# 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 – 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».

*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).

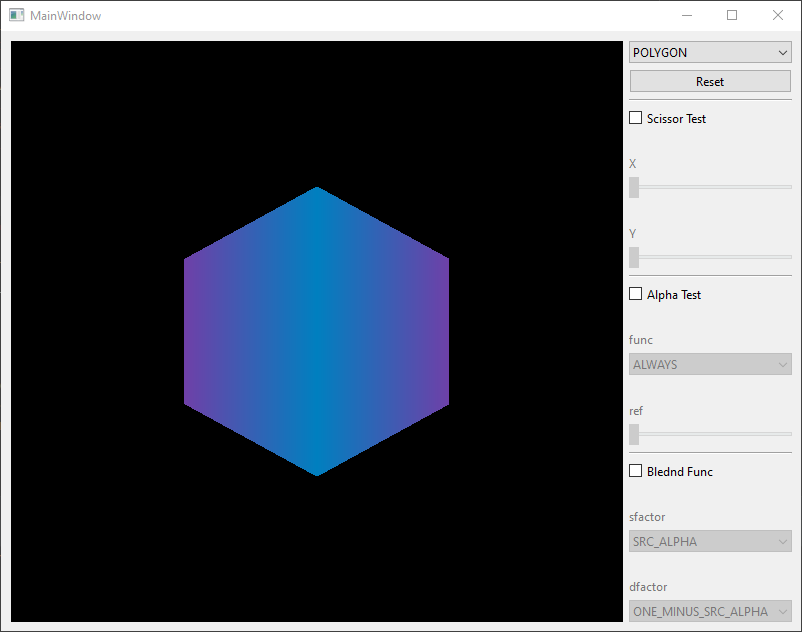


Рисунок 1 – Фигура без применения тестов

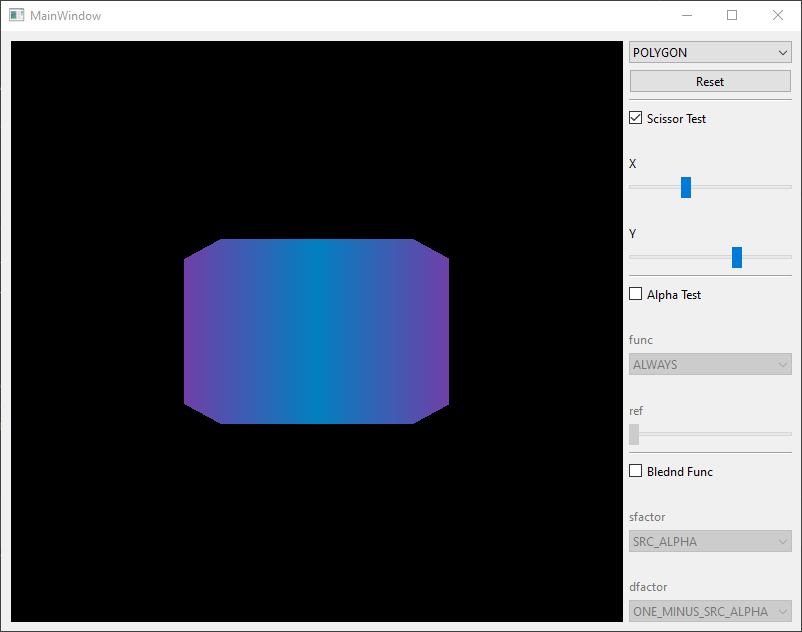


Рисунок 2 – Фигура с включенным тестом на отсечение

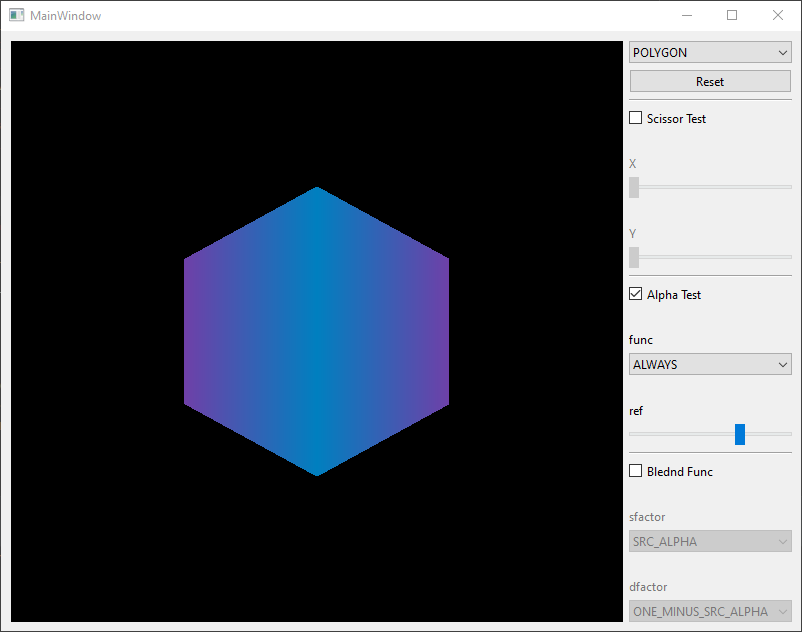


Рисунок 3 - Фигура с включенным тестом на прозрачность

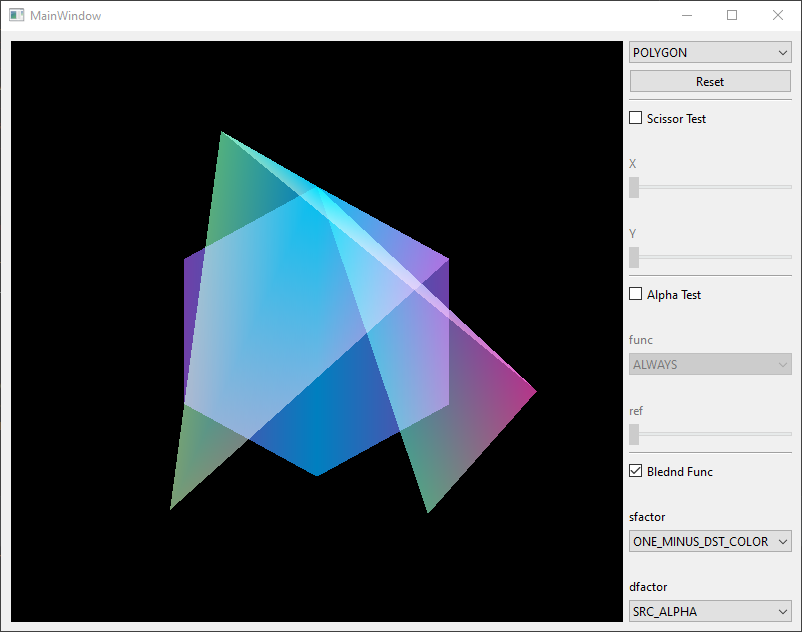


Рисунок 4 - Фигура с включенным тестом на смешение цветов

**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>