Санкт-Петербургский Государственный
Электротехнический Университет

Кафедра МОЭВМ

Задание для лабораторной работы № 2
"Примитивы OpenGL"

Выполнили:
Быков, Спас, 6383

Факультет: ФКТИ

Преподаватель: Герасимова Т.В.

Санкт-Петербург
2019 г.

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.

Выполнение работы

Работа выполнена в среде разработки Qt 5.9.5. На основе выполненной программы из первой лабораторной работы создается программа с дополнительным функционалом представления тестов отсечения, прозрачности и смешения цветов. Для удобного тестирования создадим фигуру из трех кругов. Код представлен ниже:

```
GLfloat theta;
GLfloat pi     = acos(-1.0);
GLfloat radius = 0.4f;
GLfloat step   = 1.0f;

glBegin(GL_TRIANGLE_FAN);
for(GLfloat a = 0.0f; a < 360.0f; a += step) {
    theta = 2.0f * pi * a / 180.0f;
    glColor4f(a/360, 0, 0, a/360);
    glVertex3f(radius * cos(theta)-0.3, radius * sin(theta)-0.3, 0.0f);
}
glEnd();

glBegin(GL_TRIANGLE_FAN);
for(GLfloat a = 0.0f; a < 360.0f; a += step) {
    theta = 2.0f * pi * a / 180.0f;
    glColor4f(0, a/360, 0,  a/360);
    glVertex3f(radius * cos(theta), radius * sin(theta)+0.1, 0.0f);
}
glEnd();
```

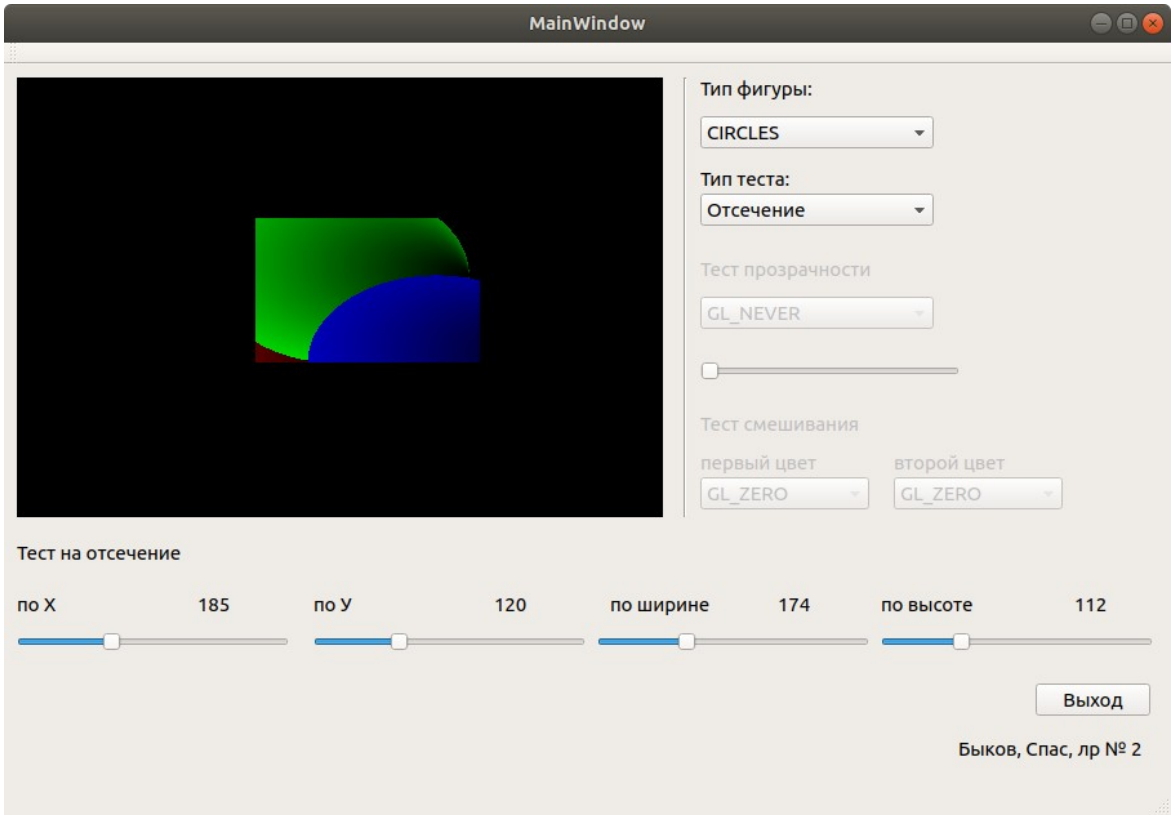
```
glBegin(GL_TRIANGLE_FAN);
for(GLfloat a = 0.0f; a < 360.0f; a += step) {
    theta = 2.0f * pi * a / 180.0f;
    glColor4f(0, 0, a/360,  a/360);
    glVertex3f(radius * cos(theta)+0.3, radius * sin(theta)-0.3, 0.0f);
}
glEnd();
```

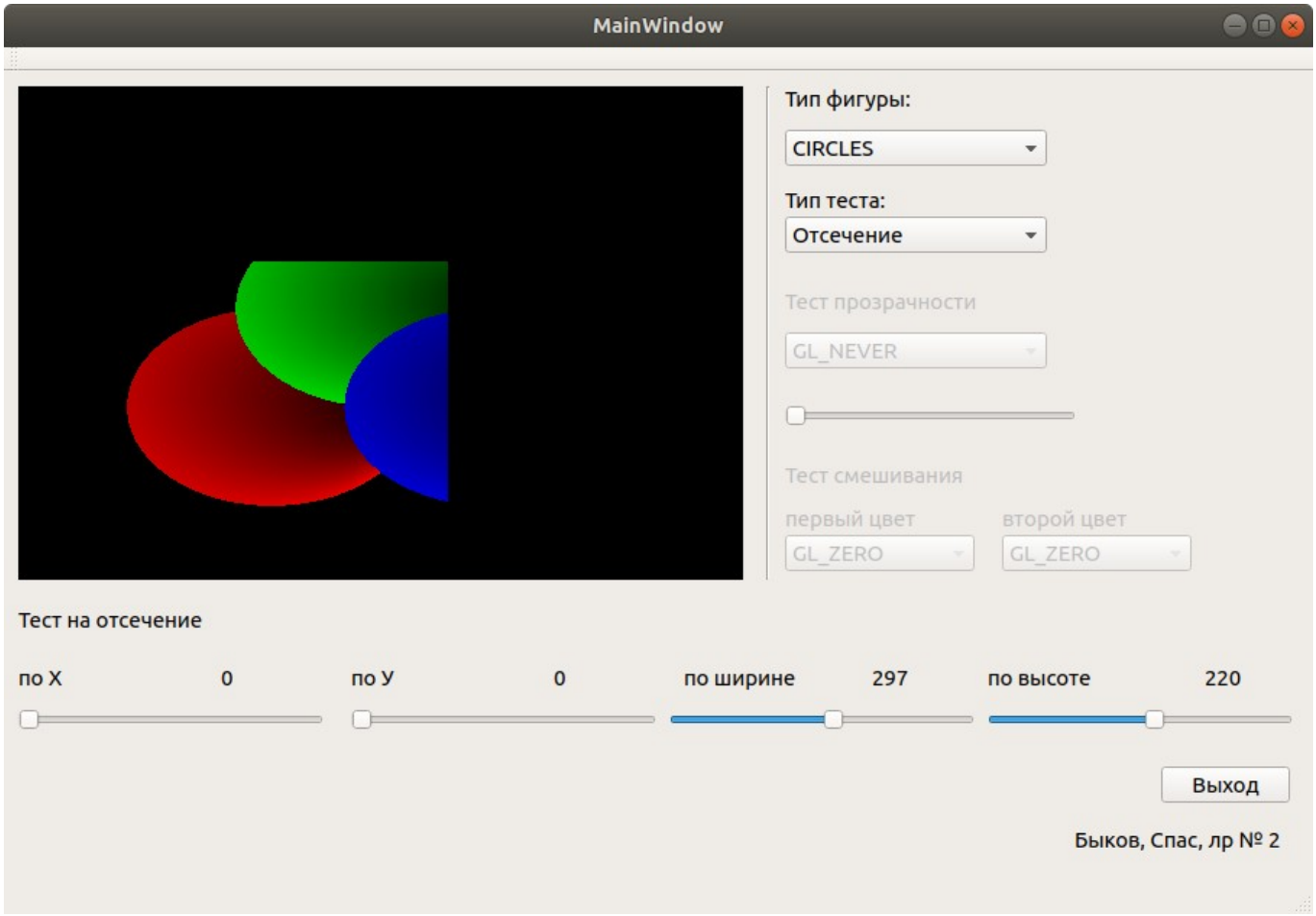
Пользователем выбирается фигура для тестирования и тип теста. При выборе определенного теста из соответствующего combobox становится доступным данный тест. В тесте прозрачности значения для параметров **void glAlphaFunc(GLenum func, GLclampf ref)** считываются из combobox и слайдера соответственно. В тесте смешения значения для параметров **void glBlendFunc(GLenum sfactor, GLenum dfactor)** считываются из соответствующих combobox. В тесте отсечения значения для параметров **void glScissor(GLint x, GLint y, GLsizei width, GLsizei height)** считываются из слайдеров.

Тестирование

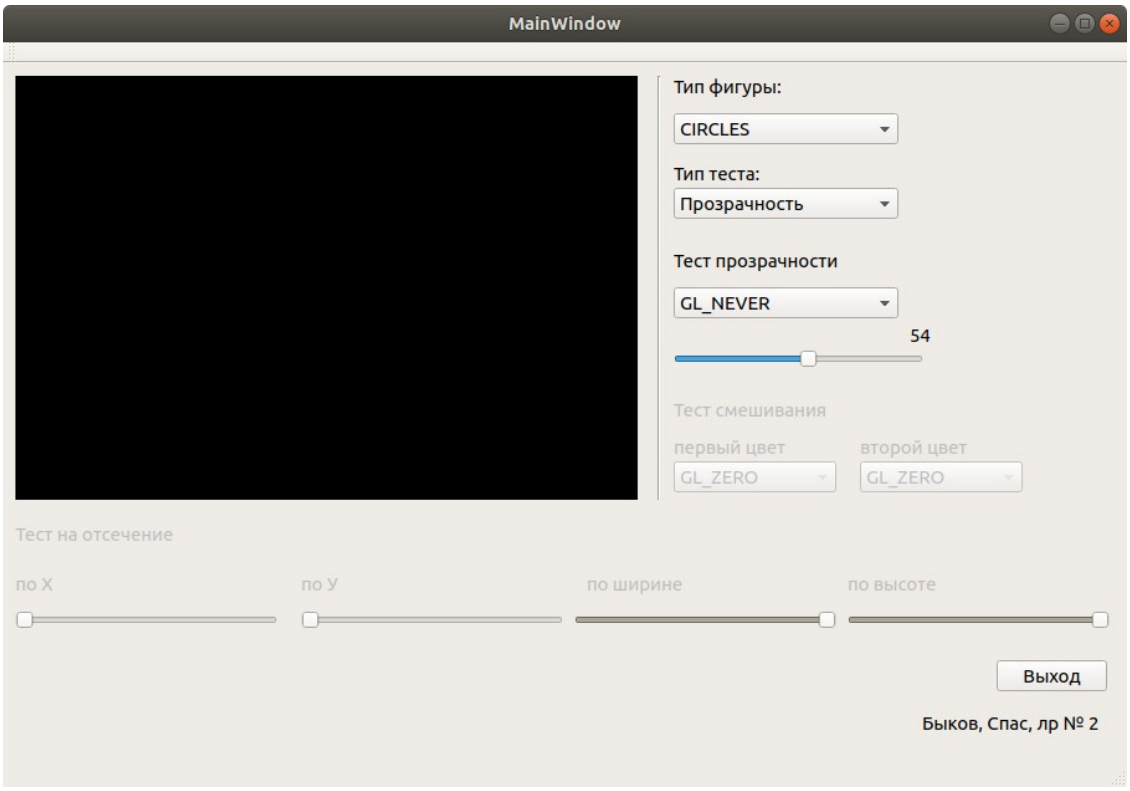
Тестирование проводилось методом черного ящика. Результаты тестирования представлены на снимках экрана.

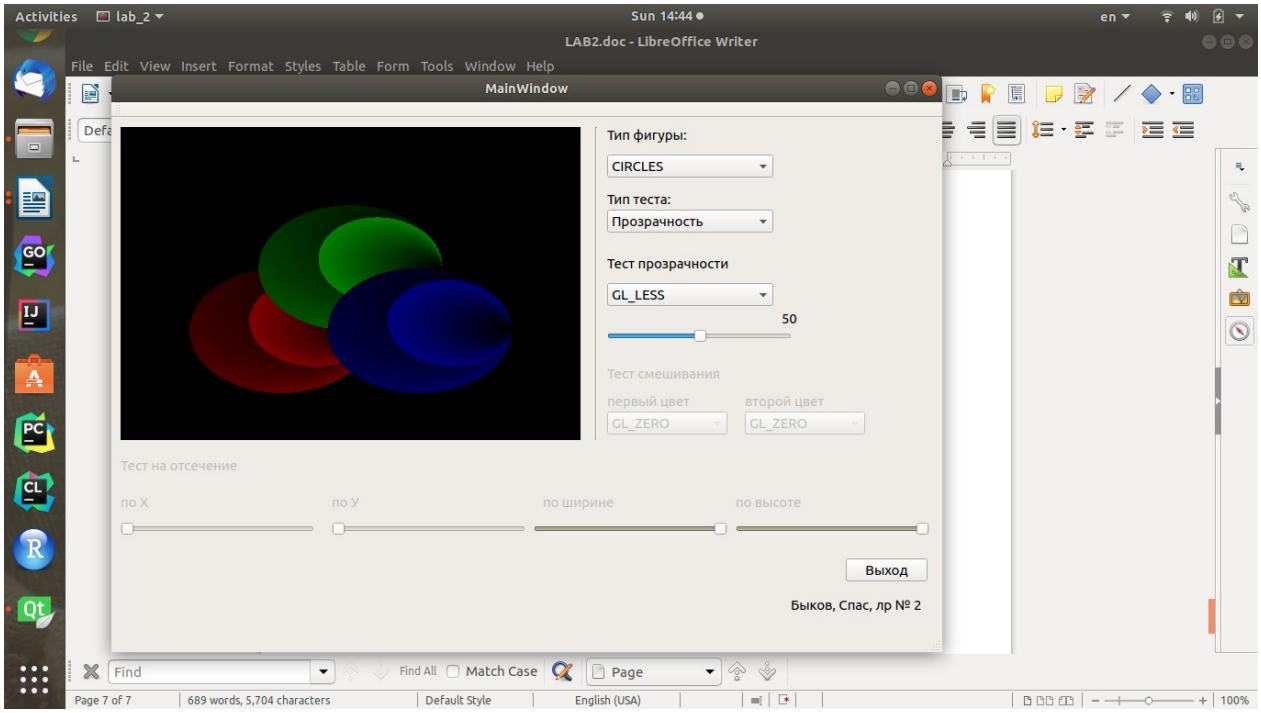
Тестирование отсечения:

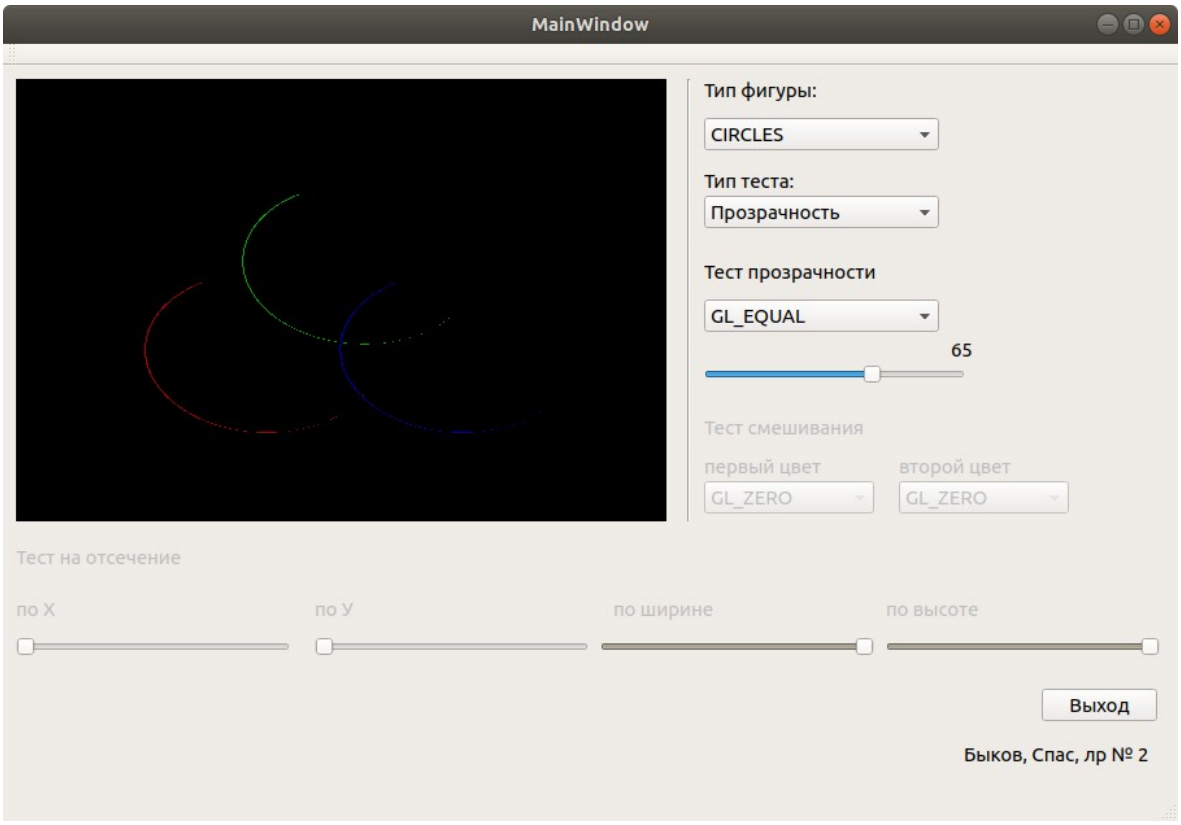


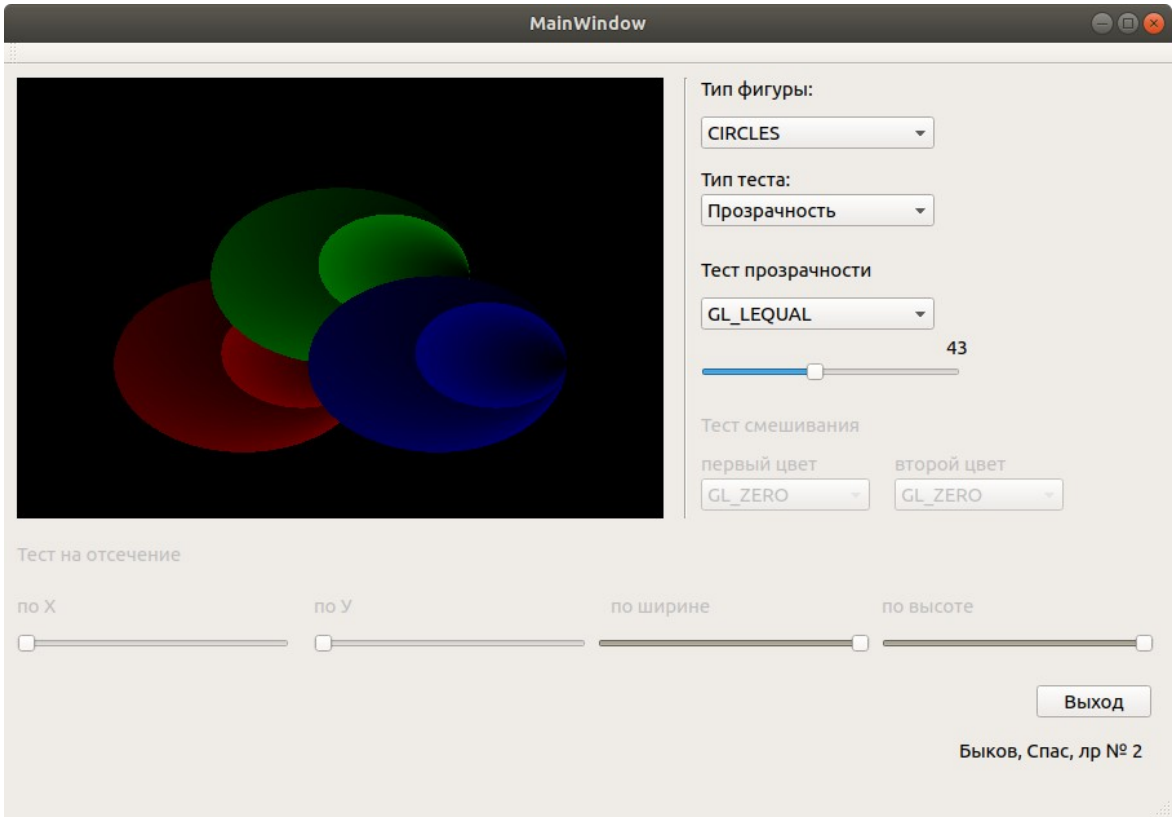

MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Отсечение
Тест прозрачности
GL_NEVER
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
0
по У
0
по ширине
297
по высоте
220
Выход
Быков, Спас, лр № 2

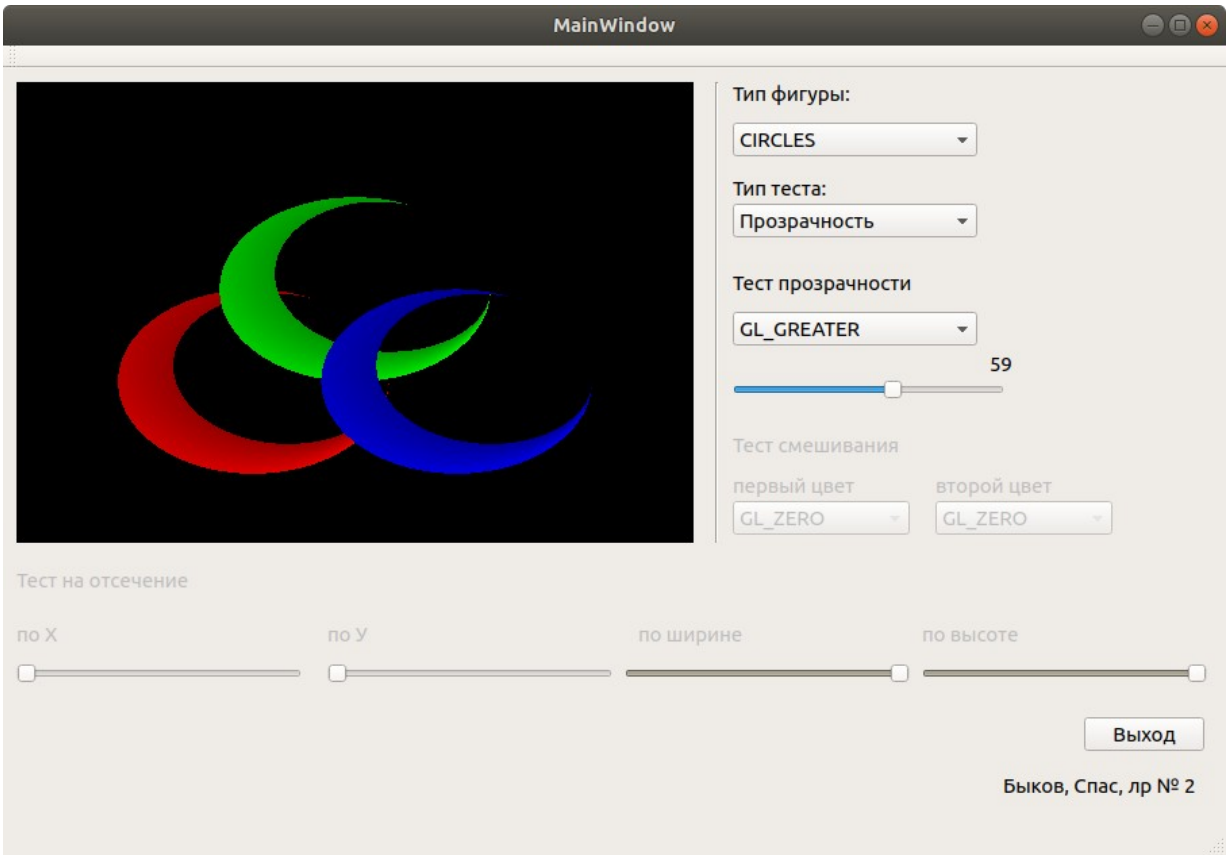
Тестирование прозрачности

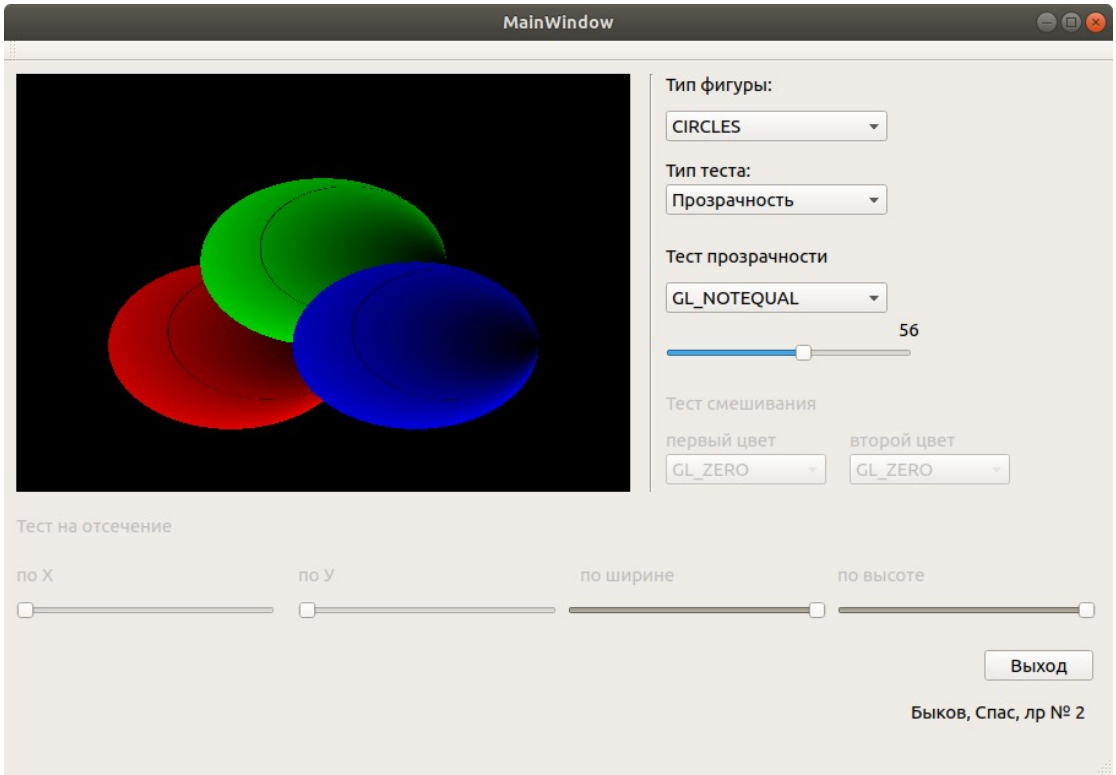


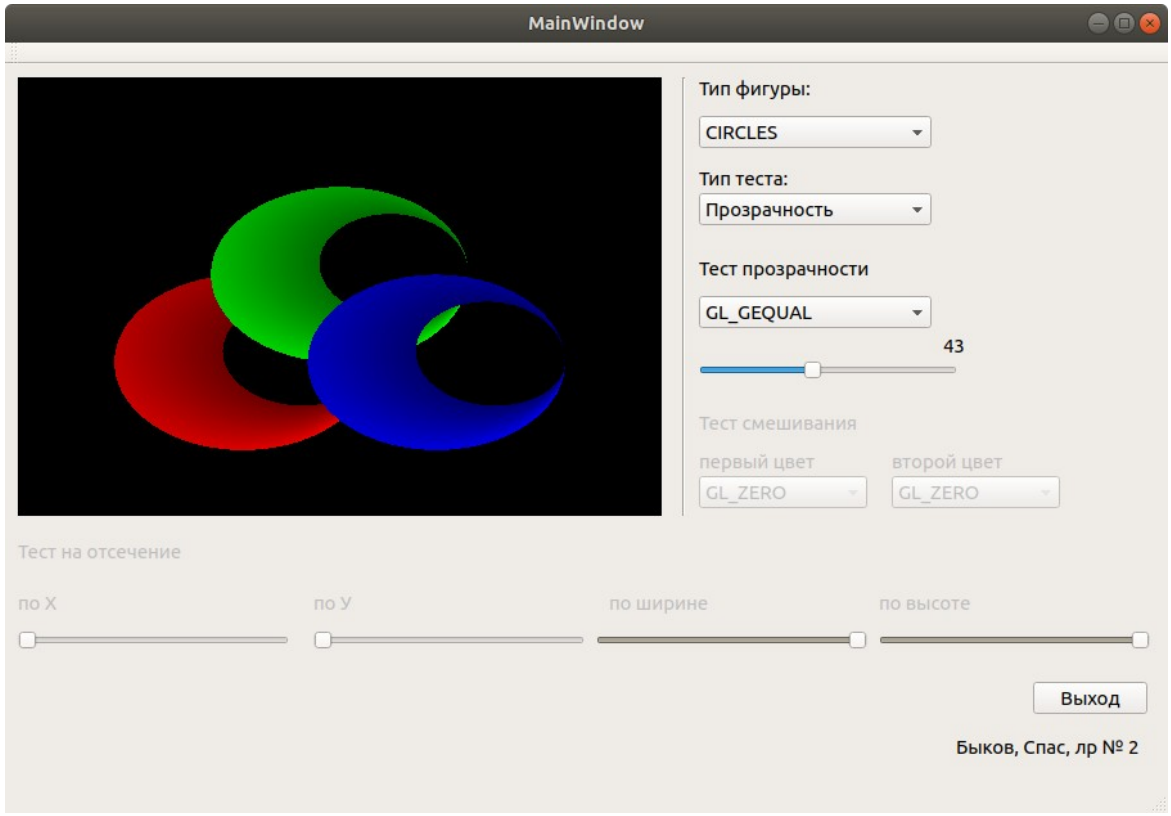


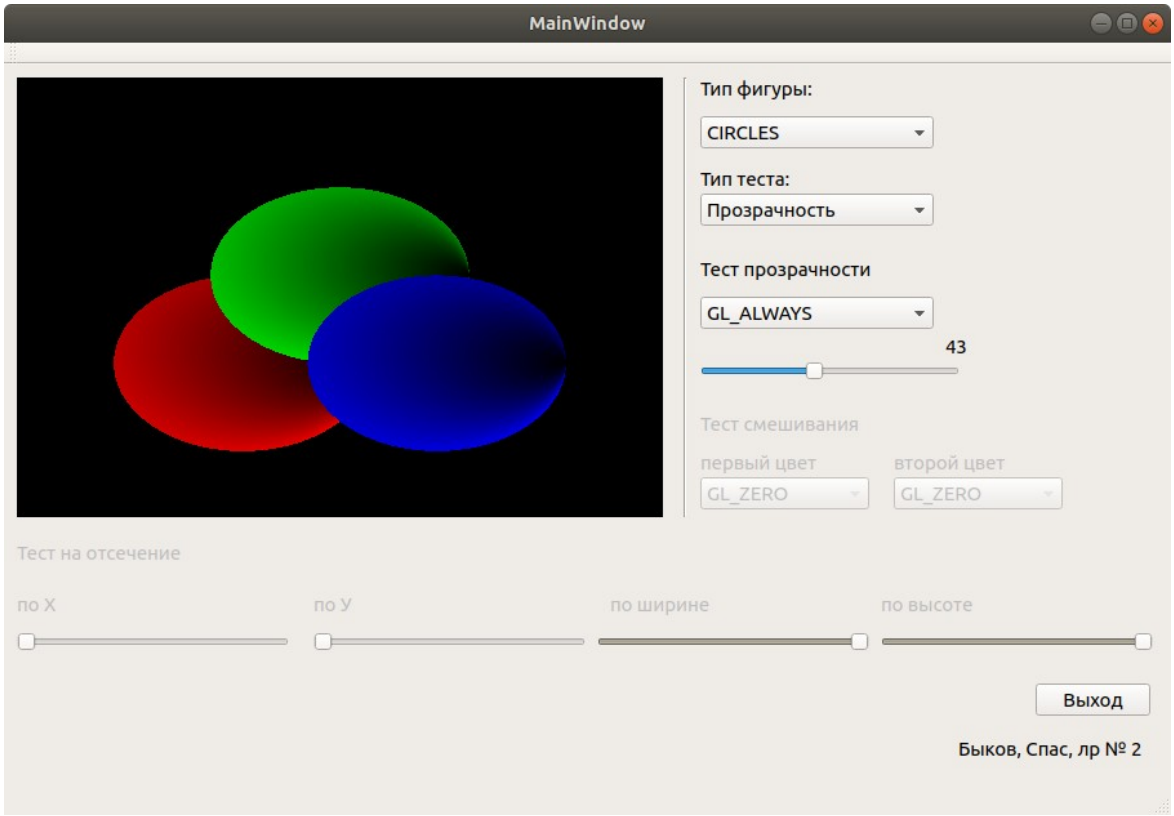

MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Прозрачность
Тест прозрачности
GL_EQUAL
65
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2


MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Прозрачность
Тест прозрачности
GL_LEQUAL
43
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2

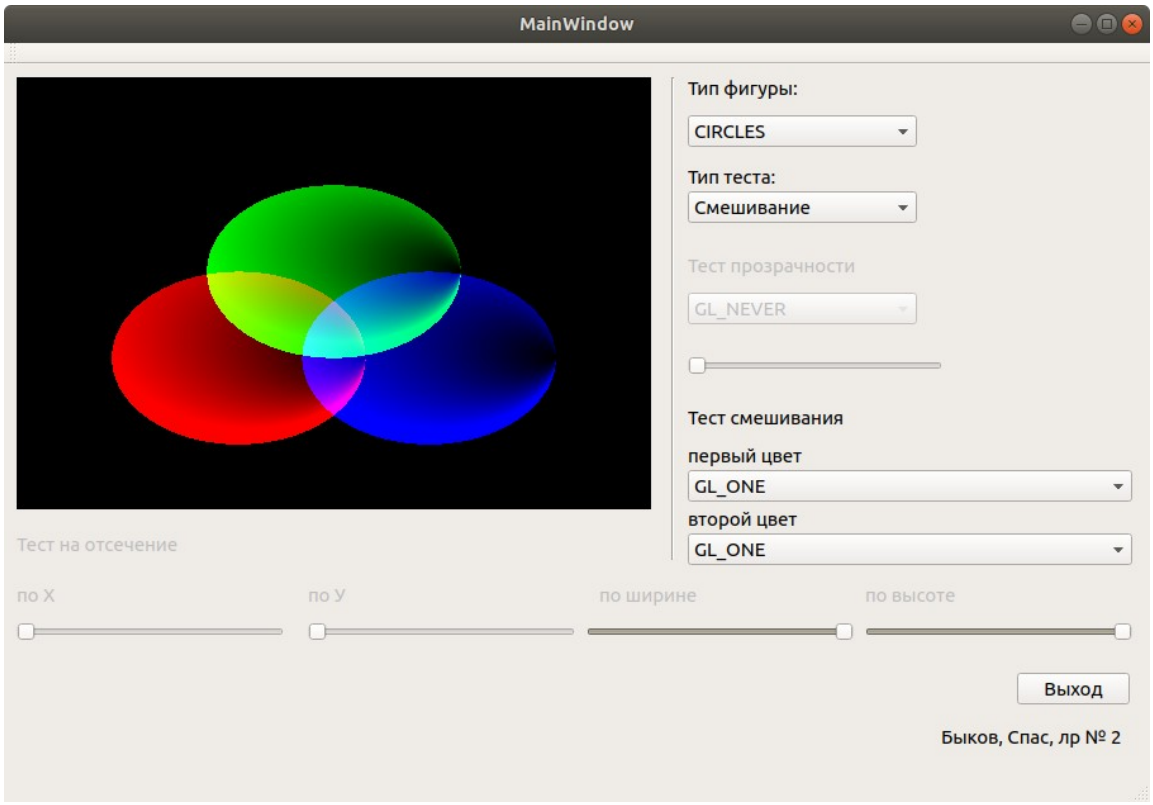

MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Прозрачность
Тест прозрачности
GL_GREATER
59
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2

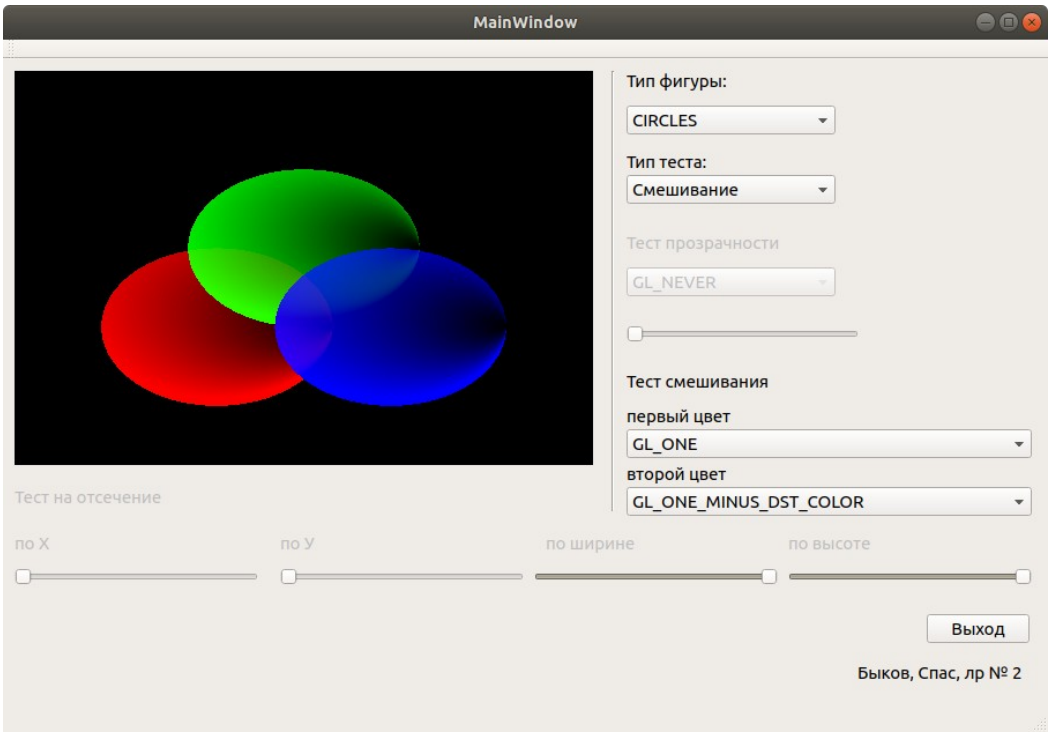

MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Прозрачность
Тест прозрачности
GL_NOTEQUAL
56
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2

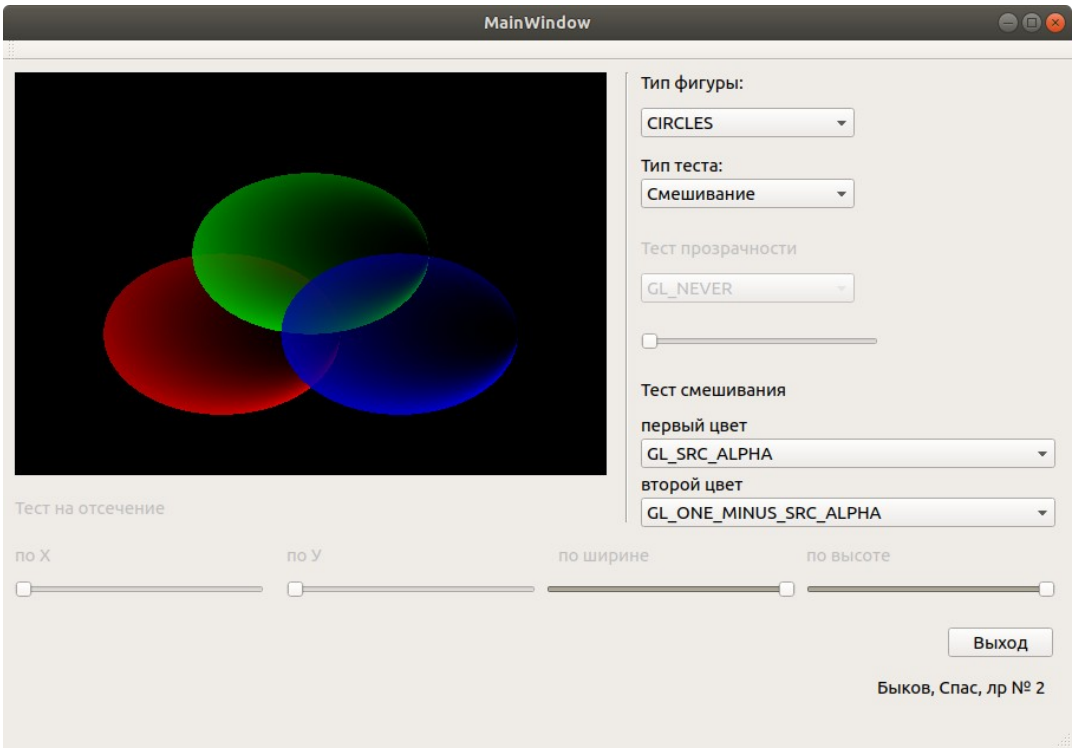

MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Прозрачность
Тест прозрачности
GL_GEQUAL
43
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2

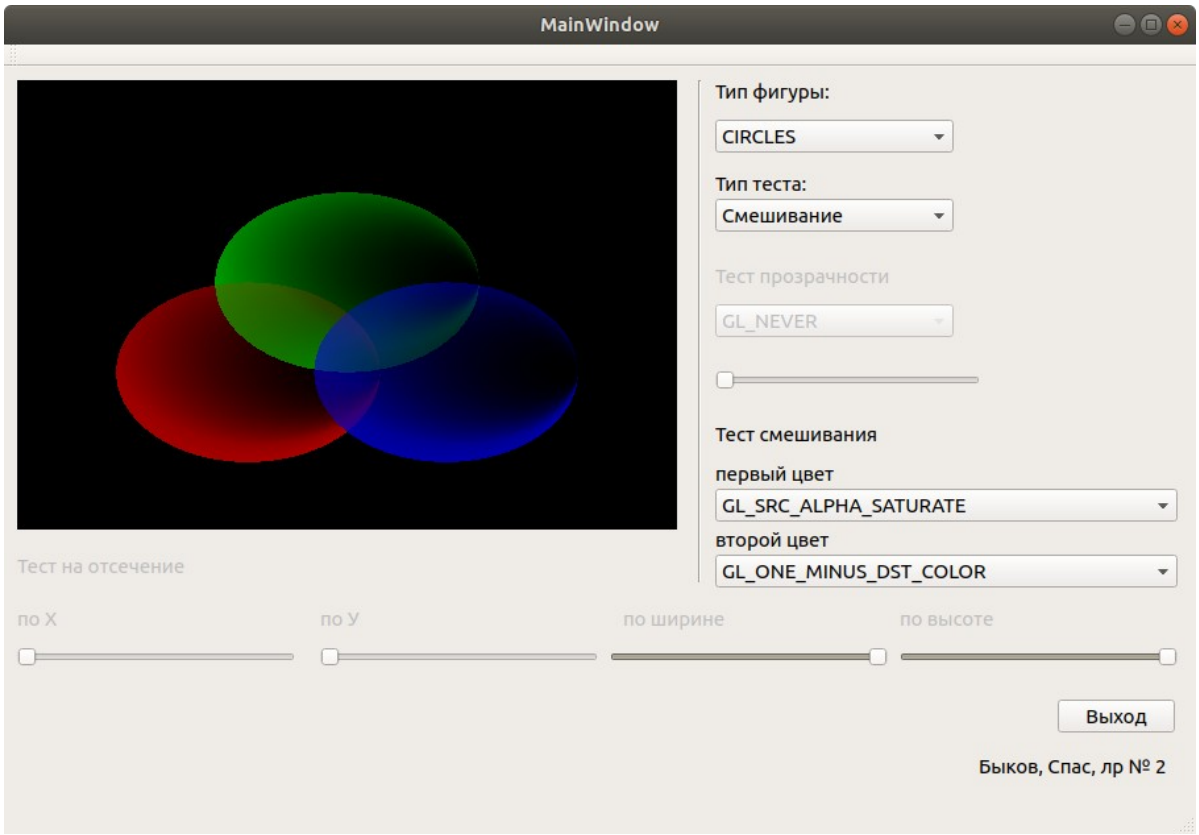

MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Прозрачность
Тест прозрачности
GL_ALWAYS
43
Тест смешивания
первый цвет
второй цвет
GL_ZERO
GL_ZERO
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2

Тестирование смешения






MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Смешивание
Тест прозрачности
GL_NEVER
Тест смешивания
первый цвет
GL_SRC_ALPHA
второй цвет
GL_ONE_MINUS_SRC_ALPHA
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2


MainWindow
Тип фигуры:
CIRCLES
Тип теста:
Смешивание
Тест прозрачности
GL_NEVER
Тест смешивания
первый цвет
GL_SRC_ALPHA_SATURATE
второй цвет
GL_ONE_MINUS_DST_COLOR
Тест на отсечение
по X
по У
по ширине
по высоте
Выход
Быков, Спас, лр № 2

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. Коды программ

glwidget.cpp

#include "glwidget.h"

```
#include "mainwindow.h"
#include <QMessageBox>
#include <vector>
#include <cstdlib>
#include <QDebug>

std::vector<GLenum> arrOfFigures =
{GL_POINTS,GL_LINES,GL_LINE_STRIP,GL_LINE_LOOP,GL_TRIANGLES,GL_TRIANGLE_STRIP,

GL_TRIANGLE_FAN,GL_QUADS,GL_QUAD_STRIP,GL_POLYGON, GL_CCW};

std::vector<GLenum> opacityTests = {GL_NEVER, GL_LESS, GL_EQUAL, GL_LEQUAL,
GL_GREATER, GL_NOTEQUAL, GL_GEQUAL,
                                          GL_ALWAYS};

std::vector<GLenum> blendTests = {GL_ZERO, GL_ONE, GL_DST_COLOR,
GL_ONE_MINUS_DST_COLOR,GL_SRC_COLOR, GL_ONE_MINUS_SRC_COLOR, GL_SRC_ALPHA,
                                    GL_ONE_MINUS_SRC_ALPHA, GL_DST_ALPHA,
GL_ONE_MINUS_DST_ALPHA,
                                  GL_SRC_ALPHA_SATURATE};

GLWidget::GLWidget(QWidget *parent)
    : QOpenGLWidget(parent) {
    indexOfFigure = 0;
}

GLWidget::~GLWidget() {}

void GLWidget::initializeGL() {
    initializeOpenGLFunctions();
    glClearColor(0,0,0,0);
}

void GLWidget::paintGL() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    start();
}

void GLWidget::start() {
    switch (filter) {
    case 1:
        glEnable(GL_ALPHA_TEST);
        opacityTest();
        draw(0.1, 0.1, indexOfFigure);
```

```
            glDisable(GL_ALPHA_TEST);
            break;
        case 2:
            glEnable(GL_BLEND);
            blendTest();
            draw(0.1, 0.1, indexOfFigure);
            glDisable(GL_BLEND);
            break;
        case 3:
            glEnable(GL_SCISSOR_TEST);
            scissorTest();
            draw(0.1, 0.1, indexOfFigure);
            glDisable(GL_SCISSOR_TEST);
            break;
        case 4:
            glEnable(GL_ALPHA_TEST);
            opacityTest();
            glEnable(GL_BLEND);
            blendTest();
            glEnable(GL_SCISSOR_TEST);
            scissorTest();
            draw(0.1, 0.1, indexOfFigure);
            glDisable(GL_SCISSOR_TEST);
            glDisable(GL_BLEND);
            glDisable(GL_ALPHA_TEST);
            break;
        default:
            draw(0.1, 0.1, indexOfFigure);
            break;
        }
}
void GLWidget::draw(float x, float y, int ind) {
    if (ind == arrOfFigures.size() - 1) {
        GLfloat theta;
        GLfloat pi     = acos(-1.0);
        GLfloat radius = 0.4f;
        GLfloat step   = 1.0f;

        glBegin(GL_TRIANGLE_FAN);
        for(GLfloat a = 0.0f; a < 360.0f; a += step) {
            theta = 2.0f * pi * a / 180.0f;
            glColor4f(a/360, 0, 0, a/360);
            glVertex3f(radius * cos(theta)-0.3, radius * sin(theta)-0.3, 0.0f);
        }
        glEnd();

        glBegin(GL_TRIANGLE_FAN);
        for(GLfloat a = 0.0f; a < 360.0f; a += step) {
            theta = 2.0f * pi * a / 180.0f;
            glColor4f(0, a/360, 0,  a/360);
            glVertex3f(radius * cos(theta), radius * sin(theta)+0.1, 0.0f);
        }
        glEnd();

        glBegin(GL_TRIANGLE_FAN);
```

```
        for(GLfloat a = 0.0f; a < 360.0f; a += step) {
            theta = 2.0f * pi * a / 180.0f;
            glColor4f(0, 0, a/360,  a/360);
            glVertex3f(radius * cos(theta)+0.3, radius * sin(theta)-0.3, 0.0f);
        }
        glEnd();
    } else {
        int n = 12;
        glPointSize(3);
        glBegin(arrOfFigures[ind]);
            for (int i = 0; i < n; i++) {
                float angle = 2 * 3.14 * i / (n);
                float x_ = (( -0.2 + cos(angle) * 0.8 + x));
                float y_ = (( -0.1 + sin(angle) * 0.8 + y));
                glColor3f((float)1/(i+1),(float)2/(i+1),(float)3/(i+1));
                glVertex2f(x_, y_);
            }
        glEnd();
    }
}

void GLWidget::setFilter(int par) {
    qDebug() << par;
    filter = par;
    update();
}
void GLWidget::setFigure(int par) {
    indexOfFigure = par;
    update();
}

void GLWidget::setAlphaIndex(int par) {
    alphaTestIndex = par;
    update();
}

void GLWidget::setAlphaValue(double par) {
    alphaTestValue = (double) par / 100;
    update();
}

void GLWidget::setBlendBegin(int par) {
    blendTestIndexBegin = par;
    update();
}

void GLWidget::setBlendEnd(int par) {
    blendTestIndexEnd = par;
    update();
}

void GLWidget::setX(double par) {
    scissorTestX = par;
    update();
```

```
}

void GLWidget::setY(double par) {
    scissorTestY = par;
    update();
}

void GLWidget::setW(double par) {
    scissorTestW = par;
    update();
}

void GLWidget::setH(double par) {
    scissorTestH = par;
    update();
}

void GLWidget::opacityTest() {
    glAlphaFunc(opacityTests[alphaTestIndex], alphaTestValue);
}

void GLWidget::blendTest() {
    glBlendFunc(blendTests[blendTestIndexBegin], blendTests[blendTestIndexEnd]);
}

void GLWidget::scissorTest() {
     glScissor(scissorTestX, scissorTestY, scissorTestW, scissorTestH);
}
```

mainWindow.cpp

#include "mainwindow.h"

```
#include "ui_mainwindow.h"
#include <QMessageBox>
#include <string>
#include <QDebug>

std::vector<std::string> arrOfFigures =
{"GL_POINT","GL_LINES","GL_LINE_STRIP","GL_LINE_LOOP","GL_TRIANGLES","GL_TRIANGLE_STR
IP",

"GL_TRIANGLE_FAN","GL_QUADS","GL_QUAD_STRIP","GL_POLYGON","CIRCLES"};
std::vector<std::string> typesOfTests = {"Выбирай тест", "Прозрачность",
"Смешивание", "Отсечение", "Все тесты"};
std::vector<std::string> opasityTests = { "GL_NEVER", "GL_LESS", "GL_EQUAL",
"GL_LEQUAL", "GL_GREATER", "GL_NOTEQUAL", "GL_GEQUAL",
                                          "GL_ALWAYS"};
```

```
std::vector<std::string> blendTests = {"GL_ZERO", "GL_ONE", "GL_DST_COLOR",
"GL_ONE_MINUS_DST_COLOR", "GL_SRC_COLOR", "GL_ONE_MINUS_SRC_COLOR", "GL_SRC_ALPHA",
                                       "GL_ONE_MINUS_SRC_ALPHA",
"GL_DST_ALPHA","GL_ONE_MINUS_DST_ALPHA", "GL_SRC_ALPHA_SATURATE"};

std::vector<int> startPositions = {0, 0, 0 , 550, 450};

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    for (std::string i : arrOfFigures) {
            ui->comboBox->addItem(QString::fromStdString(i));
    }
    QObject::connect(ui->comboBox, SIGNAL(activated(int)), ui->openGLWidget,
SLOT(setFigure(int)));

    for (std::string i : typesOfTests) {
        ui->comboBox_2->addItem(QString::fromStdString(i));
    }
    QObject::connect(ui->comboBox_2, SIGNAL(activated(int)), this,
SLOT(testAll(int)));
    for (std::string i : opasityTests) {
        ui->comboBox_3->addItem(QString::fromStdString(i));
    }
    QObject::connect(ui->comboBox_3, SIGNAL(activated(int)), ui->openGLWidget,
SLOT(setAlphaIndex(int)));

    QObject::connect(ui->horizontalSlider, SIGNAL(valueChanged(int)), this,
SLOT(dispOpacity(int)));

    for (std::string i : blendTests) {
        ui->comboBox_4->addItem(QString::fromStdString(i));
        ui->comboBox_5->addItem(QString::fromStdString(i));
    }
    QObject::connect(ui->comboBox_4, SIGNAL(activated(int)), ui->openGLWidget,
SLOT(setBlendBegin(int)));
    QObject::connect(ui->comboBox_5, SIGNAL(activated(int)), ui->openGLWidget,
SLOT(setBlendEnd(int)));

    QObject::connect(ui->horizontalSlider_2, SIGNAL(valueChanged(int)), this,
SLOT(dispX(int)));
    QObject::connect(ui->horizontalSlider_3, SIGNAL(valueChanged(int)), this,
SLOT(dispY(int)));
    QObject::connect(ui->horizontalSlider_4, SIGNAL(valueChanged(int)), this,
SLOT(dispW(int)));
    QObject::connect(ui->horizontalSlider_5, SIGNAL(valueChanged(int)), this,
SLOT(dispH(int)));
    testAll(0);
}

MainWindow::~MainWindow()
{
```

```
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
     QApplication::exit();
}

void MainWindow::dispOpacity(int par) {
    double p = (double) par;
    ui->label_13->setNum(p);
    ui->openGLWidget->setAlphaValue(p);
}

void MainWindow::dispX(int par) {
    double p = (double) par;
    ui->label_14->setNum(p);
    ui->openGLWidget->setX(p);
}

void MainWindow::dispY(int par) {
    double p = (double) par;
    ui->label_15->setNum(p);
    ui->openGLWidget->setY(p);
}

void MainWindow::dispW(int par) {
    double p = (double) par;
    ui->label_16->setNum(p);
    ui->openGLWidget->setW(p);
}

void MainWindow::dispH(int par) {
    double p = (double) par;
    ui->label_17->setNum(p);
    ui->openGLWidget->setH(p);
}

void MainWindow::turnAlphaTest() {
    qDebug() << "alphatest\n";
    ui->openGLWidget->setAlphaIndex(ui->comboBox_2->currentIndex());
    dispOpacity(ui->horizontalSlider->value());
}

void MainWindow::turnBlendTest() {
    qDebug() << "blendtest\n";
    ui->openGLWidget->setBlendBegin(ui->comboBox_3->currentIndex());
    ui->openGLWidget->setBlendEnd(ui->comboBox_4->currentIndex());
}

void MainWindow::turnScissorTest() {
    qDebug() << "scissortest\n";
    dispX(ui->horizontalSlider_2->value());
    dispY(ui->horizontalSlider_3->value());
```

```
    dispW(ui->horizontalSlider_4->value());
    dispH(ui->horizontalSlider_5->value());

}

void MainWindow::testAll(int val) {
    qDebug() << val ;
    switch(val) {
    case 1:
        ui->openGLWidget->setFilter(1);
        turnAlphaTest();

        ui->label_2->setEnabled(true);
        ui->label_13->setEnabled(true);
        ui->comboBox_3->setEnabled(true);
        ui->horizontalSlider->setEnabled(true);

        ui->label_5->setEnabled(false);
        ui->label_6->setEnabled(false);
        ui->label_7->setEnabled(false);
        ui->comboBox_4->setEnabled(false);
        ui->comboBox_5->setEnabled(false);

        ui->label_8->setEnabled(false);
        ui->label_9->setEnabled(false);
        ui->label_10->setEnabled(false);
        ui->label_11->setEnabled(false);
        ui->label_12->setEnabled(false);
        ui->label_14->setEnabled(false);
        ui->label_15->setEnabled(false);
        ui->label_16->setEnabled(false);
        ui->label_17->setEnabled(false);
        ui->horizontalSlider_2->setEnabled(false);
        ui->horizontalSlider_3->setEnabled(false);
        ui->horizontalSlider_4->setEnabled(false);
        ui->horizontalSlider_5->setEnabled(false);
        break;
    case 2:
        ui->openGLWidget->setFilter(2);
        turnBlendTest();

        ui->label_2->setEnabled(false);
        ui->label_13->setEnabled(false);
        ui->comboBox_3->setEnabled(false);
        ui->horizontalSlider->setEnabled(false);

        ui->label_5->setEnabled(true);
        ui->label_6->setEnabled(true);
        ui->label_7->setEnabled(true);
        ui->comboBox_4->setEnabled(true);
        ui->comboBox_5->setEnabled(true);

        ui->label_8->setEnabled(false);
        ui->label_9->setEnabled(false);
```

```
    ui->label_10->setEnabled(false);
    ui->label_11->setEnabled(false);
    ui->label_12->setEnabled(false);
    ui->label_14->setEnabled(false);
    ui->label_15->setEnabled(false);
    ui->label_16->setEnabled(false);
    ui->label_17->setEnabled(false);
    ui->horizontalSlider_2->setEnabled(false);
    ui->horizontalSlider_3->setEnabled(false);
    ui->horizontalSlider_4->setEnabled(false);
    ui->horizontalSlider_5->setEnabled(false);
    break;
case 3:
    ui->openGLWidget->setFilter(3);
    turnScissorTest();

    ui->label_2->setEnabled(false);
    ui->label_13->setEnabled(false);
    ui->comboBox_3->setEnabled(false);
    ui->horizontalSlider->setEnabled(false);

    ui->label_5->setEnabled(false);
    ui->label_6->setEnabled(false);
    ui->label_7->setEnabled(false);
    ui->comboBox_4->setEnabled(false);
    ui->comboBox_5->setEnabled(false);

    ui->label_8->setEnabled(true);
    ui->label_9->setEnabled(true);
    ui->label_10->setEnabled(true);
    ui->label_11->setEnabled(true);
    ui->label_12->setEnabled(true);
    ui->label_14->setEnabled(true);
    ui->label_15->setEnabled(true);
    ui->label_16->setEnabled(true);
    ui->label_17->setEnabled(true);
    ui->horizontalSlider_2->setEnabled(true);
    ui->horizontalSlider_3->setEnabled(true);
    ui->horizontalSlider_4->setEnabled(true);
    ui->horizontalSlider_5->setEnabled(true);
    break;
case 4:
    ui->openGLWidget->setFilter(4);
    turnAlphaTest();
    turnBlendTest();
    turnScissorTest();

    ui->label_2->setEnabled(true);
    ui->label_13->setEnabled(true);
    ui->comboBox_3->setEnabled(true);
    ui->horizontalSlider->setEnabled(true);

    ui->label_5->setEnabled(true);
    ui->label_6->setEnabled(true);
    ui->label_7->setEnabled(true);
```

```
        ui->comboBox_4->setEnabled(true);
        ui->comboBox_5->setEnabled(true);

        ui->label_8->setEnabled(true);
        ui->label_9->setEnabled(true);
        ui->label_10->setEnabled(true);
        ui->label_11->setEnabled(true);
        ui->label_12->setEnabled(true);
        ui->label_14->setEnabled(true);
        ui->label_15->setEnabled(true);
        ui->label_16->setEnabled(true);
        ui->label_17->setEnabled(true);
        ui->horizontalSlider_2->setEnabled(true);
        ui->horizontalSlider_3->setEnabled(true);
        ui->horizontalSlider_4->setEnabled(true);
        ui->horizontalSlider_5->setEnabled(true);
        break;

    default:
        ui->openGLWidget->setFilter(0);

        ui->label_2->setEnabled(false);
        ui->label_13->setEnabled(false);
        ui->comboBox_3->setEnabled(false);
        ui->horizontalSlider->setEnabled(false);

        ui->label_5->setEnabled(false);
        ui->label_6->setEnabled(false);
        ui->label_7->setEnabled(false);
        ui->comboBox_4->setEnabled(false);
        ui->comboBox_5->setEnabled(false);

        ui->label_8->setEnabled(false);
        ui->label_9->setEnabled(false);
        ui->label_10->setEnabled(false);
        ui->label_11->setEnabled(false);
        ui->label_12->setEnabled(false);
        ui->label_14->setEnabled(false);
        ui->label_15->setEnabled(false);
        ui->label_16->setEnabled(false);
        ui->label_17->setEnabled(false);
        ui->horizontalSlider_2->setEnabled(false);
        ui->horizontalSlider_3->setEnabled(false);
        ui->horizontalSlider_4->setEnabled(false);
        ui->horizontalSlider_5->setEnabled(false);
        break;
    }
}
```

main.cpp

#include "mainwindow.h"

```
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

glwidget.h

#ifndef GLWIDGET_H

```
#define GLWIDGET_H
#include <GL/freeglut.h>
#include <QWidget>
#include <QOpenGLWidget>
#include <QOpenGLFunctions>

class GLWidget : public QOpenGLWidget, protected QOpenGLFunctions
{
    Q_OBJECT
public:
    GLWidget(QWidget *parent = 0);
    ~GLWidget();
    void initializeGL();
    void paintGL() override;
    void draw(float, float, int);
    void opacityTest();
    void blendTest();
    void scissorTest();
    void start();

    int filter = 0;
    int indexOfFigure;
    int alphaTestIndex = 0;
    double alphaTestValue = 0;
    int blendTestIndexBegin = 0;
    int blendTestIndexEnd = 0;
    double scissorTestX = 0;
    double scissorTestY = 0;
    double scissorTestW = 550;
    double scissorTestH = 450;

public slots:
    void setFilter(int);
    void setFigure(int);
```

```
    void setAlphaIndex(int);
    void setAlphaValue(double);
    void setBlendBegin(int);
    void setBlendEnd(int);
    void setX(double);
    void setY(double);
    void setW(double);
    void setH(double);

};

#endif // GLWIDGET_H
```

mainWindow.h

#ifndef MAINWINDOW_H

```
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtGui>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  explicit MainWindow(QWidget *parent = 0);
  ~MainWindow();

protected slots:
  void testAll(int);
  void turnAlphaTest();
  void turnBlendTest();
  void turnScissorTest();

  void dispOpacity(int);
  void dispX(int);
  void dispY(int);
  void dispW(int);
  void dispH(int);

private slots:
    void on_pushButton_clicked();
```

```
public:
  Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```