МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Компьютерная графика»
Тема: Примитивы OpenGL

Выполнила: Сергеев А.Д.
Факультет: ФКТИ
Группа: 8304
Преподаватель: Герасимова Т.В.

Санкт-Петербург
2021

Задание.

На базе предложенного шаблона 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.

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 – cap, 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».

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.

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.

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

К интерфейсу программы из лабораторной работы №1 были добавлены три переключателя управляющие необходимыми тестами. При включении переключателя разблокируются соответствующие ему настройки в виде выпадающих списков (в тех случаях, когда необходимо выбрать какой-либо режим) или слайдеров со 100 делениями каждый (в тех случаях, когда необходимо выбрать значение от 0 до 1).

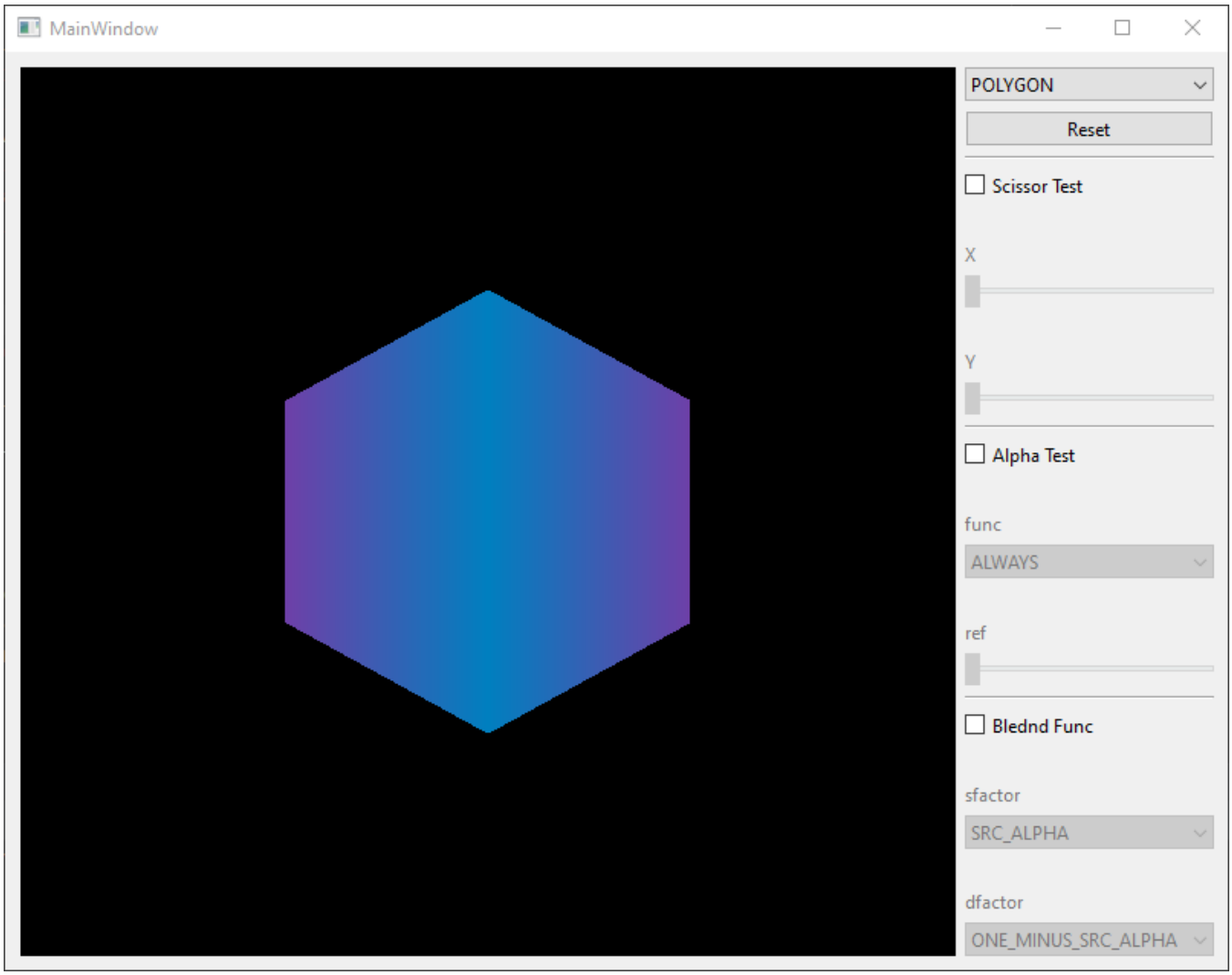

Рисунок 1 – Фигура без применения тестов

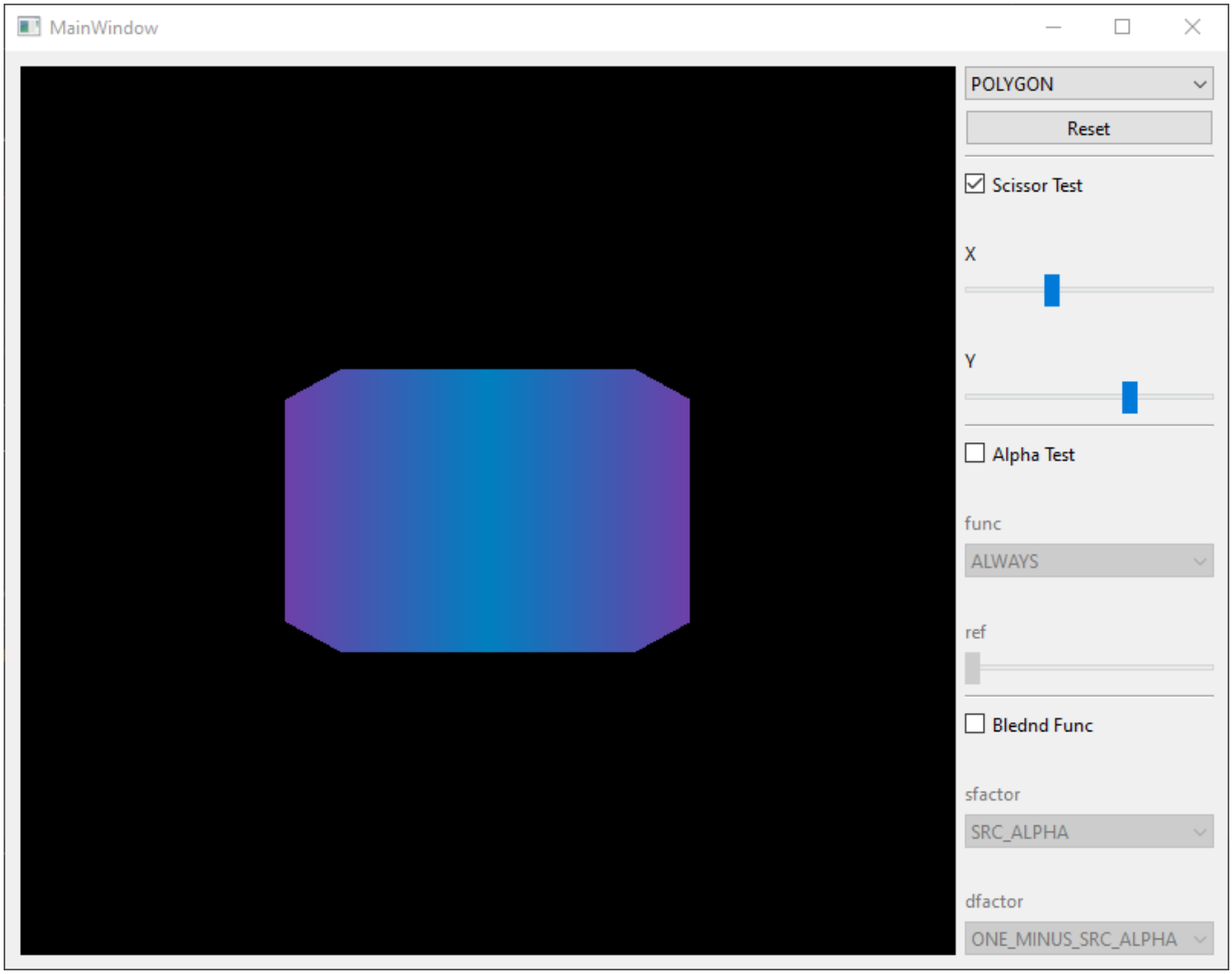

Рисунок 2 – Фигура с включенным тестом на отсечение

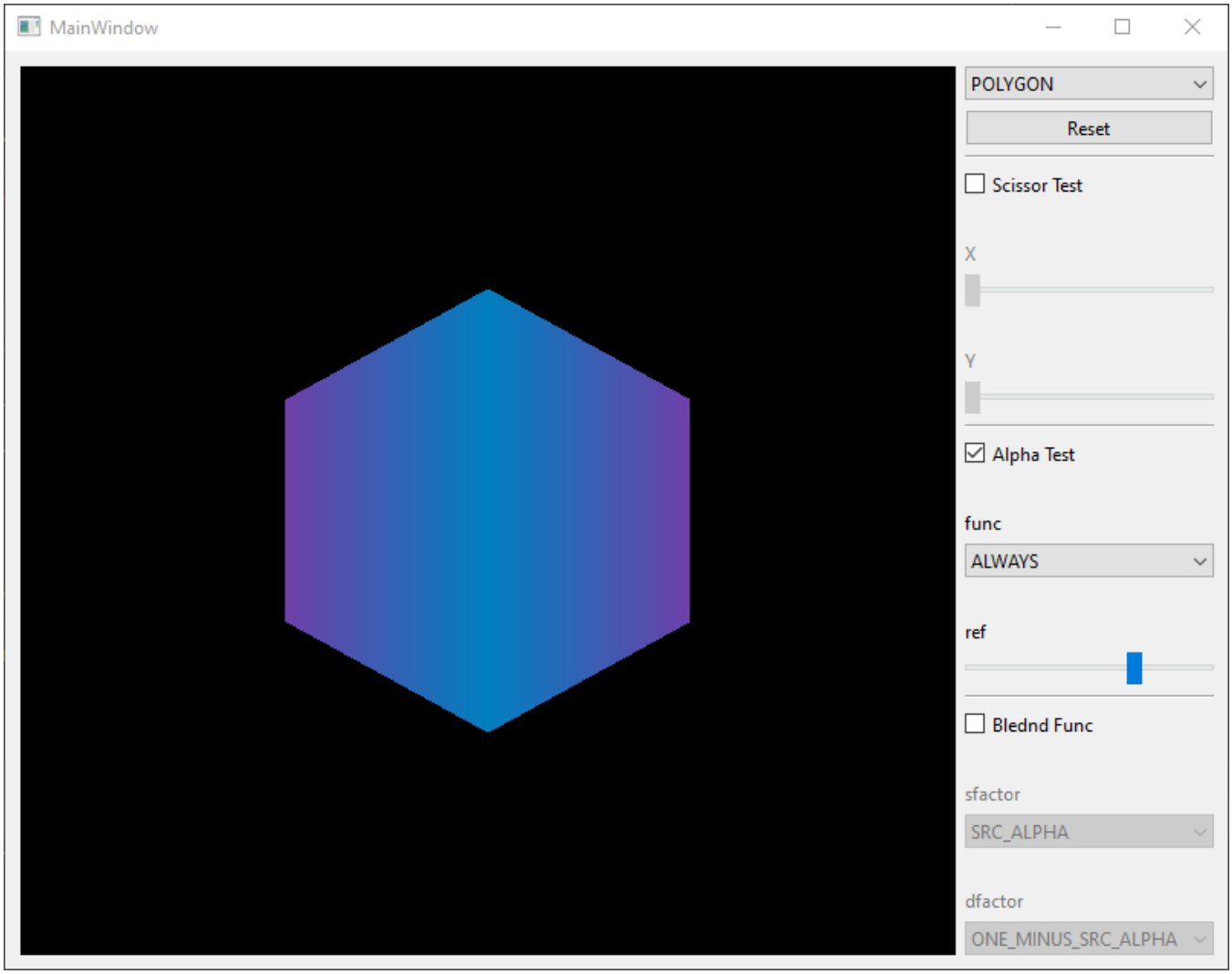

Рисунок 3 - Фигура с включенным тестом на прозрачность

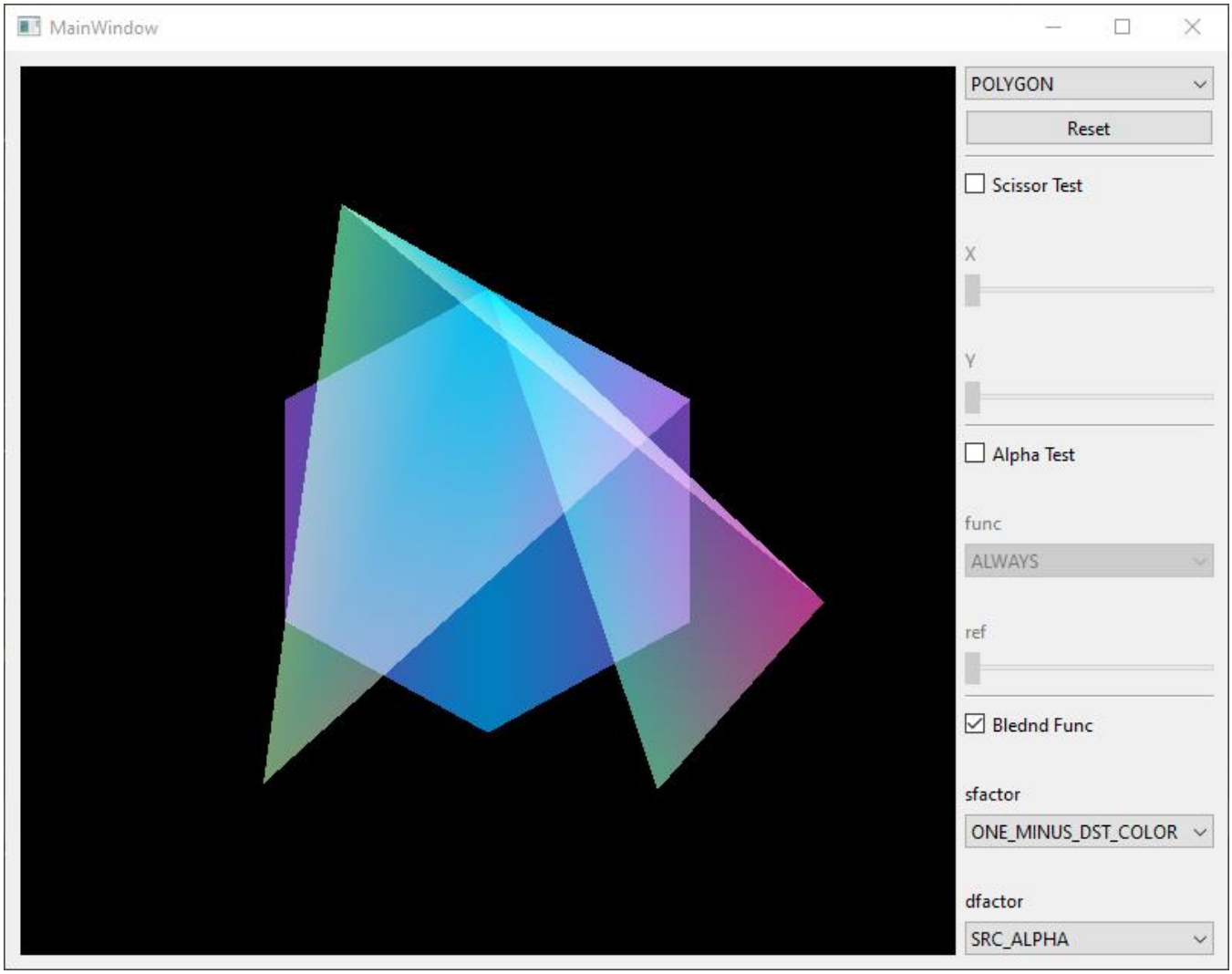

Рисунок 4 - Фигура с включенным тестом на смешение цветов

Выводы

В результате выполнения лабораторной работы была разработана программа, реализующая отрисовку графических примитивов OpenGL и применение тестов на отсечение, смешение цветов и прозрачность к получившимся фигурам. При тестировании ошибок выявлено не было. При выполнении работы были приобретены знания об использовании режимов в графической библиотеке OpenGL.

Приложение А. Исходный код программы.

Файл *main.py*:

```
import sys
from enum import Enum
from OpenGL.GL import *
from PyQt6 import QtWidgets, uic
from PyQt6.QtWidgets import QCheckBox, QVBoxLayout, QLabel, QSlider, QComboBox

from GLTest import GLTest, Func, SFactor, DFactor



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_test(display, win, test_name):
    check = win.findChild(QCheckBox, test_name + 'Test')
    layout = win.findChild(QVBoxLayout, test_name + 'Args')
    children = [layout.itemAt(x).widget() for x in range(0, layout.count())]

    def toggle_test(checked):
        if test_name == 'scissor':
            test = GLTest.SCISSOR
        elif test_name == 'alpha':
            test = GLTest.ALPHA_FUNK
        elif test_name == 'blend':
            test = GLTest.BLEND_FUNK
        else:
            return
        display.toggle_test(test, checked)
        for c in children:
            c.setEnabled(checked)

    check.stateChanged.connect(toggle_test)

    def set_arg(arg_name, arg_val):
```

```
            GLTest.set_arg({arg_name: arg_val})
            display.update()

        for child in children:
            if isinstance(child, QLabel):
                continue
            elif isinstance(child, QSlider):
                child.valueChanged.connect(lambda value, c=child: set_arg(c.objectName(), value / 100))
                child.setRange(0, 100)
                child.setValue(0)
                set_arg(child.objectName(), 0)
            elif isinstance(child, QComboBox):
                if child.objectName() == 'func':
                    enumerator = Func
                elif child.objectName() == 'sfactor':
                    enumerator = SFactor
                elif child.objectName() == 'dfactor':
                    enumerator = DFactor
                else:
                    continue
                child.currentIndexChanged.connect(lambda    index,    c=child:    set_arg(c.objectName(),
c.itemData(index)))
                for enum in enumerator:
                    child.addItem(enum.name, enum.value)


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())

    configure_test(display, win, 'scissor')
    configure_test(display, win, 'alpha')
    configure_test(display, win, 'blend')


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
from GLTest import GLTest


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
        self._tests = []

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT)
        for test in self._tests:
            glEnable(test.value)
            test.apply()
        glBegin(self._mode)
        for vertex in self._vert:
            glColor3f(abs(vertex[0]), abs(vertex[1]), abs(vertex[2]))
            glVertex2f(vertex[0], vertex[1])
        glEnd()
        for test in self._tests:
            glDisable(test.value)

    def resizeGL(self, w, h):
        GLTest.set_arg({'size_x': w, 'size_y': h})

    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()

    def toggle_test(self, test, exists):
        if (test in self._tests) and not exists:
            self._tests.remove(test)
        elif (test not in self._tests) and exists:
            self._tests.append(test)
        self.update()
```

Файл *GLTest.py*:

```
from enum import Enum
from OpenGL.GL import *


class Func(Enum):
    ALWAYS = GL_ALWAYS
    NEVER = GL_NEVER
    LESS = GL_LESS
    EQUAL = GL_EQUAL
    LEQUAL = GL_LEQUAL
    GREATER = GL_GREATER
    NOTEQUAL = GL_NOTEQUAL
    GEQUAL = GL_GEQUAL


class SFactor(Enum):
    SRC_ALPHA = GL_SRC_ALPHA
    ZERO = GL_ZERO
    ONE = GL_ONE
    DST_COLOR = GL_DST_COLOR
    ONE_MINUS_DST_COLOR = GL_ONE_MINUS_DST_COLOR
    ONE_MINUS_SRC_ALPHA = GL_ONE_MINUS_SRC_ALPHA
```

```
    DST_ALPHA = GL_DST_ALPHA
    ONE_MINUS_DST_ALPHA = GL_ONE_MINUS_DST_ALPHA
    SRC_ALPHA_SATURATE = GL_SRC_ALPHA_SATURATE


class DFactor(Enum):
    ONE_MINUS_SRC_ALPHA = GL_ONE_MINUS_SRC_ALPHA
    ZERO = GL_ZERO
    ONE = GL_ONE
    SRC_COLOR = GL_SRC_COLOR
    ONE_MINUS_SRC_COLOR = GL_ONE_MINUS_SRC_COLOR
    SRC_ALPHA = GL_SRC_ALPHA
    DST_ALPHA = GL_DST_ALPHA
    ONE_MINUS_DST_ALPHA = GL_ONE_MINUS_DST_ALPHA


class GLTest(Enum):
    _ignore_ = ['_args']
    _args = {}

    @classmethod
    def _apply_scissor(cls):
        side_x = int(cls._args['size_x'] * (1 - cls._args['rad_x']))
        side_y = int(cls._args['size_y'] * (1 - cls._args['rad_y']))
        offset_x = int((cls._args['size_x'] - side_x) / 2)
        offset_y = int((cls._args['size_y'] - side_y) / 2)
        glScissor(offset_x, offset_y, side_x, side_y)

    @classmethod
    def _apply_alpha(cls):
        glAlphaFunc(cls._args['func'], cls._args['ref'])

    @classmethod
    def _apply_blend(cls):
        glBlendFunc(cls._args['sfactor'], cls._args['dfactor'])

    def apply(self):
        pass

    def __init__(self, value):
        if value is GL_SCISSOR_TEST:
            self.apply = self._apply_scissor
        elif value is GL_ALPHA_TEST:
            self.apply = self._apply_alpha
        elif value is GL_BLEND:
            self.apply = self._apply_blend
```

```
    @classmethod
    def set_arg(cls, arg):
        cls._args.update(arg)

    SCISSOR = GL_SCISSOR_TEST
    ALPHA_FUNK = GL_ALPHA_TEST
    BLEND_FUNK = GL_BLEND


GLTest._args = {
    'rad_x': 0, 'rad_y': 0, 'size_x': 0, 'size_y': 0,
    'func': GL_ALWAYS, 'ref': 0,
    'sfactor': GL_SRC_ALPHA, 'dfactor': GL_ONE_MINUS_SRC_ALPHA
}
```

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" stretch="8,2">
     <item>
      <widget class="GlWidget" name="mainGLWidget"/>
```

```
</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>
  <item>
   <widget class="Line" name="line">
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
   </widget>
  </item>
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_2">
    <item>
     <widget class="QCheckBox" name="scissorTest">
      <property name="text">
       <string>Scissor Test</string>
      </property>
     </widget>
    </item>
    <item>
     <layout class="QVBoxLayout" name="scissorArgs">
      <item>
       <widget class="QLabel" name="label_3">
        <property name="enabled">
         <bool>false</bool>
        </property>
        <property name="text">
         <string>X</string>
        </property>
        <property name="alignment">
         <set>Qt::AlignBottom|Qt::AlignLeading|Qt::AlignLeft</set>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QSlider" name="rad_x">
```

```
      <property name="enabled">
       <bool>false</bool>
      </property>
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_4">
      <property name="enabled">
       <bool>false</bool>
      </property>
      <property name="text">
       <string>Y</string>
      </property>
      <property name="alignment">
       <set>Qt::AlignBottom|Qt::AlignLeading|Qt::AlignLeft</set>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QSlider" name="rad_y">
      <property name="enabled">
       <bool>false</bool>
      </property>
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
     </widget>
    </item>
   </layout>
  </item>
 </layout>
</item>
<item>
 <widget class="Line" name="line_2">
  <property name="orientation">
   <enum>Qt::Horizontal</enum>
  </property>
 </widget>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_4">
  <item>
   <widget class="QCheckBox" name="alphaTest">
```

```
  <property name="text">
   <string>Alpha Test</string>
  </property>
 </widget>
</item>
<item>
 <layout class="QVBoxLayout" name="alphaArgs">
  <item>
   <widget class="QLabel" name="label">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="text">
     <string>func</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignBottom|Qt::AlignLeading|Qt::AlignLeft</set>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QComboBox" name="func">
    <property name="enabled">
     <bool>false</bool>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="label_2">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="text">
     <string>ref</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignBottom|Qt::AlignLeading|Qt::AlignLeft</set>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QSlider" name="ref">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="orientation">
```

```
       <enum>Qt::Horizontal</enum>
      </property>
     </widget>
    </item>
   </layout>
  </item>
 </layout>
</item>
<item>
 <widget class="Line" name="line_3">
  <property name="orientation">
   <enum>Qt::Horizontal</enum>
  </property>
 </widget>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_3">
  <item>
   <widget class="QCheckBox" name="blendTest">
    <property name="text">
     <string>Blednd Func</string>
    </property>
   </widget>
  </item>
  <item>
   <layout class="QVBoxLayout" name="blendArgs">
    <item>
     <widget class="QLabel" name="label_5">
      <property name="enabled">
       <bool>false</bool>
      </property>
      <property name="text">
       <string>sfactor</string>
      </property>
      <property name="alignment">
       <set>Qt::AlignBottom|Qt::AlignLeading|Qt::AlignLeft</set>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QComboBox" name="sfactor">
      <property name="enabled">
       <bool>false</bool>
      </property>
     </widget>
    </item>
```

```
            <item>
             <widget class="QLabel" name="label_6">
              <property name="enabled">
               <bool>false</bool>
              </property>
              <property name="text">
               <string>dfactor</string>
              </property>
              <property name="alignment">
               <set>Qt::AlignBottom|Qt::AlignLeading|Qt::AlignLeft</set>
              </property>
             </widget>
            </item>
            <item>
             <widget class="QComboBox" name="dfactor">
              <property name="enabled">
               <bool>false</bool>
              </property>
             </widget>
            </item>
           </layout>
          </item>
         </layout>
        </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>
```