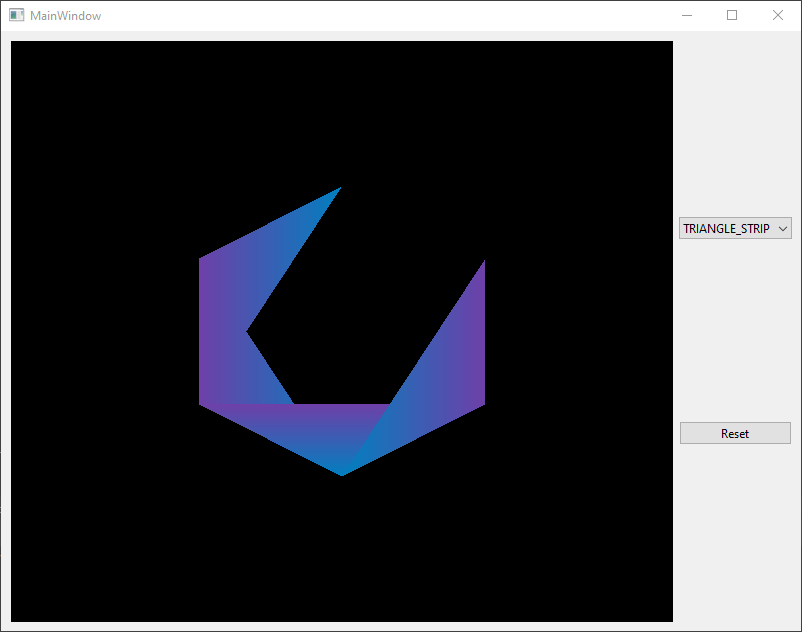


Рисунок 6 - GL\_TRIANGLE\_STRIP

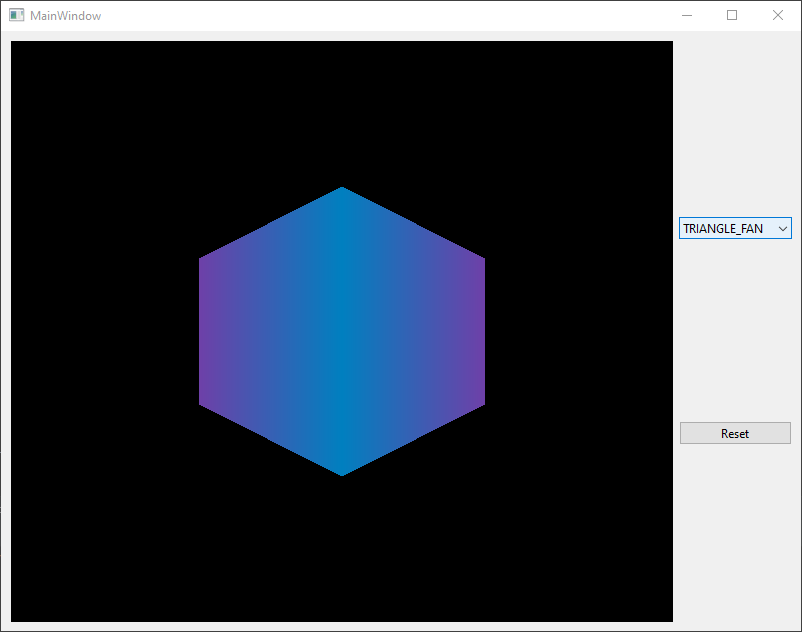


Рисунок 7 - GL\_TRIANGLE\_FAN

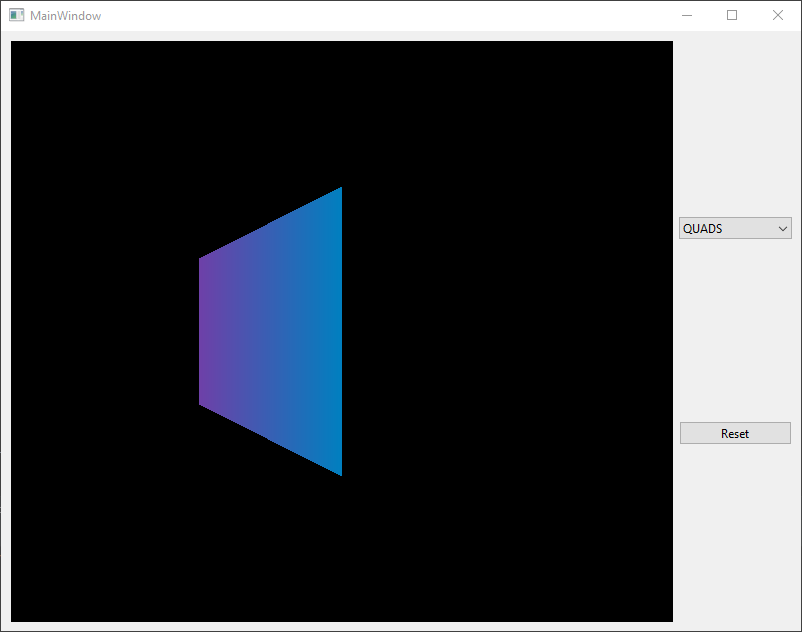


Рисунок 8 - GL\_QUADS

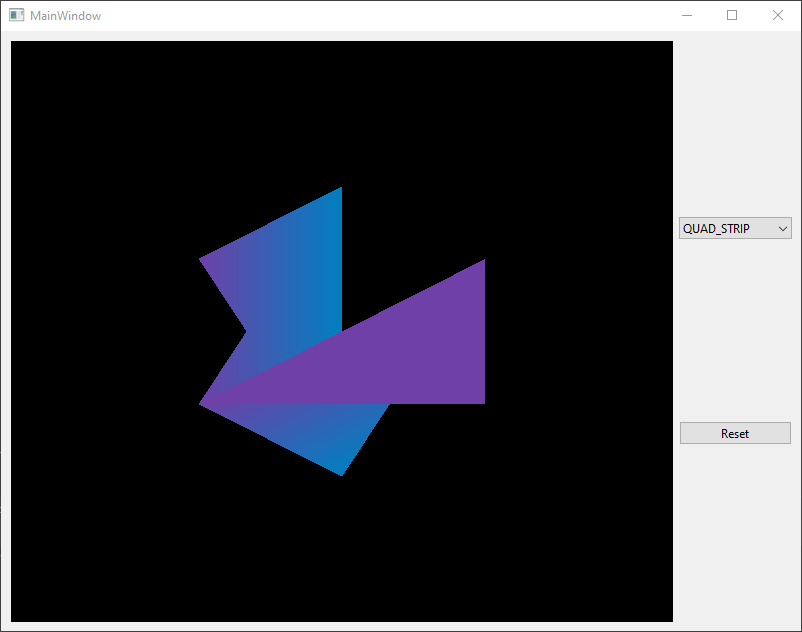


Рисунок 9 - GL\_QUAD\_STRIP

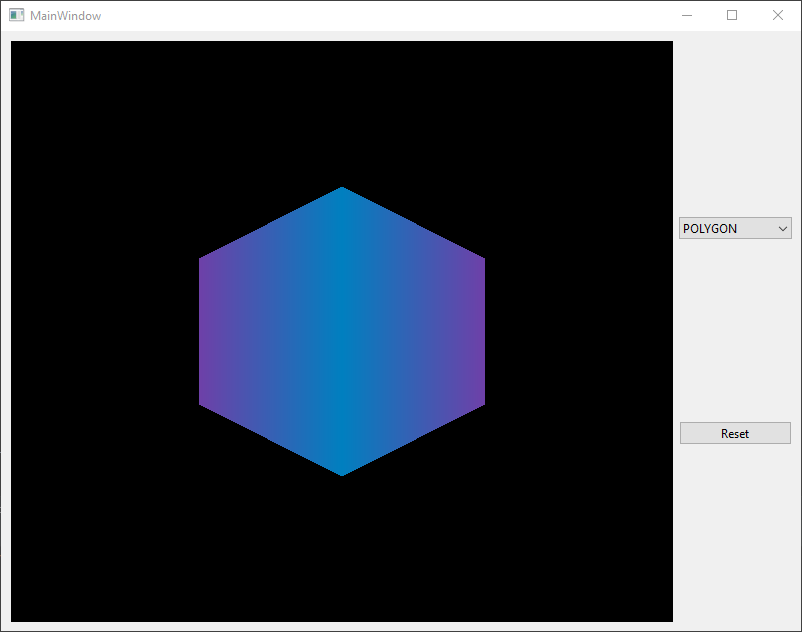


Рисунок 10 - GL\_POLYGON

**Выводы**

В результате выполнения лабораторной работы была разработана программа, реализующая отрисовку графических примитивов OpenGL. При тестировании ошибок выявлено не было. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.

**Приложение А. Исходный код программы.**

Файл *main.py*:

import sys

from enum import Enum

from OpenGL.GL import \*

from PyQt6 import QtWidgets, uic

class Mode(Enum):

POINTS = GL\_POINTS

LINES = GL\_LINES

LINE\_STRIP = GL\_LINE\_STRIP

LINE\_LOOP = GL\_LINE\_LOOP

TRIANGLES = GL\_TRIANGLES

TRIANGLE\_STRIP = GL\_TRIANGLE\_STRIP

TRIANGLE\_FAN = GL\_TRIANGLE\_FAN

QUADS = GL\_QUADS

QUAD\_STRIP = GL\_QUAD\_STRIP

POLYGON = GL\_POLYGON

def configure\_window(win):

mode\_box = win.modeBox

display = win.mainGLWidget

reset = win.resetButton

mode\_box.currentIndexChanged.connect(lambda index: display.set\_mode(mode\_box.itemData(index)))

for mode in Mode:

mode\_box.addItem(mode.name, mode.value)

reset.clicked.connect(lambda: display.clear\_vertexes())

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

window = uic.loadUi('main.ui')

configure\_window(window)

window.show()

sys.exit(app.exec())

Файл *GLWidget.py*:

import math

from OpenGL.GL import \*

from PyQt6.QtOpenGLWidgets import QOpenGLWidget

def \_blues(x, y):

return (2 - abs(x) - abs(y)) / 2

class GlWidget(QOpenGLWidget):

def \_\_init\_\_(self, parent=None):

def combinator(grad):

rad = math.radians(grad)

mono\_x = math.cos(rad) / 2

mono\_y = math.sin(rad) / 2

return mono\_x, mono\_y, \_blues(mono\_x, mono\_y)

QOpenGLWidget.\_\_init\_\_(self, parent)

self.\_vert = [combinator(x) for x in range(90, 450, 60)]

self.\_mode = GL\_POINTS

def paintGL(self):

glClear(GL\_COLOR\_BUFFER\_BIT)

glBegin(self.\_mode)

for vertex in self.\_vert:

glColor3f(abs(vertex[0]), abs(vertex[1]), abs(vertex[2]))

glVertex2f(vertex[0], vertex[1])

glEnd()

def mousePressEvent(self, event):

center\_w = self.width() / 2

center\_h = self.height() / 2

event\_x = event.position().x() - center\_w

event\_y = event.position().y() - center\_h

gl\_x = event\_x / center\_w

gl\_y = -event\_y / center\_h

self.\_vert.append((gl\_x, gl\_y, \_blues(gl\_x, gl\_y)))

self.update()

def set\_mode(self, mode):

self.\_mode = mode

self.update()

def clear\_vertexes(self):

self.\_vert = []

self.update()

Файл *main.ui*:

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QWidget" name="horizontalLayoutWidget">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>781</width>

<height>581</height>

</rect>

</property>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="GlWidget" name="mainGLWidget">

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

</widget>

</item>

<item>

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<widget class="QComboBox" name="modeBox"/>

</item>

<item>

<widget class="QPushButton" name="resetButton">

<property name="text">

<string>Reset</string>

</property>

</widget>

</item>

</layout>

</item>

</layout>

</widget>

</widget>

</widget>

<customwidgets>

<customwidget>

<class>GlWidget</class>

<extends>QOpenGLWidget</extends>

<header location="global">GLWidget</header>

</customwidget>

</customwidgets>

<resources/>

<connections/>

</ui>