**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**По лабораторной работе № 2**

**по дисциплине «Компьютерная графика»**

Тема: **Примитивы OpenGL**

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

Санкт-Петербург

2023

**Цель работы.**

Ознакомиться с тестами отсечения, прозрачности и смешения цветов и реализовать их в своей программе.

**Задания.**

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 остановки интерактивно различных атрибутов тестов 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.

Макет пользовательского интерфейса был разработан в программе Qt Designer (см. рис. 1).

Изображение выглядит как текст, снимок экрана, электроника, монитор

Автоматически созданное описание

Рисунок 1 – разработанный в Qt Designer макет пользовательского интерфейса

Класс главного окна MainWindow наследуется от базового для PyQt класса QMainWindow и содержит внутри себя виджет glWidget унаследованный от QOpenGLWidget, для отображения графики, а также меню для настройки параметров изображения (выбор отображаемого примитива, активация тестов и настройка их параметров). После любого изменения в меню у виджета glWidget вызывается метод update(), который выполняет перерисовку изображения.

Поля и методы класса MainWindow:

* ui – поле, необходимое для работы с элементами пользовательского интерфейса, созданными в Qt Designer.
* primitives – список примитивов для сопоставления необходимой константы индексу выбранного элемента пользователем в списке.
* transparency – список возможных параметров функции glAlphaFunc,для сопоставления необходимой константы индексу выбранного элемента пользователем в списке.
* sfactor и dfactor – списки возможных параметров функции glBlendFunc для сопоставления необходимой константы индексу выбранного элемента пользователем в списке.
* quit() – осуществляет выход из приложения по нажатию на кнопку.
* draw\_primitive() – рисует выбранный пользователем примитив.
* cutting() – активирует\деактивирует тест отсечения.
* transparent() – активирует\деактивирует тест прозрачности.
* blender() – активирует\деактивирует тест смешения цветов.

Поля и методы класса glWidget:

* primitive – поле для хранения отображаемого примитива.
* makeNewPoints – флаг, определяющий необходимость генерации новых точек и цветов для них.
* points – список точек для построения примитива.
* colors – список цветов для каждой точки.
* mw – ссылка на MainWindow.
* scissors – список параметров для теста отсечения. В общем виде может быть записан как [x, y, width, height], где x – абсцисса левого нижнего угла прямоугольника, y – ордината левого нижнего угла прямоугольника, width – ширина прямоугольника, height – высота прямоугольника.
* scissors\_test – флаг, определяющий активирован ли тест отсечения.
* rubberBand – ссылка на выделяемую курсором область, которая будет передана в тест отсечения.
* initializeGL() – метод, осуществляющий подготовку кадра.
* paintGL() – метод, осуществляющий отрисовку изображения.
* draw\_primitive() – метод, предназначенный для рисования примитивов.
* draw\_circles() – метод, рисующий три круга: зеленый, красный и синий.
* mousePressEvent(), mouseMoveEvent(), mouseReleaseEvent() – обработчики событий мыши. С их помощью можно определить параметры выделяемой пользователем области для теста отсечения с помощью мышки.

Метод paintGL() у виджета glWidget осуществляет перерисовку изображения. Сначала происходит проверка на задание пользователем примитива. Если примитив не задан, то программа ничего не нарисует. Далее проверяется флаг scissors\_test и последние два значения в списке scissors. Если флаг установлен в 0 или последние значения в списке scissors равны 0(это значит, что пользователь задал прямоугольник с нулевой длиной и шириной), то программа не активирует тест отсечения. В противном случае с помощью команды glEnable выставляется соответствующий режим GL\_SCISSOR\_TEST и происходит вызов функции glScissor с параметрами из поля scissors.

Следующим проверяется статус теста прозрачности, если он неактивен, то программа не выставляет необходимый для этого теста режим, иначе устанавливается режим GL\_ALPHA\_TEST и вызывается функция glAlphaFunc.

Последним проверяется статус теста смешения цветов. Аналогично тесту прозрачности, если он не активирован пользователем, то программа не устанавливает соответствующий режим, если же пользователь активировал этот тест, то выставляется режим GL\_BLEND и вызывается функция glBlendFunc.

Проверка статусов последних двух тестов осуществляется с помощью получения значения соответствующих чекбоксов из основного окна MainWindow.

Далее вызывается функция draw\_primitive. В начале своей работы она проверяет флаг makeNewPoints и в зависимости от его значения генерирует новые цвета и точки для отображения или же оставляет старые. Если пользователь выбрал круги, то вызывается функция draw\_circles, отвечающая за их рисование.

Для того чтобы нарисовать круг, был использован примитив GL\_TRIANGLE\_FAN – это треугольники, которые имеют одну общую точку. С помощью тригонометрических функций синуса и косинуса высчитывается значение для каждой последующей точки треугольника и проводится его отрисовка.

Разработанный программный код см. в приложении А.

**Тестирование.**

Тестирование проводилось методом проверки правильного выполнения

разработанного кода.

Пример результатов тестирования представлены на снимках экрана (см. рис. 2 – 10 ).

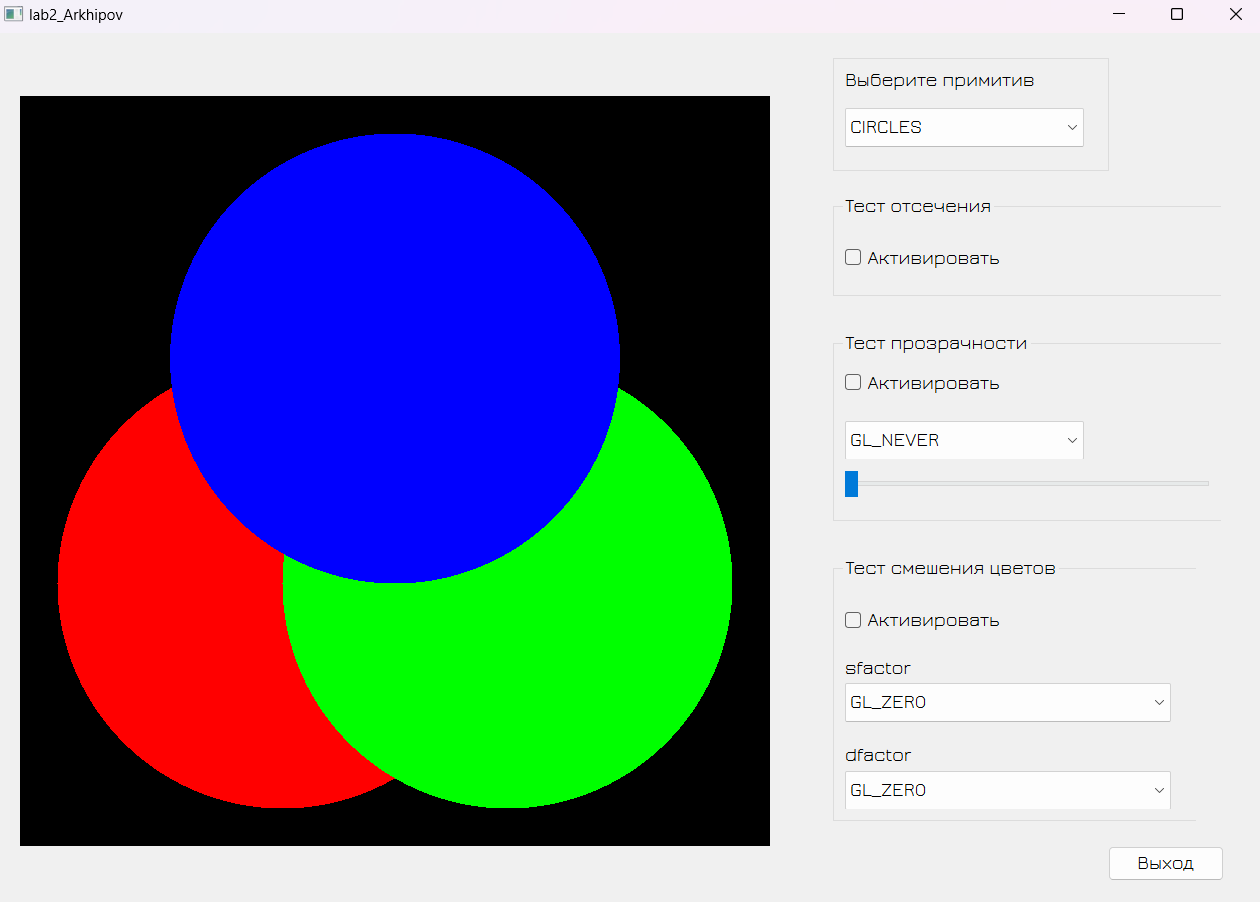


Рисунок 2 – круги без активных тестов.

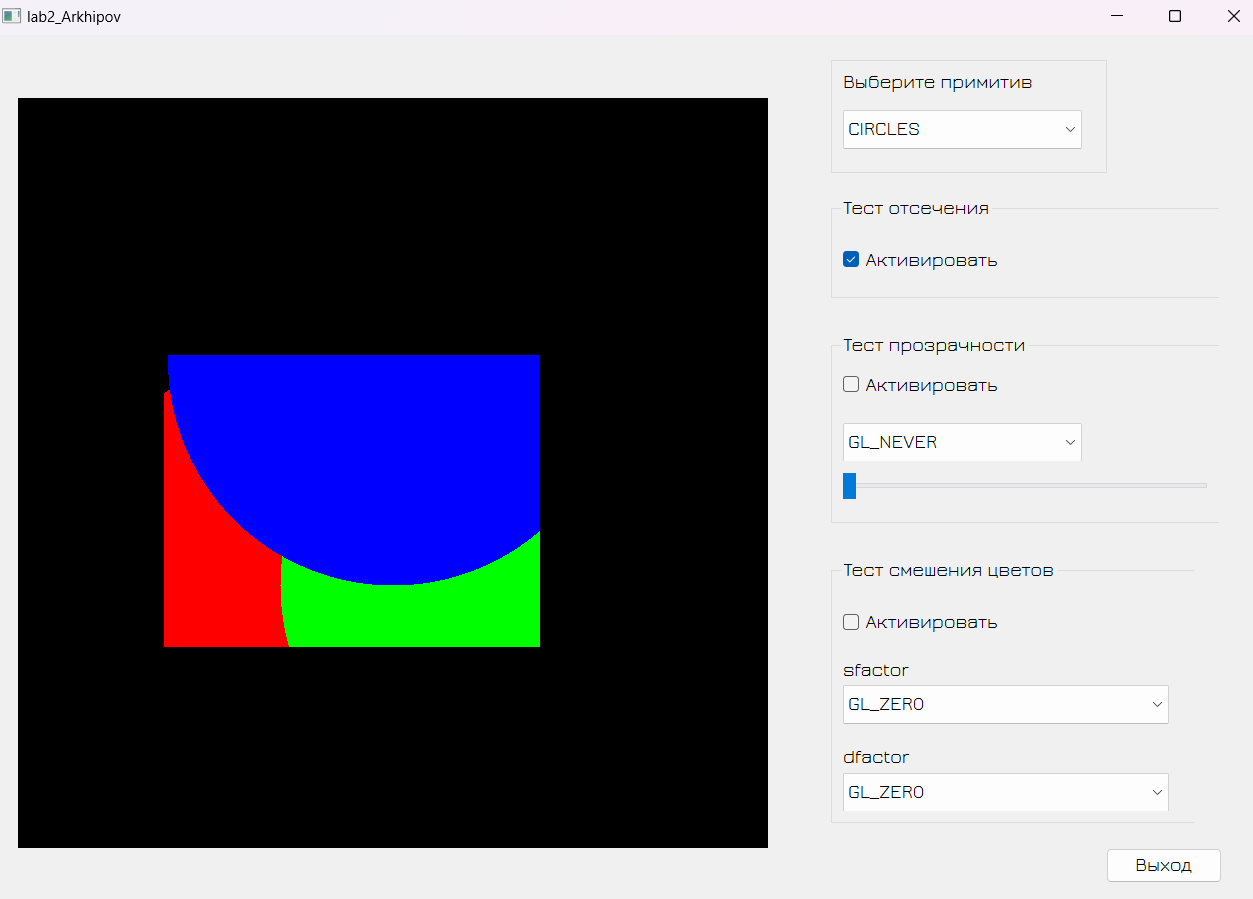


Рисунок 3 – тест отсечения

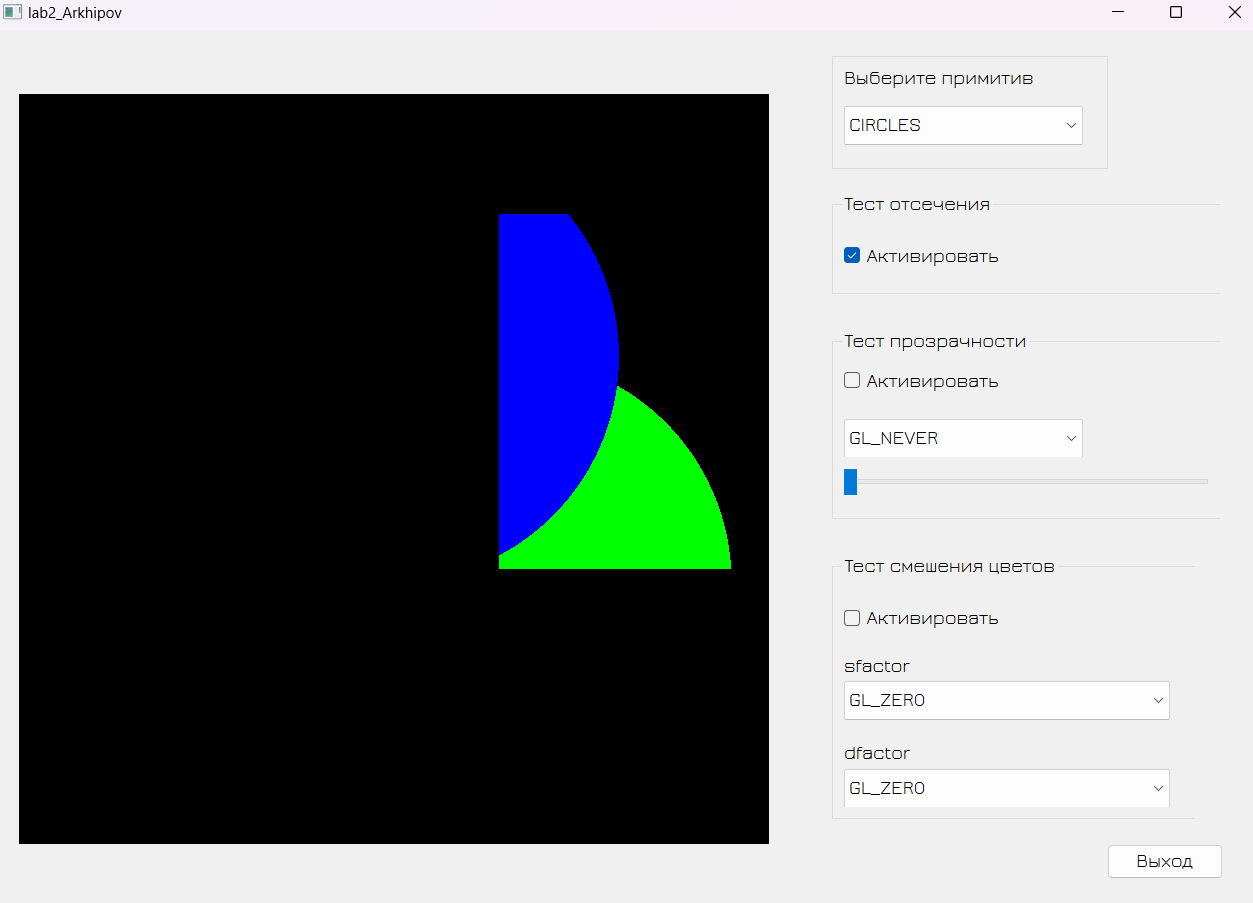


Рисунок 4 – тест отсечения

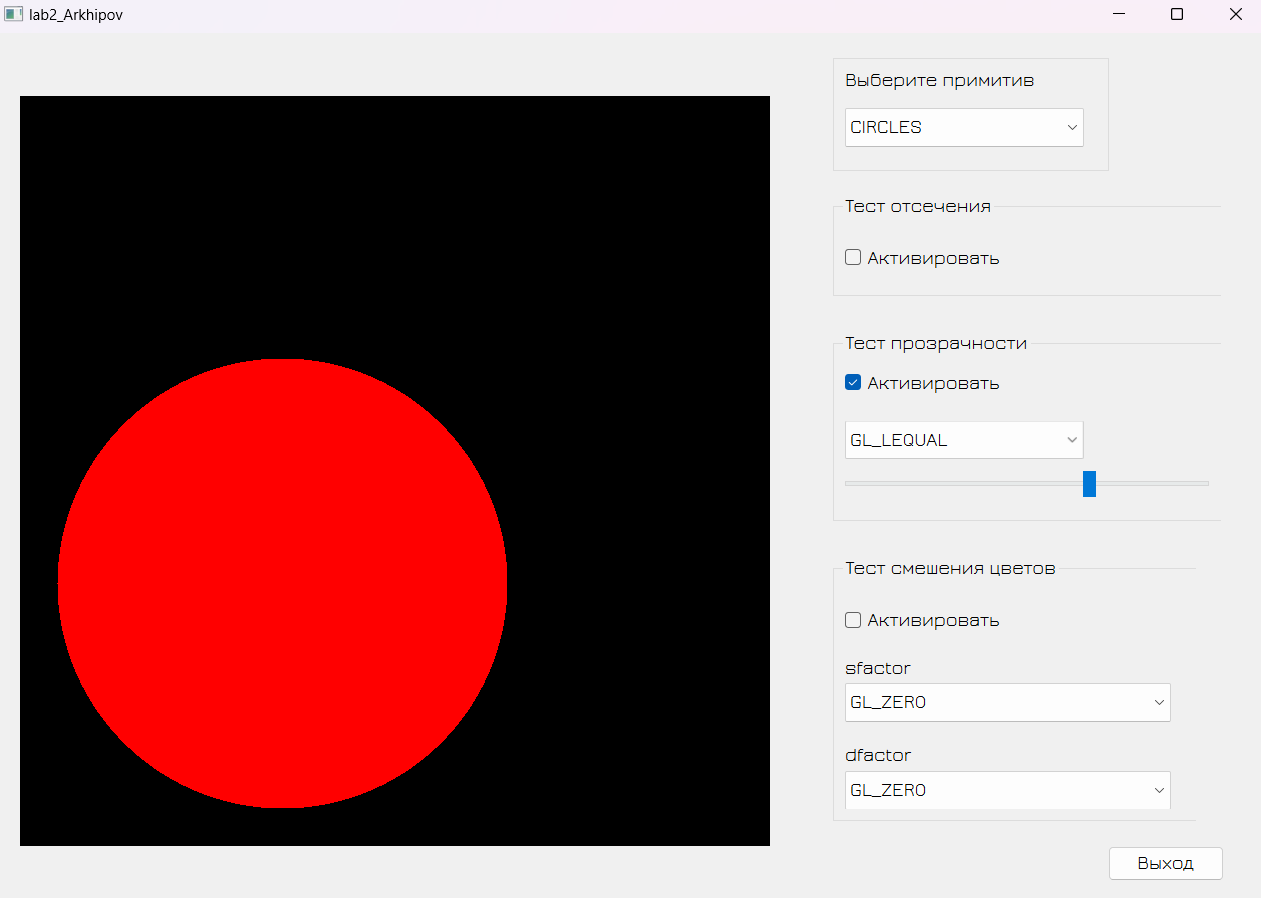


Рисунок 5 – тест прозрачности

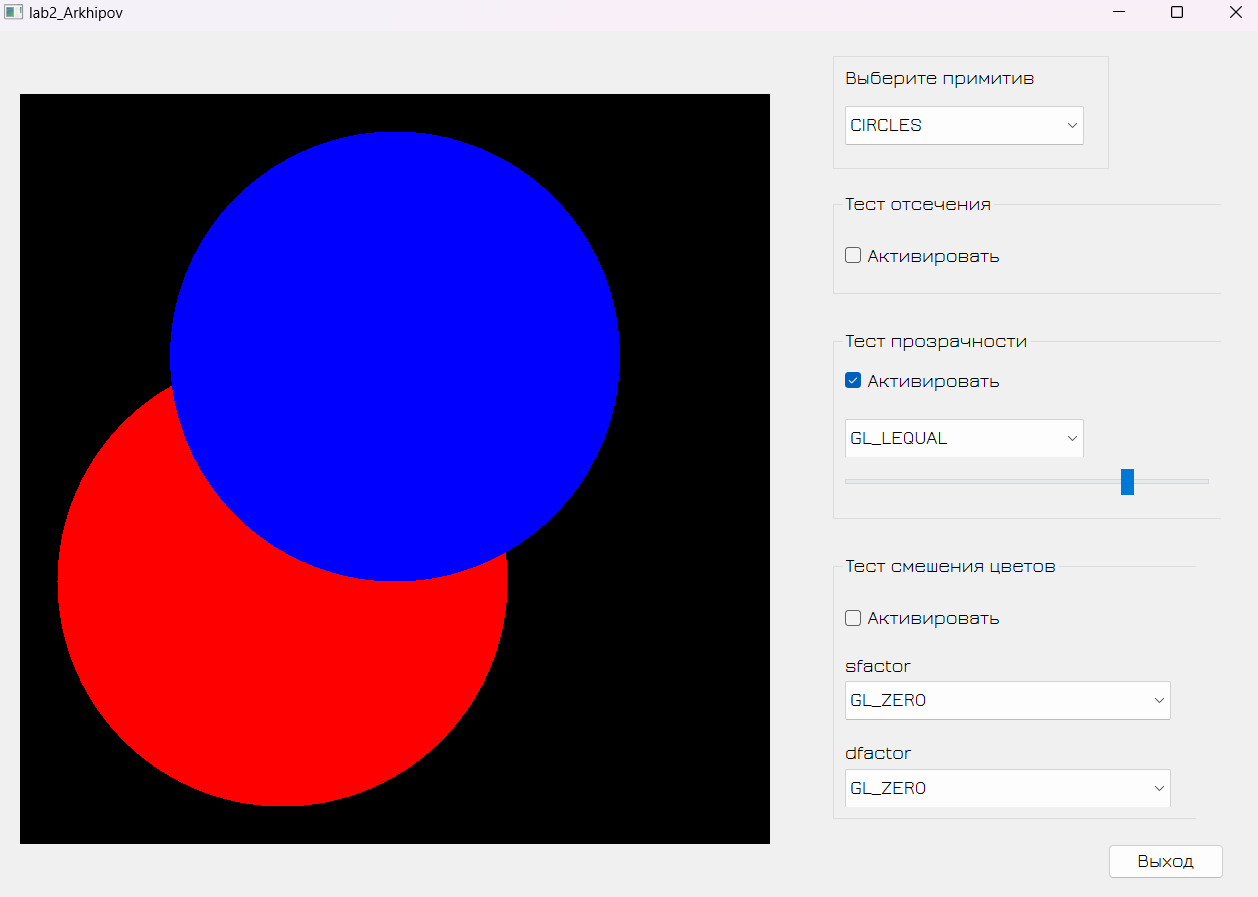


Рисунок 6 – тест прозрачности

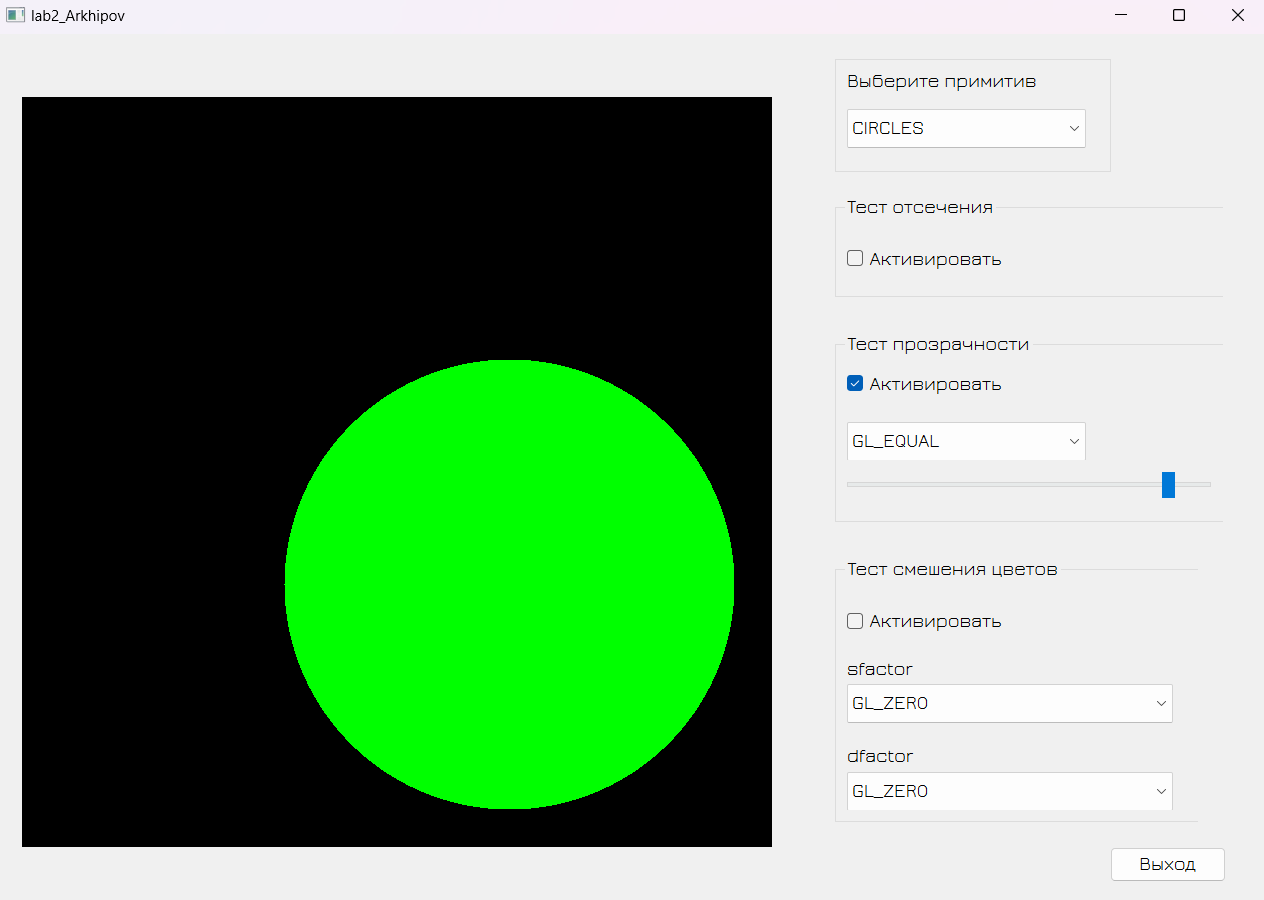


Рисунок 7 – тест прозрачности

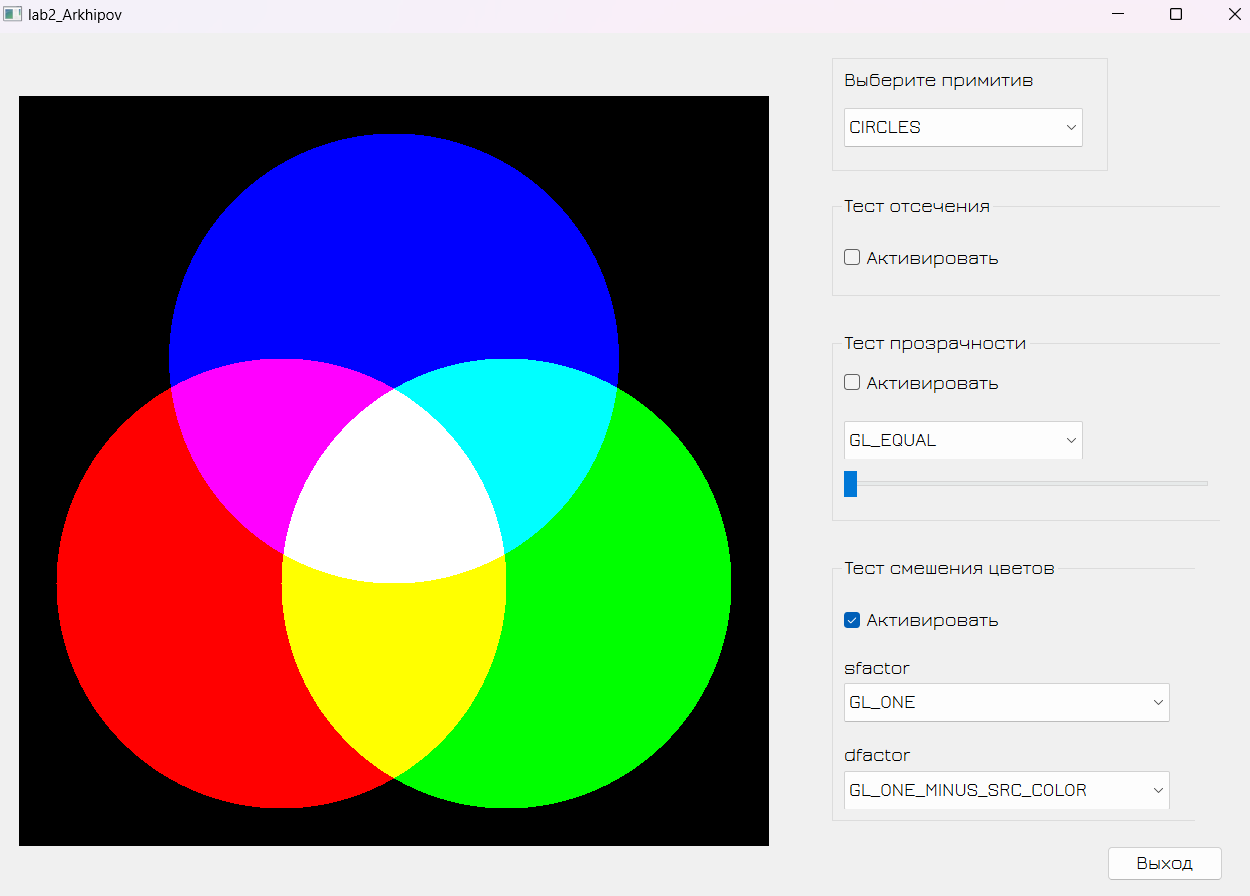


Рисунок 8 – тест смешения цветов

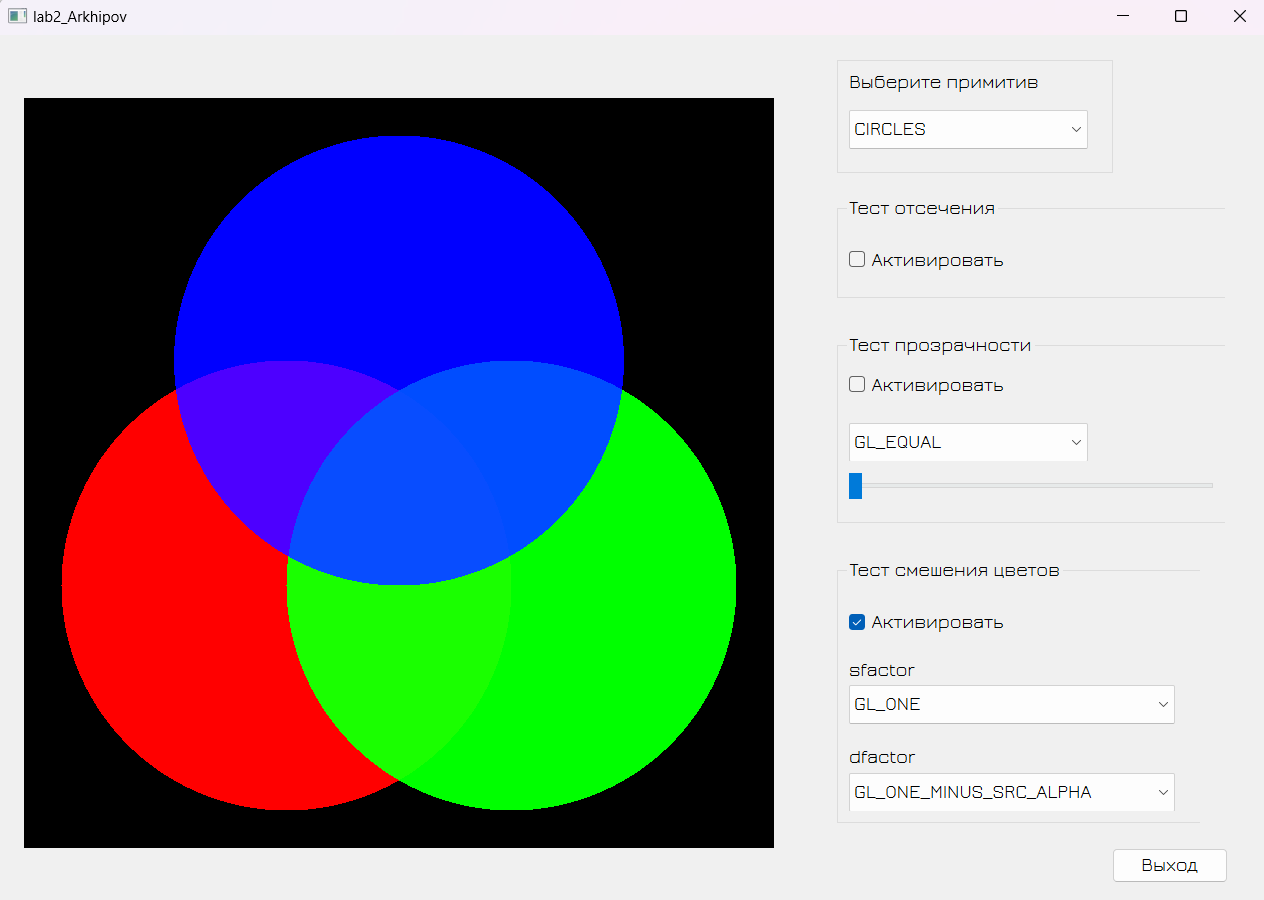


Рисунок 9 – тест смешения цветов

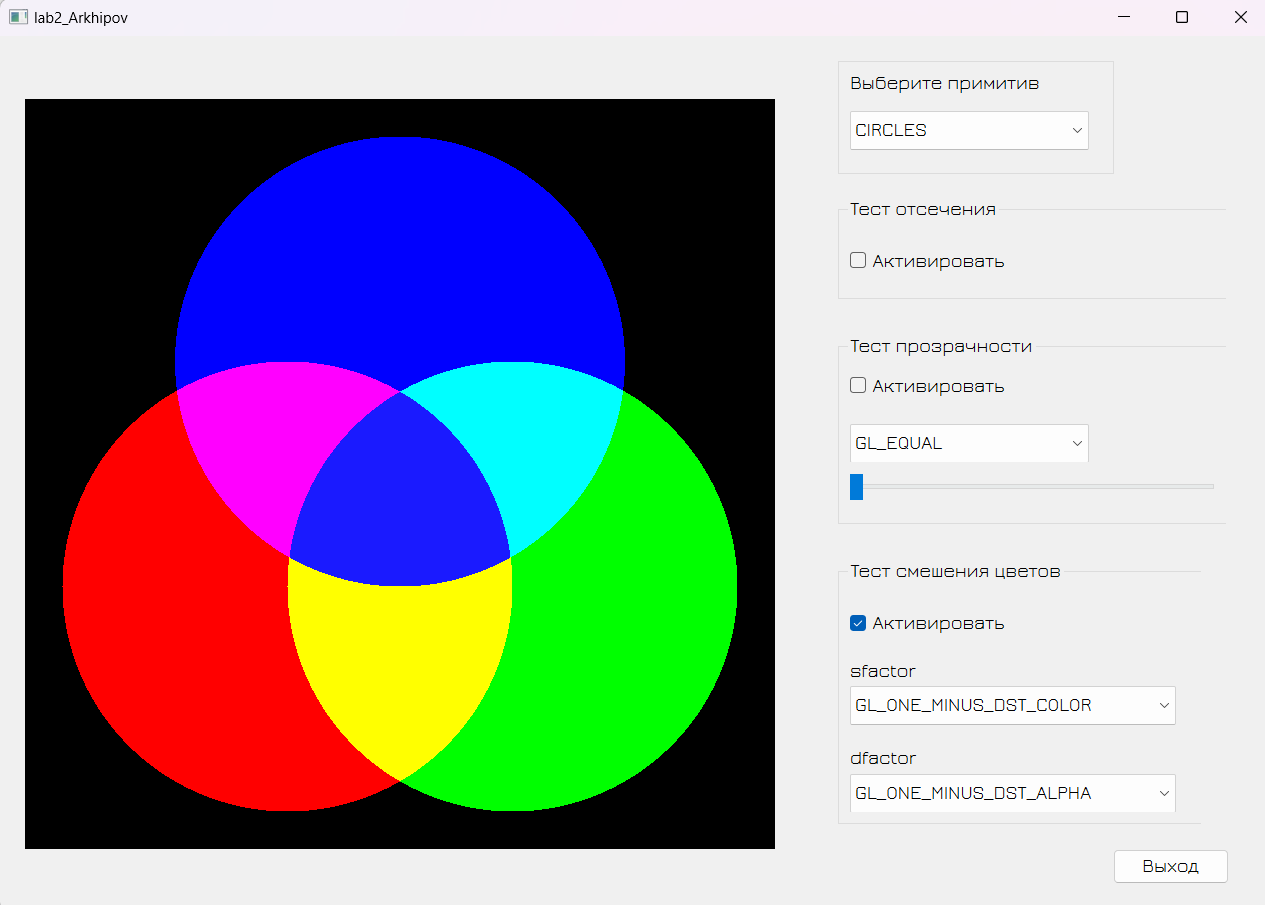


Рисунок 10 – тест смешения цветов

**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**

**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  
import math  
from PyQt6.QtWidgets import QRubberBand  
from PyQt6.QtCore import QSize, QRect  
  
  
class glWidget(QOpenGLWidget):  
 def \_\_init\_\_(self, main\_window):  
 super().\_\_init\_\_(parent=main\_window.ui.centralwidget)  
 *# Примитив, который будет нарисован* self.primitive = None  
 *# Флаг, содержащий информацию о том, нужно ли вычислять точки заново* self.makeNewPoints = False  
 *# Ссылка на родительское окно* self.mw = main\_window  
 *# Параметры для ножниц* self.scissors = [0, 0, 0, 0]  
 *# Флаг для ножниц* self.scissors\_test = False  
 *# Область выбора* self.rubberBand = None  
  
 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:  
 *# Тест отсечения* if self.scissors\_test and self.scissors[2:4] != [0, 0]:  
 gl.glEnable(gl.GL\_SCISSOR\_TEST)  
 gl.glScissor(int(self.scissors[0] \* 750 / 600), int(self.scissors[1] \* 750 / 600),  
 int(self.scissors[2] \* 750 / 600), int(self.scissors[3] \* 750 / 600))  
 else:  
 gl.glDisable(gl.GL\_SCISSOR\_TEST)  
 *# gl.glScissor(0, 0, 750, 750)  
  
 # Тест прозрачности* if self.mw.ui.checkBox\_2.isChecked():  
 gl.glEnable(gl.GL\_ALPHA\_TEST)  
 gl.glAlphaFunc(self.mw.transparency[self.mw.ui.comboBox\_2.currentIndex()],  
 self.mw.ui.horizontalSlider\_3.value() / 100)  
 else:  
 gl.glDisable(gl.GL\_ALPHA\_TEST)  
  
 *# Тест смешение цветов* if self.mw.ui.checkBox\_3.isChecked():  
 gl.glEnable(gl.GL\_BLEND)  
 gl.glBlendFunc(self.mw.sfactor[self.mw.ui.comboBox\_4.currentIndex()],  
 self.mw.dfactor[self.mw.ui.comboBox\_3.currentIndex()])  
 else:  
 gl.glDisable(gl.GL\_BLEND)  
  
 self.draw\_primitive()  
  
 *# От диагонали к диагонали  
 # if self.scissors\_test:  
 # self.scissors\_test = False  
 # gl.glDisable(gl.GL\_SCISSOR\_TEST)  
  
 # Функция для рисования примитивов* def draw\_primitive(self):  
 if self.primitive != "CIRCLES":  
 if self.makeNewPoints:  
 *# Генерация новых точек, если это необходимо* self.points = [[uniform(-0.99, 0.99), uniform(-0.99, 0.99)] for \_ in range(12)]  
 *# Генерация цветов точек* self.colors = [[uniform(0, 1) for i in range(4)] for \_ in range(len(self.points))]  
  
 *# Установка размера точки* gl.glPointSize(5)  
 *# Установка толщины линии* gl.glLineWidth(5)  
  
 *# Рисование* gl.glBegin(self.primitive)  
 for index, point in enumerate(self.points):  
 gl.glColor4fv(self.colors[index])  
 gl.glVertex2fv(point)  
 gl.glEnd()  
  
 elif self.primitive == "CIRCLES":  
 self.draw\_circles()  
  
 self.makeNewPoints = False  
  
 def draw\_circles(self):  
 for color, pos\_x\_y in zip([[1, 0, 0, 0.6], [0, 1, 0, 0.9], [0, 0, 1, 0.7]],  
 [[-0.3, -0.3], [0.3, -0.3], [0, 0.3]]):  
 steps = 200  
 radius = 0.6  
 a = math.pi \* 2 / steps  
  
 gl.glBegin(gl.GL\_TRIANGLE\_FAN)  
 gl.glColor4fv(color)  
 gl.glVertex2fv(pos\_x\_y)  
 for i in range(steps + 1):  
 x = pos\_x\_y[0] + math.sin(a \* i) \* radius  
 y = pos\_x\_y[1] + math.cos(a \* i) \* radius  
 gl.glVertex2f(x, y)  
 gl.glEnd()  
  
 *# Окно с выбором для обрезания* def mousePressEvent(self, event):  
 if self.mw.ui.checkBox.isChecked():  
 self.origin = event.pos()  
  
 self.scissors[0] = self.origin.x()  
 self.scissors[1] = self.origin.y()  
  
 if not self.rubberBand:  
 self.rubberBand = QRubberBand(QRubberBand.Shape.Rectangle, self)  
 self.rubberBand.setGeometry(QRect(self.origin, QSize()))  
 self.rubberBand.show()  
  
 def mouseMoveEvent(self, event):  
 if self.mw.ui.checkBox.isChecked():  
 self.rubberBand.setGeometry(QRect(self.origin, event.pos()).normalized())  
  
 def mouseReleaseEvent(self, event):  
 if self.mw.ui.checkBox.isChecked():  
 if event.pos().x() < 0:  
 width = abs(self.scissors[0])  
 elif event.pos().x() > 600:  
 width = abs(600 - self.scissors[0])  
 else:  
 width = abs(self.scissors[0] - event.pos().x())  
  
 if event.pos().y() < 0:  
 height = abs(self.scissors[1])  
 elif event.pos().y() > 600:  
 height = abs(600 - self.scissors[1])  
 else:  
 height = abs(event.pos().y() - self.scissors[1])  
  
 self.scissors[0] = min(self.scissors[0], event.pos().x())  
 self.scissors[1] = 600 - max(self.scissors[1], event.pos().y())  
 self.scissors[2] = width  
 self.scissors[3] = height  
  
 self.rubberBand.hide()  
 if height \* width != 0:  
 self.scissors\_test = True  
 self.update()

Название файла: MainWindow.py

import sys  
from PyQt6.QtWidgets import QMainWindow, QRubberBand  
from glWidget import glWidget  
from ui\_mainwindow import Ui\_MainWindow  
from OpenGL import GL as gl  
from PyQt6.QtCore import QSize, QRect, Qt  
  
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)  
  
 *# Создание списка примитивов и минимального числа точек, необходимого для их построения* self.primitives = [  
 gl.GL\_POINTS,  
 gl.GL\_LINES,  
 gl.GL\_LINE\_STRIP,  
 gl.GL\_LINE\_LOOP,  
 gl.GL\_TRIANGLES,  
 gl.GL\_TRIANGLE\_STRIP,  
 gl.GL\_TRIANGLE\_FAN,  
 gl.GL\_QUADS,  
 gl.GL\_QUAD\_STRIP,  
 gl.GL\_POLYGON,  
 "CIRCLES"  
 ]  
  
 *# Создание списка для теста прозрачности* self.transparency = [  
 gl.GL\_NEVER,  
 gl.GL\_LESS,  
 gl.GL\_EQUAL,  
 gl.GL\_LEQUAL,  
 gl.GL\_GREATER,  
 gl.GL\_NOTEQUAL,  
 gl.GL\_GEQUAL,  
 gl.GL\_ALWAYS  
 ]  
  
 *# Создание списка для sfactor* self.sfactor = [  
 gl.GL\_ZERO,  
 gl.GL\_ONE,  
 gl.GL\_DST\_COLOR,  
 gl.GL\_ONE\_MINUS\_DST\_COLOR,  
 gl.GL\_SRC\_ALPHA,  
 gl.GL\_ONE\_MINUS\_SRC\_ALPHA,  
 gl.GL\_DST\_ALPHA,  
 gl.GL\_ONE\_MINUS\_DST\_ALPHA,  
 gl.GL\_SRC\_ALPHA\_SATURATE  
 ]  
  
 *# Создание списка для dfactor* self.dfactor = [  
 gl.GL\_ZERO,  
 gl.GL\_ONE,  
 gl.GL\_SRC\_COLOR,  
 gl.GL\_ONE\_MINUS\_SRC\_COLOR,  
 gl.GL\_SRC\_ALPHA,  
 gl.GL\_ONE\_MINUS\_SRC\_ALPHA,  
 gl.GL\_DST\_ALPHA,  
 gl.GL\_ONE\_MINUS\_DST\_ALPHA  
 ]  
  
  
 *#Обработка событий* self.ui.pushButton.clicked.connect(self.quit)  
 self.ui.comboBox.activated.connect(self.draw\_primitive)  
  
 *# Тест отсечения* self.ui.checkBox.stateChanged.connect(self.cutting)  
  
 *# Тест прозрачности* self.ui.checkBox\_2.stateChanged.connect(self.transparent)  
 self.ui.comboBox\_2.activated.connect(self.transparent)  
 self.ui.horizontalSlider\_3.valueChanged.connect(self.transparent)  
  
 *# Смешение цветов* self.ui.checkBox\_3.stateChanged.connect(self.blender)  
 self.ui.comboBox\_3.activated.connect(self.blender)  
 self.ui.comboBox\_4.activated.connect(self.blender)  
  
  
 *# Демонстрация окна* self.show()  
  
 def quit(self):  
 sys.exit()  
  
 *# Рисование примитивов* def draw\_primitive(self, index):  
 self.ui.openGLWidget.primitive = self.primitives[index]  
 self.ui.openGLWidget.makeNewPoints = True  
 self.ui.openGLWidget.scissors\_test = False  
 self.ui.openGLWidget.scissors = [0,0,0,0]  
  
 *# Сохранение обрезания  
 # if self.ui.checkBox.isChecked():  
 # self.ui.openGLWidget.scissors\_test = True* self.ui.openGLWidget.update()  
  
 *# Тест отсечения* def cutting(self, value):  
 if value == 2:  
 self.ui.openGLWidget.scissors\_test = True  
 else:  
 self.ui.openGLWidget.scissors\_test = False  
 self.ui.openGLWidget.scissors = [0,0,0,0]  
 self.ui.openGLWidget.makeNewPoints = False  
 self.ui.openGLWidget.update()  
  
 *# Тест прозрачности* def transparent(self):  
 self.ui.openGLWidget.update()  
  
 *# Тест смешения цветов* def blender(self):  
 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(1017, 700)  
 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  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 self.openGLWidget = None  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(1017, 700)  
 self.centralwidget = QtWidgets.QWidget(parent=MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.groupBox = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox.setGeometry(QtCore.QRect(670, 20, 221, 91))  
 self.groupBox.setTitle("")  
 self.groupBox.setObjectName("groupBox")  
 self.comboBox = QtWidgets.QComboBox(parent=self.groupBox)  
 self.comboBox.setGeometry(QtCore.QRect(10, 40, 191, 31))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 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.comboBox.addItem("")  
 self.label = QtWidgets.QLabel(parent=self.groupBox)  
 self.label.setGeometry(QtCore.QRect(10, 10, 181, 16))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.label.setFont(font)  
 self.label.setObjectName("label")  
 self.groupBox\_2 = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox\_2.setGeometry(QtCore.QRect(670, 130, 311, 81))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.groupBox\_2.setFont(font)  
 self.groupBox\_2.setObjectName("groupBox\_2")  
 self.checkBox = QtWidgets.QCheckBox(parent=self.groupBox\_2)  
 self.checkBox.setGeometry(QtCore.QRect(10, 40, 181, 20))  
 self.checkBox.setObjectName("checkBox")  
 self.groupBox\_3 = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox\_3.setGeometry(QtCore.QRect(670, 240, 311, 151))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.groupBox\_3.setFont(font)  
 self.groupBox\_3.setObjectName("groupBox\_3")  
 self.comboBox\_2 = QtWidgets.QComboBox(parent=self.groupBox\_3)  
 self.comboBox\_2.setGeometry(QtCore.QRect(10, 70, 191, 31))  
 self.comboBox\_2.setObjectName("comboBox\_2")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.comboBox\_2.addItem("")  
 self.horizontalSlider\_3 = QtWidgets.QSlider(parent=self.groupBox\_3)  
 self.horizontalSlider\_3.setGeometry(QtCore.QRect(10, 110, 291, 22))  
 self.horizontalSlider\_3.setMaximum(100)  
 self.horizontalSlider\_3.setSingleStep(10)  
 self.horizontalSlider\_3.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.horizontalSlider\_3.setObjectName("horizontalSlider\_3")  
 self.checkBox\_2 = QtWidgets.QCheckBox(parent=self.groupBox\_3)  
 self.checkBox\_2.setGeometry(QtCore.QRect(10, 30, 181, 20))  
 self.checkBox\_2.setObjectName("checkBox\_2")  
 self.groupBox\_4 = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox\_4.setGeometry(QtCore.QRect(670, 420, 291, 211))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.groupBox\_4.setFont(font)  
 self.groupBox\_4.setObjectName("groupBox\_4")  
 self.label\_4 = QtWidgets.QLabel(parent=self.groupBox\_4)  
 self.label\_4.setGeometry(QtCore.QRect(10, 80, 55, 16))  
 self.label\_4.setObjectName("label\_4")  
 self.label\_5 = QtWidgets.QLabel(parent=self.groupBox\_4)  
 self.label\_5.setGeometry(QtCore.QRect(10, 150, 55, 16))  
 self.label\_5.setObjectName("label\_5")  
 self.comboBox\_3 = QtWidgets.QComboBox(parent=self.groupBox\_4)  
 self.comboBox\_3.setGeometry(QtCore.QRect(10, 170, 261, 31))  
 self.comboBox\_3.setObjectName("comboBox\_3")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_3.addItem("")  
 self.comboBox\_4 = QtWidgets.QComboBox(parent=self.groupBox\_4)  
 self.comboBox\_4.setGeometry(QtCore.QRect(10, 100, 261, 31))  
 self.comboBox\_4.setObjectName("comboBox\_4")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.comboBox\_4.addItem("")  
 self.checkBox\_3 = QtWidgets.QCheckBox(parent=self.groupBox\_4)  
 self.checkBox\_3.setGeometry(QtCore.QRect(10, 40, 181, 20))  
 self.checkBox\_3.setObjectName("checkBox\_3")  
 self.pushButton = QtWidgets.QPushButton(parent=self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(890, 650, 93, 28))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.pushButton.setFont(font)  
 self.pushButton.setObjectName("pushButton")  
  
 MainWindow.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "lab2\_Arkhipov"))  
 self.comboBox.setItemText(0, \_translate("MainWindow", "GL\_POINTS"))  
 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.comboBox.setItemText(10, \_translate("MainWindow", "CIRCLES"))  
 self.label.setText(\_translate("MainWindow", "Выберите примитив"))  
 self.groupBox\_2.setTitle(\_translate("MainWindow", "Тест отсечения"))  
 self.checkBox.setText(\_translate("MainWindow", "Активировать"))  
 self.groupBox\_3.setTitle(\_translate("MainWindow", "Тест прозрачности"))  
 self.comboBox\_2.setItemText(0, \_translate("MainWindow", "GL\_NEVER"))  
 self.comboBox\_2.setItemText(1, \_translate("MainWindow", "GL\_LESS"))  
 self.comboBox\_2.setItemText(2, \_translate("MainWindow", "GL\_EQUAL"))  
 self.comboBox\_2.setItemText(3, \_translate("MainWindow", "GL\_LEQUAL"))  
 self.comboBox\_2.setItemText(4, \_translate("MainWindow", "GL\_GREATER"))  
 self.comboBox\_2.setItemText(5, \_translate("MainWindow", "GL\_NOTEQUAL"))  
 self.comboBox\_2.setItemText(6, \_translate("MainWindow", "GL\_GEQUAL"))  
 self.comboBox\_2.setItemText(7, \_translate("MainWindow", "GL\_ALWAYS"))  
 self.checkBox\_2.setText(\_translate("MainWindow", "Активировать"))  
 self.groupBox\_4.setTitle(\_translate("MainWindow", "Тест смешения цветов"))  
 self.label\_4.setText(\_translate("MainWindow", "sfactor"))  
 self.label\_5.setText(\_translate("MainWindow", "dfactor"))  
 self.comboBox\_3.setItemText(0, \_translate("MainWindow", "GL\_ZERO"))  
 self.comboBox\_3.setItemText(1, \_translate("MainWindow", "GL\_ONE"))  
 self.comboBox\_3.setItemText(2, \_translate("MainWindow", "GL\_SRC\_COLOR"))  
 self.comboBox\_3.setItemText(3, \_translate("MainWindow", "GL\_ONE\_MINUS\_SRC\_COLOR"))  
 self.comboBox\_3.setItemText(4, \_translate("MainWindow", "GL\_SRC\_ALPHA"))  
 self.comboBox\_3.setItemText(5, \_translate("MainWindow", "GL\_ONE\_MINUS\_SRC\_ALPHA"))  
 self.comboBox\_3.setItemText(6, \_translate("MainWindow", "GL\_DST\_ALPHA"))  
 self.comboBox\_3.setItemText(7, \_translate("MainWindow", "GL\_ONE\_MINUS\_DST\_ALPHA"))  
 self.comboBox\_4.setItemText(0, \_translate("MainWindow", "GL\_ZERO"))  
 self.comboBox\_4.setItemText(1, \_translate("MainWindow", "GL\_ONE"))  
 self.comboBox\_4.setItemText(2, \_translate("MainWindow", "GL\_DST\_COLOR"))  
 self.comboBox\_4.setItemText(3, \_translate("MainWindow", "GL\_ONE\_MINUS\_DST\_COLOR"))  
 self.comboBox\_4.setItemText(4, \_translate("MainWindow", "GL\_SRC\_ALPHA"))  
 self.comboBox\_4.setItemText(5, \_translate("MainWindow", "GL\_ONE\_MINUS\_SRC\_ALPHA"))  
 self.comboBox\_4.setItemText(6, \_translate("MainWindow", "GL\_DST\_ALPHA"))  
 self.comboBox\_4.setItemText(7, \_translate("MainWindow", "GL\_ONE\_MINUS\_DST\_ALPHA "))  
 self.comboBox\_4.setItemText(8, \_translate("MainWindow", "GL\_SRC\_ALPHA\_SATURATE"))  
 self.checkBox\_3.setText(\_translate("MainWindow", "Активировать"))  
 self.pushButton.setText(\_translate("MainWindow", "Выход"))