Санкт-Петербургский Государственный

Электротехнический Университет

Кафедра МОЭВМ

Задание для лабораторной работы № 2

"Примитивы OpenGL"

| Студенты гр. 1384 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Усачева Д.В.  Пчелинцева К.Р. |
| --- | --- | --- |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Герасимова Т.В. |

Санкт-Петербург

2024 г.

**Задание**

На базе разработанной вами оболочки из 1 работы разработать программу реализующую представление тестов отсечения ( glScissor), прозрачности (glAlphaFunc), смешения цветов (glBlendFunc) в библиотеке OpenGL на базе разработанных вами в предыдущей работе примитивов.

Разработанная на базе шаблона программа должна быть пополнена возможностями установки интерактивно различных атрибутов тестов через вызов соответствующих элементов интерфейса пользователя

**Общие сведения**

Управление режимами работы в OpenGL осуществляется при помощи двух команд - glEnable и glDisable, одна из которых включает, а вторая выключает некоторый режим.

void glEnable(GLenum cap)

void glDisable(GLenum cap)

Обе команды имеют один аргумент – сар, который может принимать значения определяющие тот или иной режим, например, GL\_ALPHA\_TEST, GL\_BLEND, GL\_SCISSOR\_TEST и многие другие.

*Тест отсечения*

Режим GL\_SCISSOR\_TEST разрешает отсечение тех фрагментов объекта, которые находятся вне прямоугольника "вырезки".

Прямоугольник "вырезки" определяется функцией glScissor:

**void glScissor( GLint x, GLint y, GLsizei width, GLsizei height );**

где параметры

* x, y определяют координаты левого нижнего угла прямоугольника «вырезки», исходное значение - (0,0).
* width, height - ширина и высота прямоугольника «вырезки».

В приведенном ниже фрагменте программы реализуется тест отсечения. Сначала изображается группа связных отрезков не используя режим отсечения, а затем включается этот режим.

glEnable(GL\_SCISSOR\_TEST);

InitViewport(0, windH\*2/3, vpW, vpH);

glScissor(0,windH\*2/3,vpW/2,vpH/2);

Triangles();

Quads();

glDisable(GL\_SCISSOR\_TEST);

InitViewport(windW/3, windH\*2/3, vpW, vpH);

glScissor(windW/3,windH\*2/3,vpW/2,vpH/2);

Triangles();

Quads();

*Тест прозрачности*

Режим GL\_ALPHA\_TEST задает тестирование по цветовому параметру альфа.Функция glAlphaFunc устанавливает функцию тестирования параметра альфа.

**void glAlphaFunc( GLenum func, GLclampf ref )**

где параметр – func может принимать следующие значения:

GL\_NEVER – никогда не пропускает

GL\_LESS – пропускает, если входное значение альфа меньше, чем значение ref

GL\_EQUAL – пропускает, если входное значение альфа равно значению ref

GL\_LEQUAL – пропускает, если входное значение альфа меньше или равно значения ref

GL\_GREATER – пропускает, если входное значение альфа больше, чем значение ref

GL\_NOTEQUAL – пропускает, если входное значение альфа не равно значению ref

GL\_GEQUAL – пропускает, если входное значение альфа больше или равно значения ref

GL\_ALWAYS – всегда пропускается, по умолчанию, а параметр ref – определяет значение, с которым сравнивается входное значение альфа.

Он может принимать значение от 0 до 1, причем 0 представляет наименьшее возможное значение альфа, а 1 – наибольшее. По умолчанию ref равен 0.

В приведенном ниже фрагменте программы реализуется тест прозрачности

glEnable(GL\_ALPHA\_TEST);

InitViewport(windW\*2/3, windH\*2/3, vpW, vpH);

glAlphaFunc(GL\_LESS, 0.7f);

Triangles();

Quads();

InitViewport(0, windH/3, vpW, vpH);

glAlphaFunc(GL\_GREATER, 0.7f);

Triangles();

Quads();

glDisable(GL\_ALPHA\_TEST);

*Тест смешения цветов*

Режим GL\_BLEND разрешает смешивание поступающих значений цветов RGBA со значениями, находящимися в буфере цветов.

Функция glBlendFunc устанавливает пиксельную арифметику.

**void glBlendFunc( GLenum sfactor, GLenum dfactor );**

где параметры

* sfactor устанавливает способ вычисления входящих факторов смешения RGBA. Может принимать одно из следующих значений – GL\_ZERO, GL\_ONE, GL\_DST\_COLOR, GL\_ONE\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_ONE\_MINUS\_DST\_ALPHA и GL\_SRC\_ALPHA\_SATURATE.
* dfactor устанавливает способ вычисления факторов смешения RGBA, уже находящихся в буфере кадра. Может принимать одно из следующих значений – GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA и GL\_ONE\_MINUS\_DST\_ALPHA.

В приведенном ниже фрагменте программы реализуется тест смешения

glEnable(GL\_BLEND);

InitViewport(windW/3, windH/3, vpW, vpH);

glBlendFunc(GL\_ONE, GL\_ZERO);

Triangles();

Quads();

InitViewport(windW\*2/3, windH/3, vpW, vpH);

glBlendFunc(GL\_ONE, GL\_ONE);

Triangles();

Quads();

InitViewport(0, 0, vpW, vpH);

glBlendFunc(GL\_ONE, GL\_SRC\_COLOR);

Triangles();

Quads();

InitViewport(windW/3, 0, vpW, vpH);

glBlendFunc(GL\_ONE, GL\_ONE\_MINUS\_SRC\_COLOR);

Triangles();

Quads();

InitViewport(windW\*2/3, 0, vpW, vpH);

glBlendFunc(GL\_ZERO, GL\_ONE\_MINUS\_SRC\_COLOR);

Triangles();

Quads();

Прозрачность лучше организовывать используя команду glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA). Такой же вызов применяют для устранения ступенчатости линий и точек. Для устранения ступенчатости многоугольников применяют вызов команды glBlendFunc(GL\_SRC\_ALPHA\_SATURATE, GL\_ONE).

**Выполнение работы**

Работа выполнена на операционной системе Windows 10. Приложение было создано на Python 3.10 с применением библиотеки OpenGL, в среде разработки VSCode. Для создания интерфейса использовалась библиотека PyQt6.

Пользовательский интерфейс, был взят из 1 работы и был расширен: добавлены тесты отсечения, прозрачности, смешения цветов.

**Тестирование**

Результаты тестирования представлены на снимках экрана.

*Тест прозрачности*

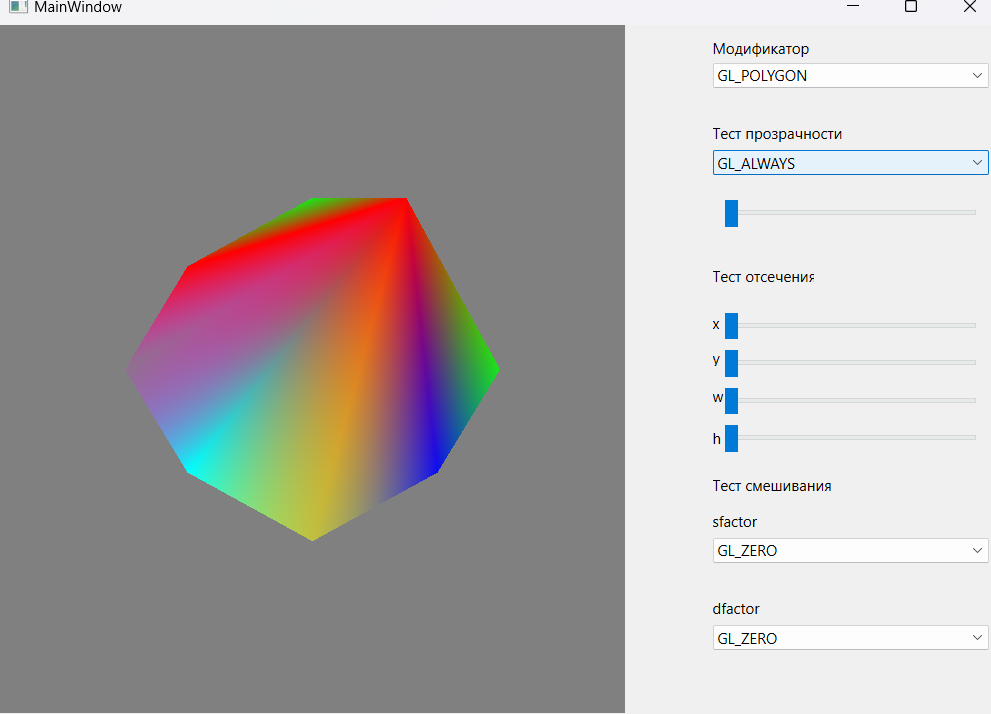


Рисунок 1 — Пропускает цвет всегда

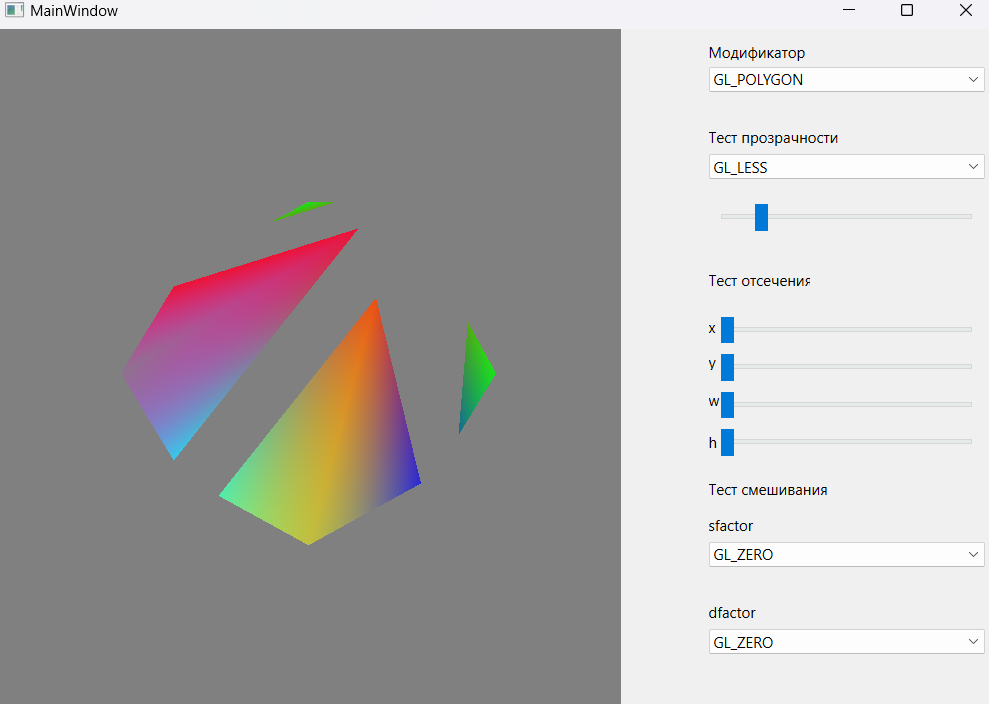


Рисунок 2 — Пропускает, если входное значение альфа меньше, чем значение ref

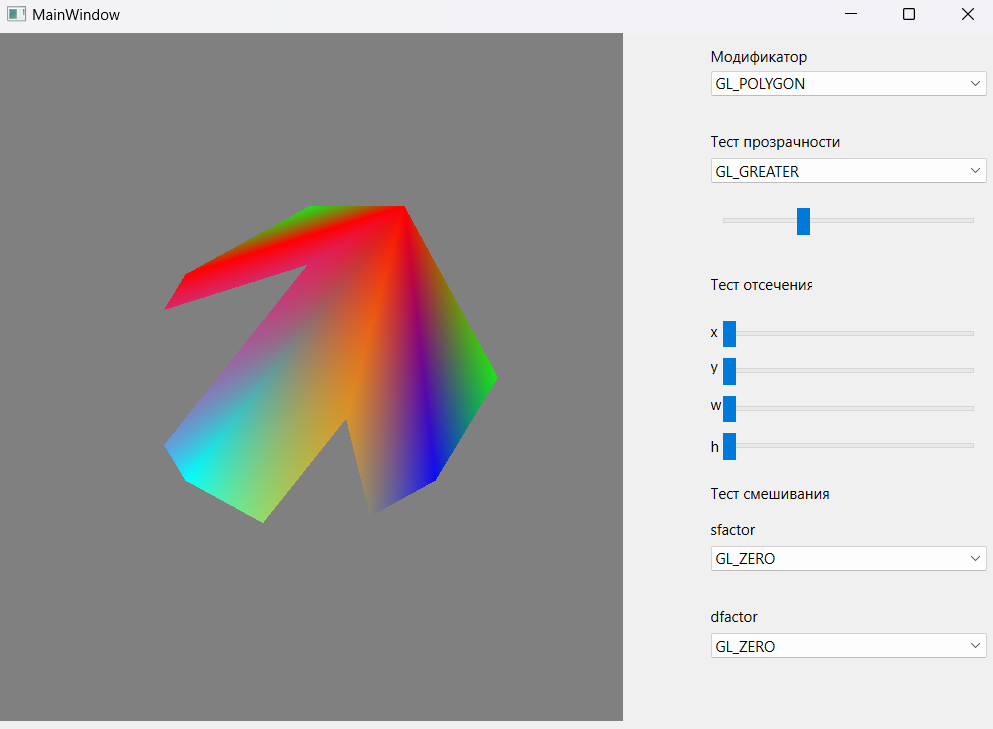


Рисунок 3 — Пропускает, если входное значение альфа больше, чем значение ref

*Тест смешивания цветов*

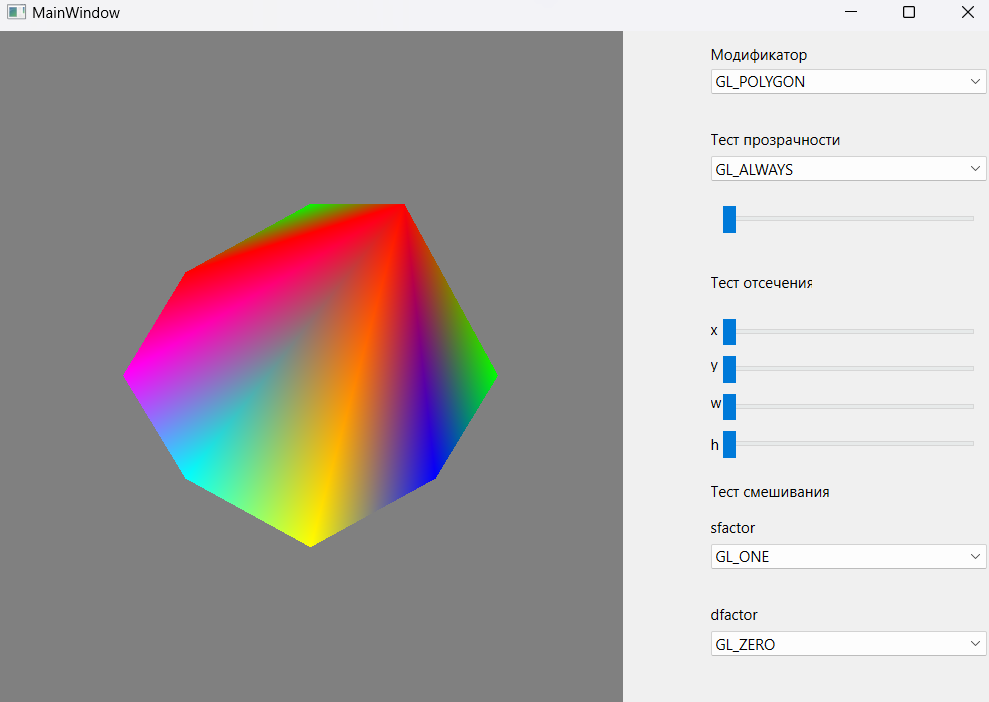


Рисунок 4 — Цвет на переднем фоне отображается, на заднем фоне нет

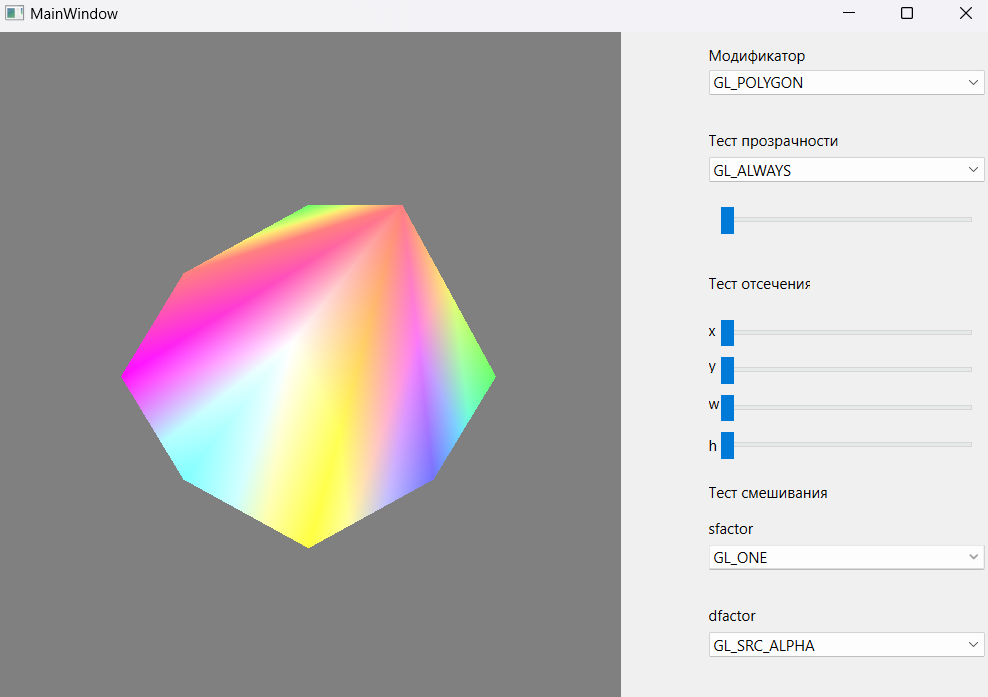


Рисунок 5 — Цвет на переднем фоне отображается, на заднем фоне отображен согласно своей прозрачности

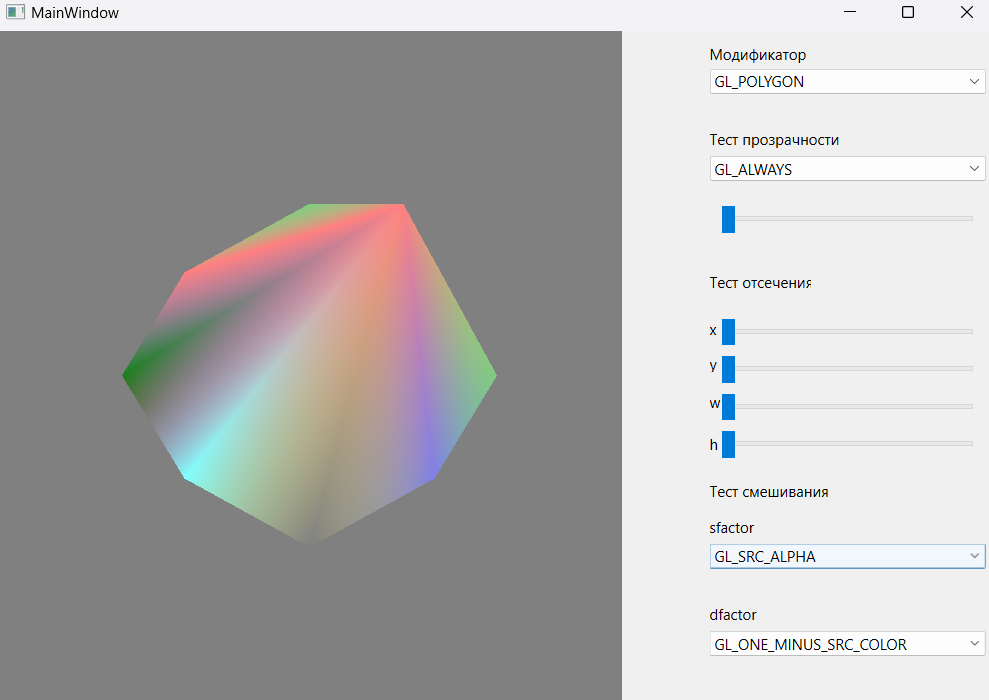


Рисунок 6 — Цвет на переднем плане отображен согласно своей прозрачности, цвет на заднем фоне отображается как цвет, обратный цвету переднего фона

*Тест отсечения*

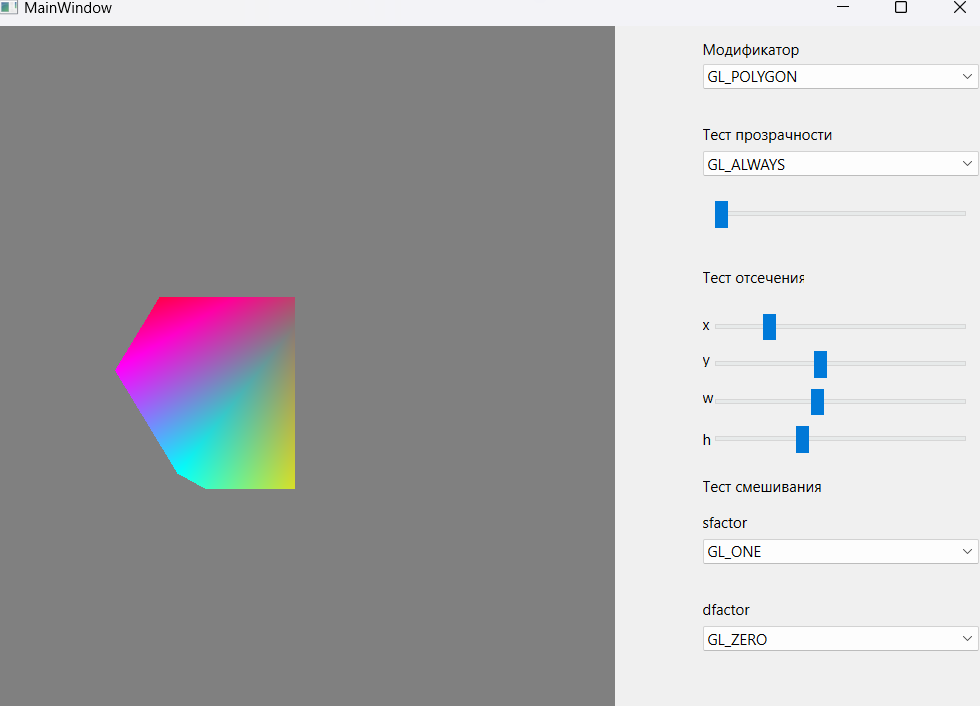


Рисунок 7 — Тест отсечения

**Вывод**

В результате выполнения лабораторной работы была разработана программа, реализующая представление тестов смешивания цветов, отсечения и прозрачности для графических примитивов OpenGL, разработанных в лабораторной работе № 1. Программа работает корректно. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

main.py

from PyQt6 import QtCore, QtWidgets   
import sys   
from OpenGL.GL import \*   
from OpenGL.GLU import \*   
from OpenGL.GLUT import \*   
from PyQt6.QtWidgets import QApplication, QMainWindow, QVBoxLayout, QComboBox, QWidget   
from PyQt6.QtOpenGLWidgets import QOpenGLWidget   
from commands import commands, func, sfactor, dfactor   
   
from PyQt6 import QtCore, QtGui, QtWidgets   
   
   
class MyGLWidget(QOpenGLWidget):   
 def \_\_init\_\_(self, parent):   
 super(MyGLWidget, self).\_\_init\_\_(parent)   
 self.x\_cut = 0   
 self.y\_cut = 0   
 self.width = 99   
 self.height = 99   
 self.ref = 0   
 self.mode = "GL\_POINTS"   
 self.func = "GL\_ALWAYS"   
 self.sfactor = "GL\_SRC\_ALPHA"   
 self.dfactor = "GL\_ONE\_MINUS\_SRC\_ALPHA"   
   
 def paintGL(self):   
 glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)   
 glClearColor(0.5, 0.5, 0.5, 1.0)   
 glEnable(GL\_ALPHA\_TEST)   
 glEnable(GL\_SCISSOR\_TEST)   
 glEnable(GL\_BLEND)   
 glBlendFunc(sfactor[self.sfactor], dfactor[self.dfactor])   
 commands[self.mode]()   
 glAlphaFunc(func[self.func], self.ref)   
 glScissor(   
 int(((self.x\_cut) / 100) \* 500),   
 int(((self.y\_cut) / 100) \* 550),   
 int(((self.width + 1) / 100) \* 500),   
 int(((self.height + 1) / 100) \* 550),   
 )   
 glDisable(GL\_SCISSOR\_TEST)   
 glDisable(GL\_ALPHA\_TEST)   
 glDisable(GL\_BLEND)   
 self.update()   
   
   
class Ui\_MainWindow(object):   
 def setupUi(self, MainWindow):   
 MainWindow.setObjectName("MainWindow")   
 MainWindow.resize(800, 585)   
 self.centralwidget = QtWidgets.QWidget(parent=MainWindow)   
 self.centralwidget.setObjectName("centralwidget")   
 self.glWidget = MyGLWidget(parent=self.centralwidget)   
 self.glWidget.setGeometry(QtCore.QRect(0, 0, 500, 550))   
 self.mode = QtWidgets.QComboBox(parent=self.centralwidget)   
 self.mode.setGeometry(QtCore.QRect(570, 30, 221, 20))   
 self.mode.setObjectName("mode")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.addItem("")   
 self.mode.currentIndexChanged.connect(self.modeChanged)   
 self.label = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label.setGeometry(QtCore.QRect(570, 10, 81, 16))   
 self.label.setObjectName("label")   
 self.func = QtWidgets.QComboBox(parent=self.centralwidget)   
 self.func.setGeometry(QtCore.QRect(570, 100, 221, 20))   
 self.func.setObjectName("func")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.addItem("")   
 self.func.currentIndexChanged.connect(self.funcChanged)   
 self.sfactor = QtWidgets.QComboBox(parent=self.centralwidget)   
 self.sfactor.setGeometry(QtCore.QRect(570, 410, 221, 20))   
 self.sfactor.setObjectName("sfactor")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.addItem("")   
 self.sfactor.currentIndexChanged.connect(self.sfactorChanged)   
 self.dfactor = QtWidgets.QComboBox(parent=self.centralwidget)   
 self.dfactor.setGeometry(QtCore.QRect(570, 480, 221, 20))   
 self.dfactor.setObjectName("dfactor")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.addItem("")   
 self.dfactor.currentIndexChanged.connect(self.dfactorChanged)   
 self.x\_cut = QtWidgets.QSlider(parent=self.centralwidget)   
 self.x\_cut.setGeometry(QtCore.QRect(580, 230, 201, 22))   
 self.x\_cut.setOrientation(QtCore.Qt.Orientation.Horizontal)   
 self.x\_cut.setObjectName("x\_cut")   
 self.x\_cut.valueChanged.connect(self.xChanged)   
 self.width = QtWidgets.QSlider(parent=self.centralwidget)   
 self.width.setGeometry(QtCore.QRect(580, 290, 201, 22))   
 self.width.setOrientation(QtCore.Qt.Orientation.Horizontal)   
 self.width.setObjectName("width")   
 self.width.valueChanged.connect(self.wChanged)   
 self.label\_2 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_2.setGeometry(QtCore.QRect(570, 76, 131, 20))   
 self.label\_2.setObjectName("label\_2")   
 self.label\_3 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_3.setGeometry(QtCore.QRect(570, 190, 81, 21))   
 self.label\_3.setObjectName("label\_3")   
 self.label\_4 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_4.setGeometry(QtCore.QRect(570, 360, 101, 16))   
 self.label\_4.setObjectName("label\_4")   
 self.label\_5 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_5.setGeometry(QtCore.QRect(570, 230, 16, 16))   
 self.label\_5.setObjectName("label\_5")   
 self.label\_6 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_6.setGeometry(QtCore.QRect(570, 320, 41, 20))   
 self.label\_6.setObjectName("label\_6")   
 self.label\_7 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_7.setGeometry(QtCore.QRect(570, 390, 47, 13))   
 self.label\_7.setObjectName("label\_7")   
 self.label\_8 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_8.setGeometry(QtCore.QRect(570, 460, 47, 13))   
 self.label\_8.setObjectName("label\_8")   
 self.y\_cut = QtWidgets.QSlider(parent=self.centralwidget)   
 self.y\_cut.setGeometry(QtCore.QRect(580, 260, 201, 22))   
 self.y\_cut.setOrientation(QtCore.Qt.Orientation.Horizontal)   
 self.y\_cut.setObjectName("y\_cut")   
 self.y\_cut.valueChanged.connect(self.yChanged)   
 self.label\_9 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_9.setGeometry(QtCore.QRect(570, 290, 47, 13))   
 self.label\_9.setObjectName("label\_9")   
 self.label\_10 = QtWidgets.QLabel(parent=self.centralwidget)   
 self.label\_10.setGeometry(QtCore.QRect(570, 260, 47, 13))   
 self.label\_10.setObjectName("label\_10")   
 self.height = QtWidgets.QSlider(parent=self.centralwidget)   
 self.height.setGeometry(QtCore.QRect(580, 320, 201, 22))   
 self.height.setOrientation(QtCore.Qt.Orientation.Horizontal)   
 self.height.setObjectName("height")   
 self.height.valueChanged.connect(self.hChanged)   
 self.ref = QtWidgets.QSlider(parent=self.centralwidget)   
 self.ref.setGeometry(QtCore.QRect(580, 140, 201, 22))   
 self.ref.setOrientation(QtCore.Qt.Orientation.Horizontal)   
 self.ref.setObjectName("ref")   
 self.ref.valueChanged.connect(self.refChanged)   
 MainWindow.setCentralWidget(self.centralwidget)   
 self.menubar = QtWidgets.QMenuBar(parent=MainWindow)   
 self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 26))   
 self.menubar.setObjectName("menubar")   
 MainWindow.setMenuBar(self.menubar)   
 self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)   
 self.statusbar.setObjectName("statusbar")   
 MainWindow.setStatusBar(self.statusbar)   
   
 self.retranslateUi(MainWindow)   
 QtCore.QMetaObject.connectSlotsByName(MainWindow)   
   
 def retranslateUi(self, MainWindow):   
 \_translate = QtCore.QCoreApplication.translate   
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))   
 self.mode.setCurrentText(\_translate("MainWindow", "GL\_POINTS"))   
 self.mode.setItemText(0, \_translate("MainWindow", "GL\_POINTS"))   
 self.mode.setItemText(1, \_translate("MainWindow", "GL\_LINES"))   
 self.mode.setItemText(2, \_translate("MainWindow", "GL\_LINE\_STRIP"))   
 self.mode.setItemText(3, \_translate("MainWindow", "GL\_LINE\_LOOP"))   
 self.mode.setItemText(4, \_translate("MainWindow", "GL\_TRIANGLES"))   
 self.mode.setItemText(5, \_translate("MainWindow", "GL\_TRIANGLE\_STRIP"))   
 self.mode.setItemText(6, \_translate("MainWindow", "GL\_TRIANGLE\_FAN"))   
 self.mode.setItemText(7, \_translate("MainWindow", "GL\_QUADS"))   
 self.mode.setItemText(8, \_translate("MainWindow", "GL\_QUAD\_STRIP"))   
 self.mode.setItemText(9, \_translate("MainWindow", "GL\_POLYGON"))   
 self.label.setText(\_translate("MainWindow", "Модификатор"))   
 self.func.setCurrentText(\_translate("MainWindow", "GL\_ALWAYS"))   
 self.func.setItemText(0, \_translate("MainWindow", "GL\_ALWAYS"))   
 self.func.setItemText(1, \_translate("MainWindow", "GL\_LESS"))   
 self.func.setItemText(2, \_translate("MainWindow", "GL\_EQUAL"))   
 self.func.setItemText(3, \_translate("MainWindow", "GL\_LEQUAL"))   
 self.func.setItemText(4, \_translate("MainWindow", "GL\_GREATER"))   
 self.func.setItemText(5, \_translate("MainWindow", "GL\_NOTEQUAL"))   
 self.func.setItemText(6, \_translate("MainWindow", "GL\_GEQUAL"))   
 self.func.setItemText(7, \_translate("MainWindow", "GL\_NEVER"))   
 self.sfactor.setCurrentText(\_translate("MainWindow", "GL\_ZERO"))   
 self.sfactor.setItemText(0, \_translate("MainWindow", "GL\_ZERO"))   
 self.sfactor.setItemText(1, \_translate("MainWindow", "GL\_ONE"))   
 self.sfactor.setItemText(2, \_translate("MainWindow", "GL\_DST\_COLOR"))   
 self.sfactor.setItemText(3, \_translate("MainWindow", "GL\_ONE\_MINUS\_DST\_COLOR"))   
 self.sfactor.setItemText(4, \_translate("MainWindow", "GL\_SRC\_ALPHA"))   
 self.sfactor.setItemText(5, \_translate("MainWindow", "GL\_ONE\_MINUS\_SRC\_ALPHA"))   
 self.sfactor.setItemText(6, \_translate("MainWindow", "GL\_DST\_ALPHA"))   
 self.sfactor.setItemText(7, \_translate("MainWindow", "GL\_ONE\_MINUS\_DST\_ALPHA"))   
 self.sfactor.setItemText(8, \_translate("MainWindow", "GL\_SRC\_ALPHA\_SATURATE"))   
 self.dfactor.setCurrentText(\_translate("MainWindow", "GL\_ZERO"))   
 self.dfactor.setItemText(0, \_translate("MainWindow", "GL\_ZERO"))   
 self.dfactor.setItemText(1, \_translate("MainWindow", "GL\_ONE"))   
 self.dfactor.setItemText(2, \_translate("MainWindow", "GL\_SRC\_COLOR"))   
 self.dfactor.setItemText(3, \_translate("MainWindow", "GL\_ONE\_MINUS\_SRC\_COLOR"))   
 self.dfactor.setItemText(4, \_translate("MainWindow", "GL\_SRC\_ALPHA"))   
 self.dfactor.setItemText(5, \_translate("MainWindow", "GL\_ONE\_MINUS\_SRC\_ALPHA"))   
 self.dfactor.setItemText(6, \_translate("MainWindow", "GL\_DST\_ALPHA"))   
 self.dfactor.setItemText(7, \_translate("MainWindow", "GL\_ONE\_MINUS\_DST\_ALPHA"))   
 self.label\_2.setText(\_translate("MainWindow", "Тест прозрачности"))   
 self.label\_3.setText(\_translate("MainWindow", "Тест отсечения"))   
 self.label\_4.setText(\_translate("MainWindow", "Тест смешивания"))   
 self.label\_5.setText(\_translate("MainWindow", "x"))   
 self.label\_6.setText(\_translate("MainWindow", "h"))   
 self.label\_7.setText(\_translate("MainWindow", "sfactor"))   
 self.label\_8.setText(\_translate("MainWindow", "dfactor"))   
 self.label\_9.setText(\_translate("MainWindow", "w"))   
 self.label\_10.setText(\_translate("MainWindow", "y"))   
   
 def modeChanged(self):   
 print(self.mode.currentText())   
 self.glWidget.mode = self.mode.currentText()   
 self.glWidget.update()   
   
 def xChanged(self):   
 print(self.x\_cut.value())   
 self.glWidget.x\_cut = self.x\_cut.value()   
 self.glWidget.update()   
   
 def yChanged(self):   
 print(self.y\_cut.value())   
 self.glWidget.y\_cut = self.y\_cut.value()   
 self.glWidget.update()   
   
 def wChanged(self):   
 print(self.width.value())   
 self.glWidget.width = self.width.value()   
 self.glWidget.update()   
   
 def hChanged(self):   
 print(self.height.value())   
 self.glWidget.height = self.height.value()   
 self.glWidget.update()   
   
 def refChanged(self):   
 print(1 - self.ref.value() / 100)   
 self.glWidget.ref = 1 - self.ref.value() / 100   
 self.glWidget.update()   
   
 def funcChanged(self):   
 print(self.func.currentText())   
 self.glWidget.func = self.func.currentText()   
 self.glWidget.update()   
   
 def sfactorChanged(self):   
 print(self.sfactor.currentText())   
 self.glWidget.sfactor = self.sfactor.currentText()   
 self.glWidget.update()   
   
 def dfactorChanged(self):   
 print(self.dfactor.currentText())   
 self.glWidget.dfactor = self.dfactor.currentText()   
 self.glWidget.update()   
   
   
if \_\_name\_\_ == "\_\_main\_\_":   
 app = QtWidgets.QApplication(sys.argv)   
 MainWindow = QtWidgets.QMainWindow()   
 ui = Ui\_MainWindow()   
 ui.setupUi(MainWindow)   
 MainWindow.show()   
 sys.exit(app.exec())

commands.py

from OpenGL.GL import \*

from OpenGL.GLUT import \*

colors = [[1.0, 0.0, 0.0, 1.0],

[0.0, 1.0, 0.0, 0.8],

[0.0, 0.0, 1.0, 0.9],

[1.0, 1.0, 0.0, 0.5],

[0.0, 1.0, 1.0, 1.0],

[1.0, 0.0, 1.0, 0.1]]

vertex = [[0.3, 0.5],

[0.6, 0],

[0.4, -0.3],

[0, -0.5],

[-0.4, -0.3],

[-0.6, 0],

[-0.4, 0.3],

[0, 0.5]]

def points():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glPointSize(3)

glBegin(GL\_POINTS)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def lines():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(3)

glBegin(GL\_LINES)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def lineStrip():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(3)

glBegin(GL\_LINE\_STRIP)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def lineLoop():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(2)

glBegin(GL\_LINE\_LOOP)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def triangles():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(4)

glBegin(GL\_TRIANGLES)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def triangleStrip():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(4)

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)

glBegin(GL\_TRIANGLE\_STRIP)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def triangleFan():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(4)

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)

glBegin(GL\_TRIANGLE\_FAN)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def quads():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(4)

glBegin(GL\_QUADS)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def quadStrip():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(4)

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)

glBegin(GL\_QUAD\_STRIP)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

def polygon():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLineWidth(4)

glBegin(GL\_POLYGON)

for i in range(len(vertex)):

glColor4f(\*colors[i%len(colors)])

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

glEnd()

commands = {

"GL\_POINTS" : points,

"GL\_LINES" : lines,

"GL\_LINE\_STRIP" : lineStrip,

"GL\_LINE\_LOOP" : lineLoop,

"GL\_TRIANGLES": triangles,

"GL\_TRIANGLE\_STRIP": triangleStrip,

"GL\_TRIANGLE\_FAN" : triangleFan,

"GL\_QUADS" : quads,

"GL\_QUAD\_STRIP" : quadStrip,

"GL\_POLYGON" : polygon

}

func = {

'GL\_ALWAYS': GL\_ALWAYS,

'GL\_NEVER': GL\_NEVER,

'GL\_LESS': GL\_LESS,

'GL\_EQUAL': GL\_EQUAL,

'GL\_LEQUAL': GL\_LEQUAL,

'GL\_GREATER': GL\_GREATER,

'GL\_NOTEQUAL': GL\_NOTEQUAL,

'GL\_GEQUAL': GL\_GEQUAL,

'GL\_NEVER': GL\_NEVER

}

sfactor = {

'GL\_ONE': GL\_ONE,

'GL\_ZERO': GL\_ZERO,

'GL\_DST\_COLOR': GL\_DST\_COLOR,

'GL\_ONE\_MINUS\_DST\_COLOR': GL\_ONE\_MINUS\_DST\_COLOR,

'GL\_SRC\_ALPHA': GL\_SRC\_ALPHA,

'GL\_ONE\_MINUS\_SRC\_ALPHA': GL\_ONE\_MINUS\_SRC\_ALPHA,

'GL\_DST\_ALPHA': GL\_DST\_ALPHA,

'GL\_ONE\_MINUS\_DST\_ALPHA': GL\_ONE\_MINUS\_DST\_ALPHA,

'GL\_SRC\_ALPHA\_SATURATE': GL\_SRC\_ALPHA\_SATURATE

}

dfactor = {

'GL\_ZERO': GL\_ZERO,

'GL\_ONE': GL\_ONE,

'GL\_SRC\_COLOR': GL\_SRC\_COLOR,

'GL\_ONE\_MINUS\_SRC\_COLOR': GL\_ONE\_MINUS\_SRC\_COLOR,

'GL\_SRC\_ALPHA': GL\_SRC\_ALPHA,

'GL\_ONE\_MINUS\_SRC\_ALPHA': GL\_ONE\_MINUS\_SRC\_ALPHA,

'GL\_DST\_ALPHA': GL\_DST\_ALPHA,

'GL\_ONE\_MINUS\_DST\_ALPHA': GL\_ONE\_MINUS\_DST\_ALPHA

}