*Горшков Алексей Олегович*

*Группа 4212*

**Отчет о выполнении лабораторной работы №2 «Преобразование модели и координат»**

**Задание №1**

1. **Задание:**

*Используя операции преобразования модели, создайте программу*:

(Вариант 3): Прямоугольник, произвольно заданный в МСК, зеркально отображается относительно произвольно заданной прямой и пульсирует.

*Все размерные величины подберите так, чтобы на экране хорошо просматривалось изображение и его особенности.*

1. **Листинг программы:**

#include <GL/glut.h>

#include <stdlib.h>

#include <windows.h> //Для sleep()

GLfloat R{ 640.0 / 480.0 }; //Форматное соотношение.

GLfloat w{40}; //Ширина мирового окна.

GLfloat h{}; //Высота мирового окна.

GLfloat l{}, r{}, b{}, t{}; //Параметры мирового окна.

GLfloat pulsing{1.0}; //Переменная для пульсации (масштабирование).

bool flag{ false }; //Переменная для пульсирования в ту или иную сторону.

bool rectDivision{ false }; //Переменная для разделения прямоугольников первоначального и масштабированного (отображаемого) (рисуются одной и той же процедурой).

void init(void)

{

//h = 30, l = - 20, r = 20, b = 15, t = 15.

h = w / R; l = -w / 2; r = w / 2; b = -h / 2; t = h / 2; //Расчет параметров мирового окна.

glClearColor(0.5, 0.5, 0.71, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(l, r, b, t);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void reshape(GLsizei W, GLsizei H)//Сохранение масштаба во время работы программы.

{

if (R > static\_cast<float>(W) / H) glViewport(0, 0, W, W / R);//0, 0, 640, 480.

else glViewport(0, 0, H \* R, H);//0, 0, 640, 480.

}

void showAxis(void)//Прорисовка осей МСК (СК0) первоначального прямоугольника.

{

glColor3f(0.0f, 0.0f, 1.0f);

glBegin(GL\_LINES);

glVertex2f(0, 0);

glVertex2f(0, t);

glVertex2f(0, 0);

glVertex2f(r, 0);

glEnd();

}

void showAxisFig0Scaled(void) //Прорисовка осей СК (СКL) прямоугольника, отображенного относительно произвольно заданной прямой.

{

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_LINES);

glVertex2f(0, 0);

glVertex2f(0, t / pulsing);

glVertex2f(0, 0);

glVertex2f(r / pulsing, 0);

glEnd();

}

void fig0(void)//Прорисовка прямоугольников.

{

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_LOOP);

if (!rectDivision)//Прорисовка первоначального прямоугольника.

{

glVertex2f(2.0, 3.0);

glVertex2f(2.0, 9.0);

glVertex2f(6.0, 9.0);

glVertex2f(6.0, 3.0);

}

else//Прорисовка масштабируемого (пульсирующего, отображаемого) прямоугольника в зависимости от масштаба с соответствующими координатами.

{

if (pulsing <= 1.0)

{

glVertex2f(2.0 / pulsing, 3.0 / pulsing);

glVertex2f(2.0 / pulsing, 9.0 / pulsing);

glVertex2f(6.0 / pulsing, 9.0 / pulsing);

glVertex2f(6.0 / pulsing, 3.0 / pulsing);

}

else if ((pulsing >= 1.2) && (pulsing <= 1.4))

{

glVertex2f(1.6 / pulsing, 2.4 / pulsing);

glVertex2f(1.6 / pulsing, 9.6 / pulsing);

glVertex2f(6.4 / pulsing, 9.6 / pulsing);

glVertex2f(6.4 / pulsing, 2.4 / pulsing);

}

else

{

glVertex2f(1.2 / pulsing, 1.8 / pulsing);

glVertex2f(1.2 / pulsing, 10.2 / pulsing);

glVertex2f(6.8 / pulsing, 10.2 / pulsing);

glVertex2f(6.8 / pulsing, 1.8 / pulsing);

}

}

glEnd();

}

void line(void)//Прорисовка линии, относительно которой отображается первоначальный прямоугольник.

{

glColor3f(0.0, 0.0, 0.0);

glLineWidth(2.0);

glBegin(GL\_LINES);

glVertex2f(-14, 14);

glVertex2f(14, -14);

glEnd();

}

void scene(void)//Сцена, прорисовка фигур с соответственными преобразованиями СК (МСК(СК0) -> СКL).

{

glClear(GL\_COLOR\_BUFFER\_BIT);

showAxis();

rectDivision = false;

fig0();

line();

glPushMatrix();

glRotatef(180, 0.0, 0.0, 1.0);

glScalef(pulsing, pulsing, 0.0);

showAxisFig0Scaled();

rectDivision = true;

fig0();

glPopMatrix();

glFlush();

glutSwapBuffers();

if (!flag)

{

pulsing += 0.2;

if (pulsing >= 1.4)

flag = true;

}

else

{

pulsing -= 0.2;

if (pulsing <= 1.0)

flag = false;

}

Sleep(300);

}

void main(int argc, char\*\* argv)//Главная функция.

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(20, 20);

glutCreateWindow("Lab2\_Prog1");

glutReshapeFunc(reshape);

glutDisplayFunc(scene);

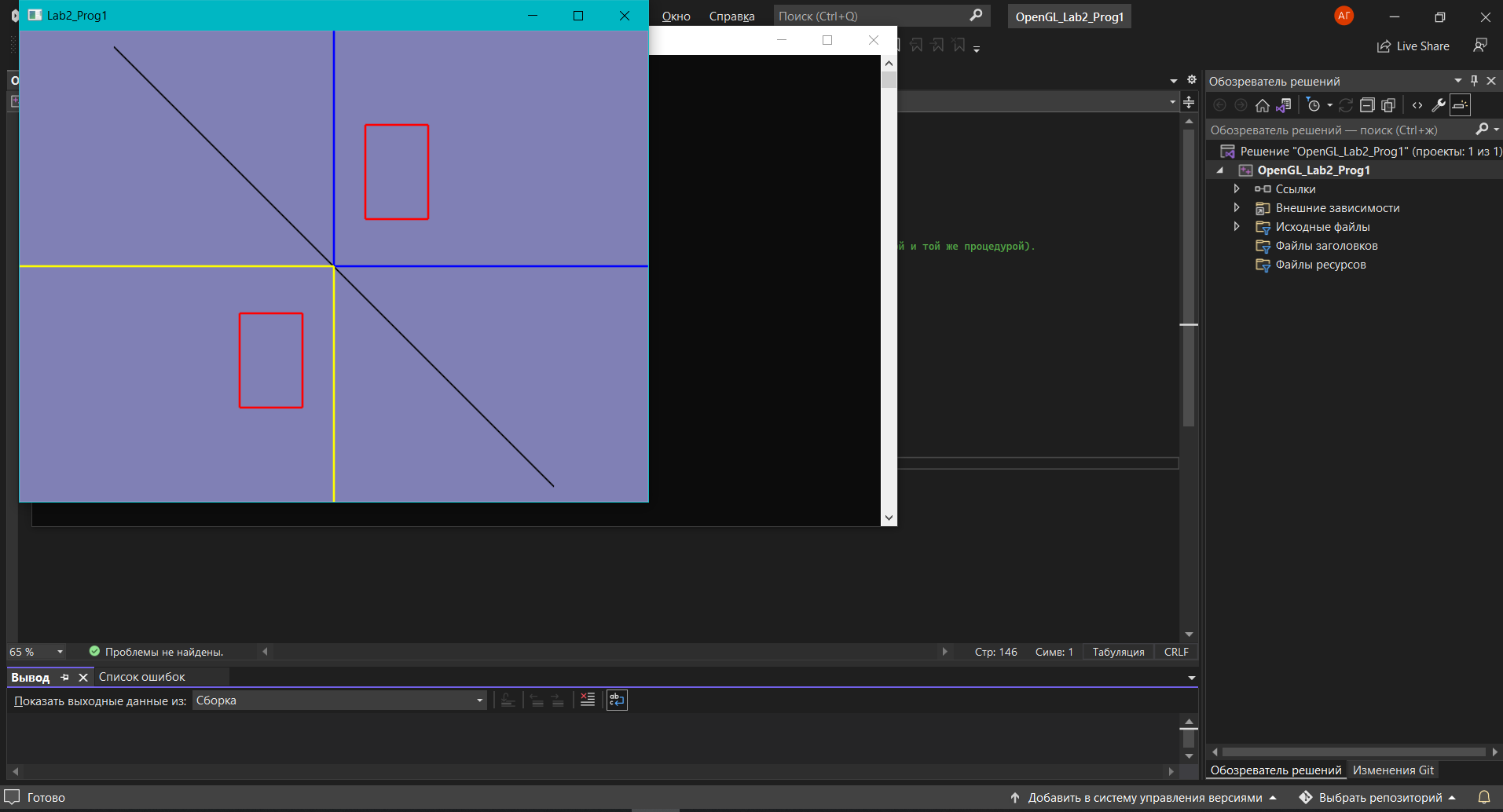
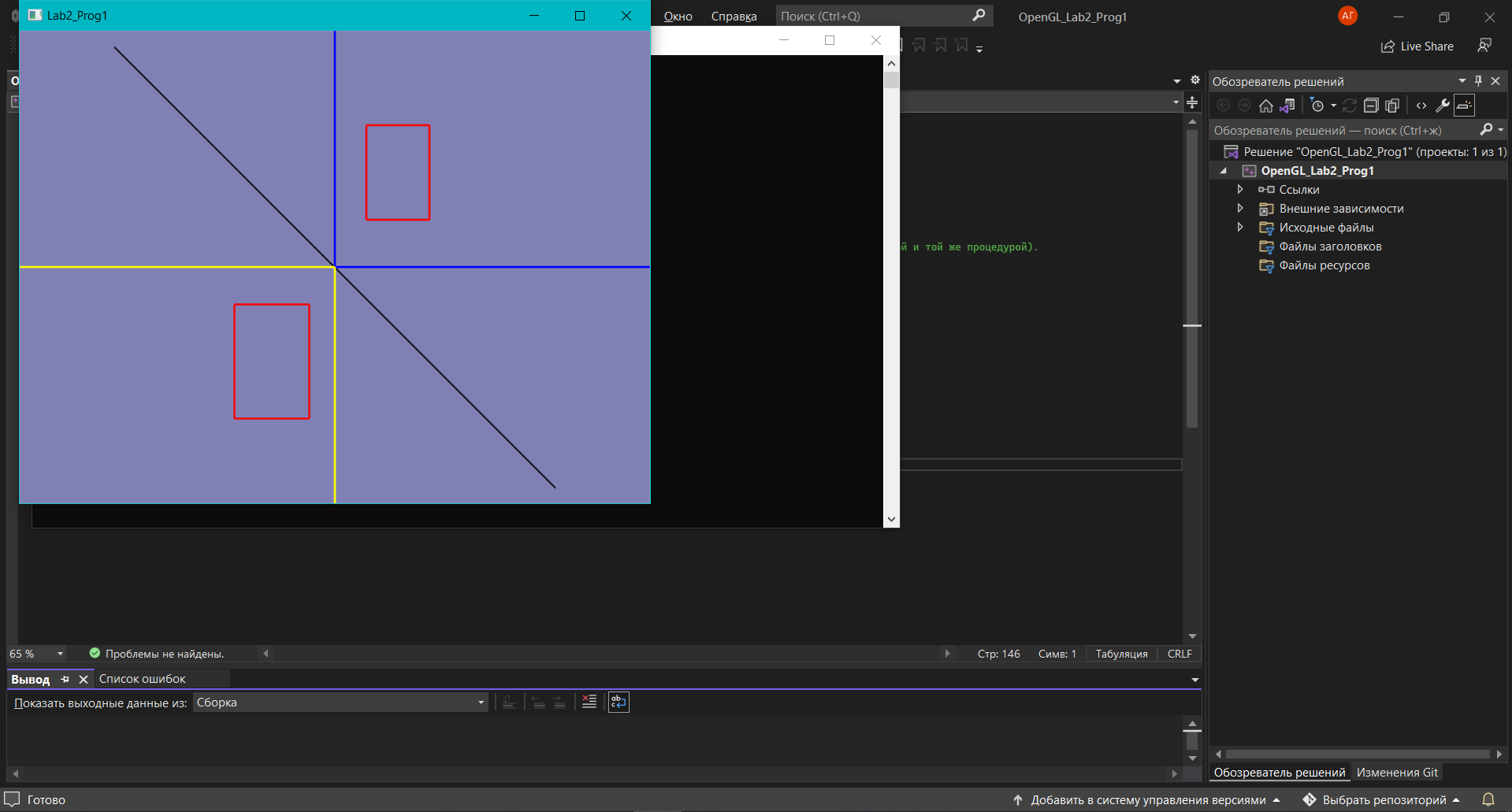
glutIdleFunc(scene);

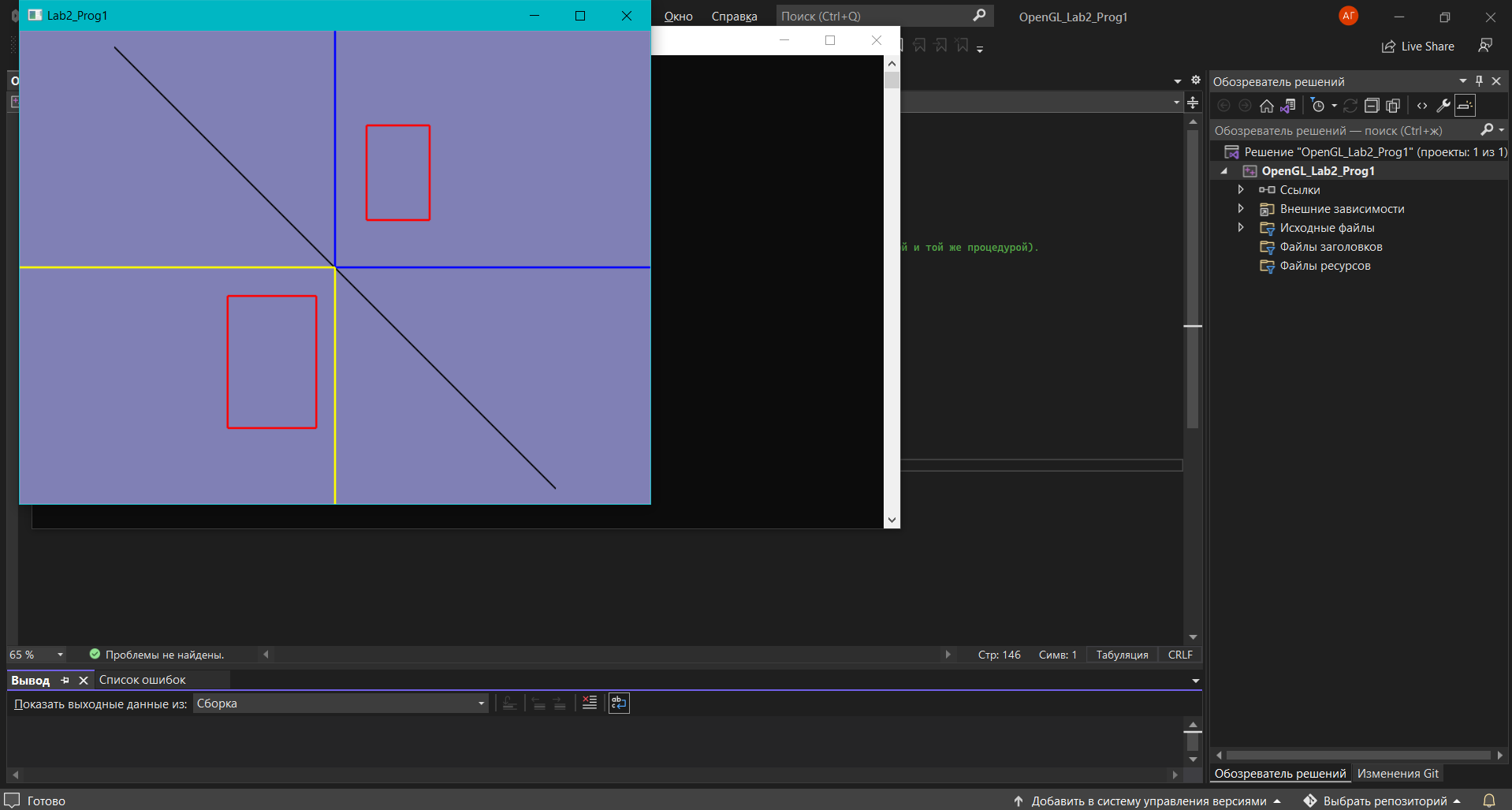
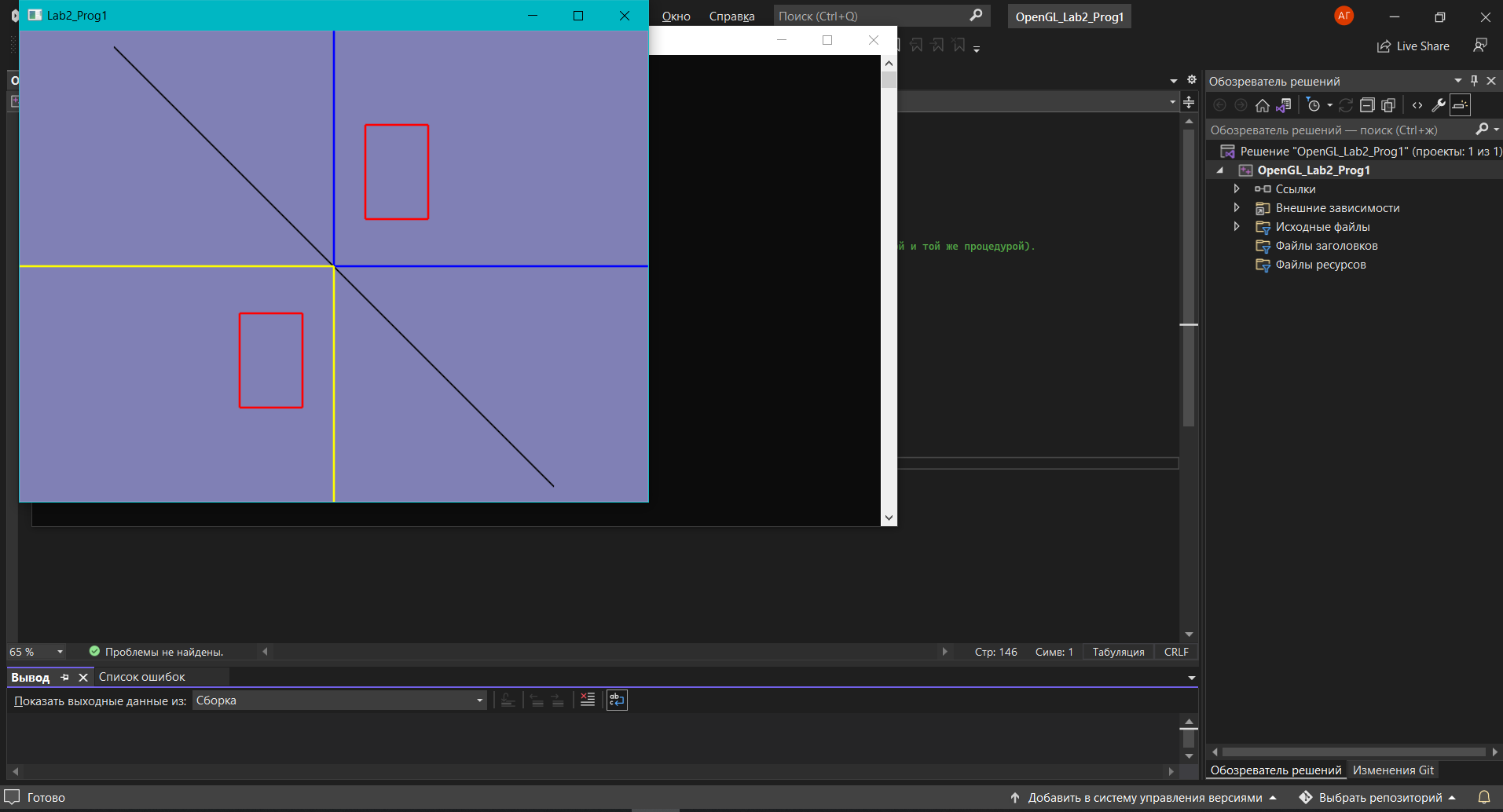
init();

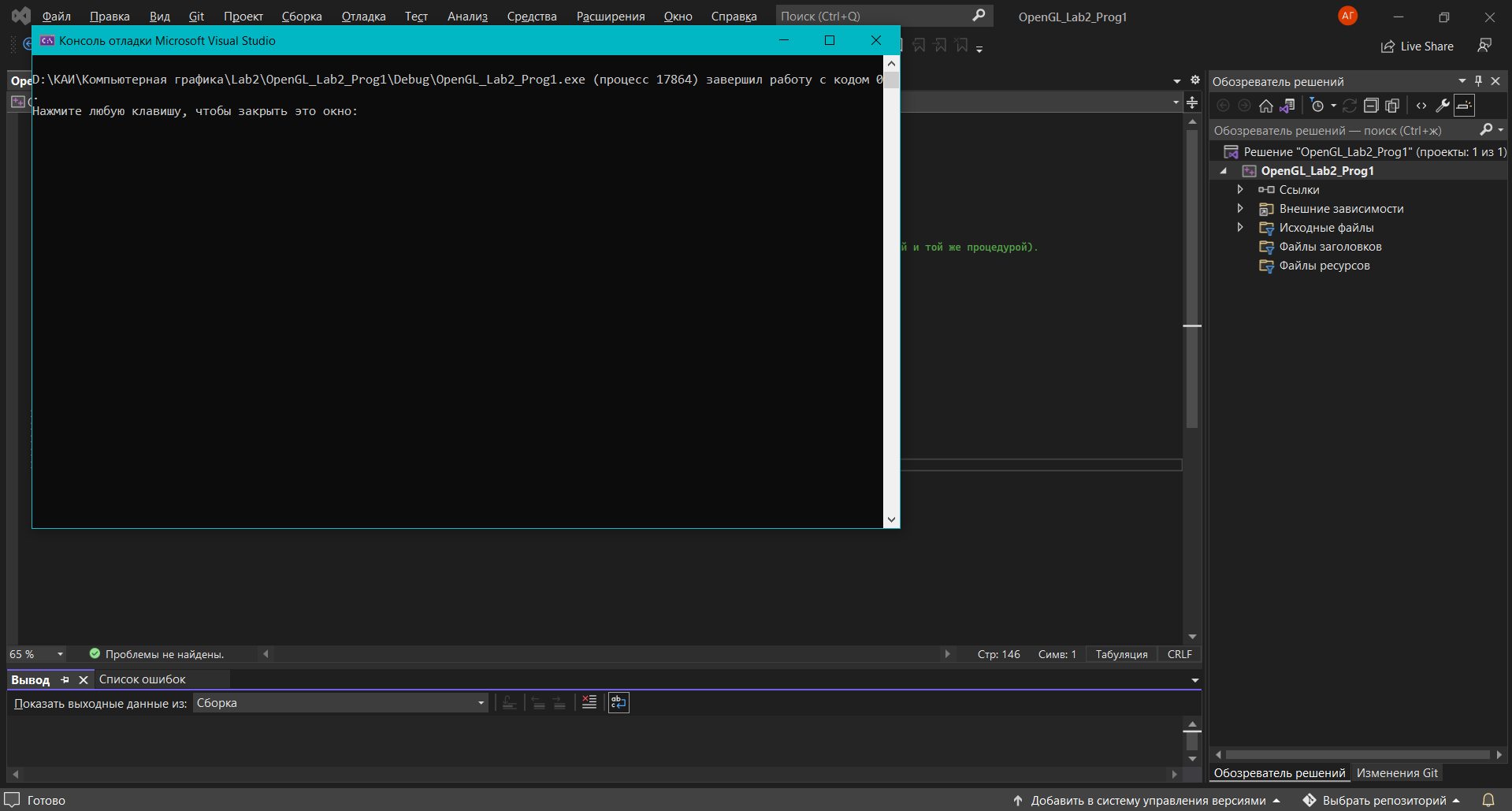
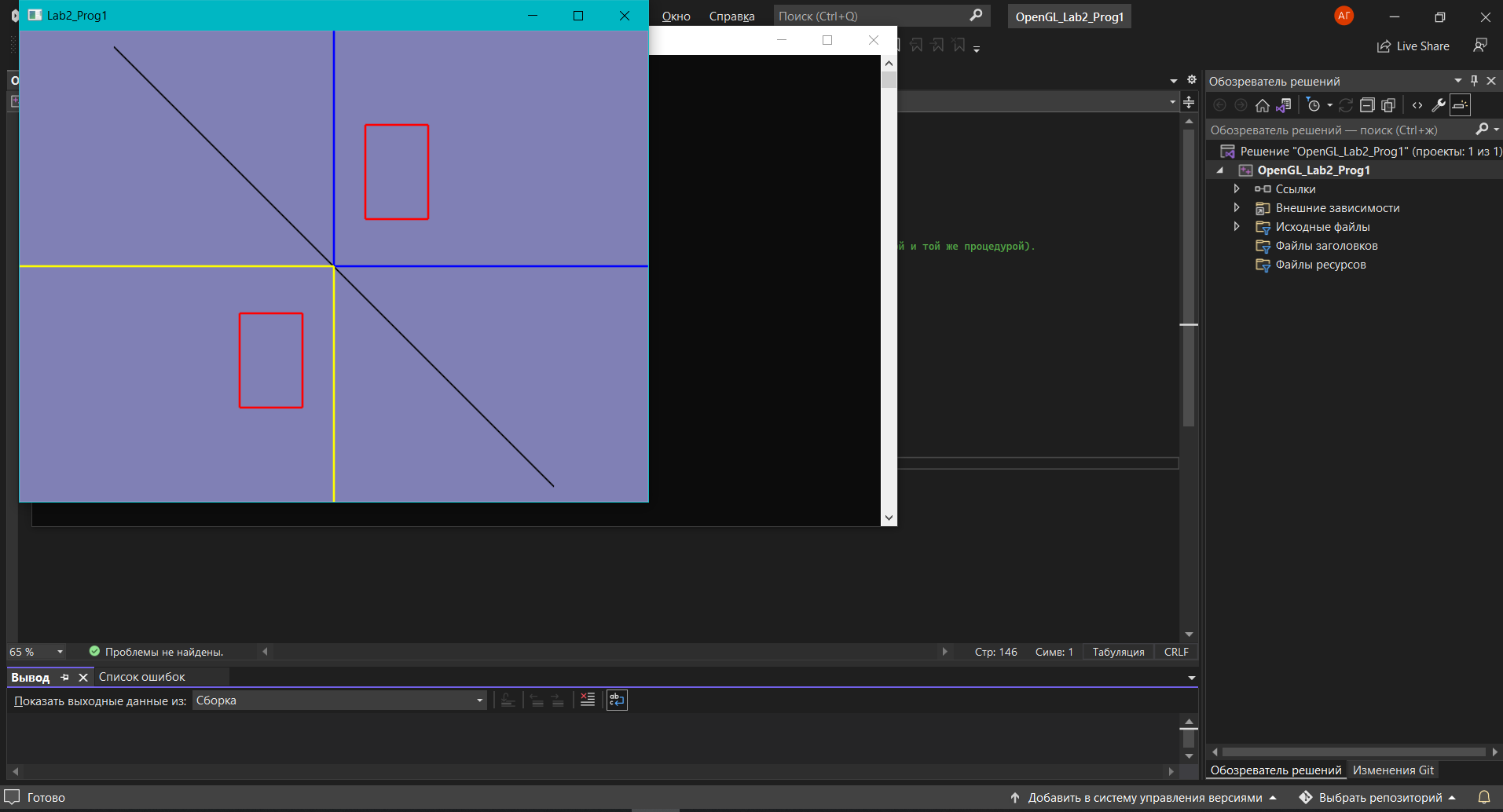
glutMainLoop();

}

1. **Работа программы:**





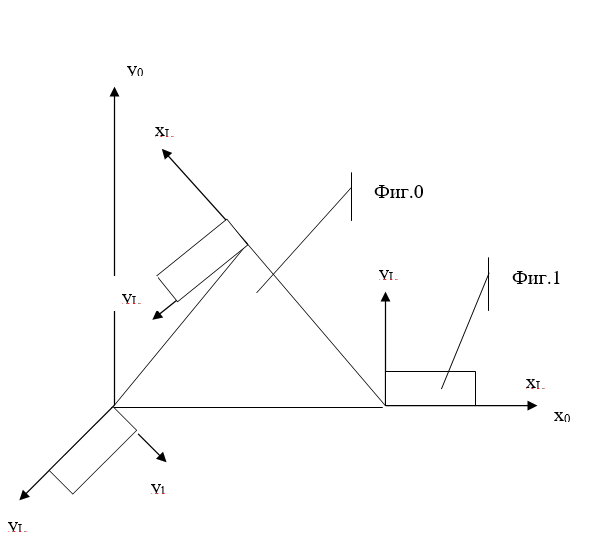
**Задание №2**

1. **Задание:**

*Для вариантов, приведенных ниже, создайте программу, которая рисует сцену, состоящую из фиг.0 и нескольких фиг.1. Для фиг.1 создайте отдельную функцию, описывающую ее в СК0, и затем перемещайте фиг.1 из позиции в позицию. В отчете для каждой позиции фиг.1 выпишите последовательность движений фиг.1, в результате которых она попадает в заданную позицию. Кроме того, считайте, что функция для фиг.1 описывает ее в локальной системе координат СКL, а полученные преобразования понимайте как преобразования координат из СКL в СК0. Для каждой позиции фиг1. изобразите цепочку преобразований систем координат и все координатные фреймы как это сделано на рис.2.*

*Примечание. Fig.0 и Fig.1 во всех позициях должны быть нарисованы с положительными координатными полуосями (стрелки не рисовать).*

*(Вариант 3):*



1. **Листинг программы:**

#include <GL/glut.h>

#include <stdlib.h>

#include <Windows.h>

GLfloat R = 640.0 / 480.0; //Форматное соотношение.

GLfloat w = 40; //Ширина мирового окна.

GLfloat h; //Высота мирового окна.

GLfloat l, r, b, t; //Параметры мирового окна.

GLfloat f{}; //Угол поворота фиг.1.

GLfloat scale{ 3.0 / 8.0 }; //Масштабирование прямоугольника (фиг. 1) и его осей СК (СКL) для случая, когда f = 120;

bool flag{ false }; //Для учета scale в размерах осей СК (СКL) фиг. 1.

void init(void)

{

//h = 30, l = - 30 10/13, r = 30 10/13, b = -23 1/13, t = 23 1/13.

h = w / R; l = -w / 1.3; r = w / 1.3; b = -h / 1.3; t = h / 1.3; //Расчет параметров мирового окна.

glClearColor(0.5, 0.5, 0.71, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(l, r, b, t);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void reshape(GLsizei W, GLsizei H)//Сохранение масштаба во время работы программы.

{

if (R > static\_cast<float>(W) / H) glViewport(0, 0, W, W / R);//0, 0, 640, 480.

else glViewport(0, 0, H \* R, H);//0, 0, 640, 480.

}

void showAxis(void)//Прорисовка осей СК (МСК(СК0)), в частности, для треугольника (фиг.0).

{

glColor3f(0.0f, 0.0f, 1.0f);

glBegin(GL\_LINES);

glVertex2f(0, 0);

glVertex2f(0, t);

glVertex2f(0, 0);

glVertex2f(r, 0);

glEnd();

}

void showAxisMovingFig1(void)//Прорисовка осей СК (СКL) смещающегося прямоугольника (фиг. 1).

{

glColor3f(1.0f, 0.7f, 0.0f);

glBegin(GL\_LINES);

if (!flag)

{

glVertex2f(0, 0);

glVertex2f(0, 10);

glVertex2f(0, 0);

glVertex2f(12, 0);

}

else

{

glVertex2f(0 / scale, 0 \* scale);

glVertex2f(0 / scale, 10 \* scale);

glVertex2f(0 / scale, 0 \* scale);

glVertex2f(12 / scale, 0 \* scale);

}

glEnd();

}

void fig0(void)//Прорисовка треугольника (фиг. 0).

{

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(0.0, 0.0);

glVertex2f(20.0, 0.0);

glVertex2f(10.0, 10.0);

glEnd();

}

void fig1(void)//Прорисовка смещающегося прямоугольника (фиг. 1).

{

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(0.0, 0.0);

glVertex2f(8.0, 0.0);

glVertex2f(8.0, 3.0);

glVertex2f(0.0, 3.0);

glEnd();

}

void scene(void)//Сцена, прорисовка фигур с соответствующим смещением и преобразованием СК МСК(СК0) -> СКL.

{

glClear(GL\_COLOR\_BUFFER\_BIT);

showAxis();

fig0();

glPushMatrix();

if (f == 0) glTranslatef(20.0, 0.0, 0.0);

else if (f == 120) glTranslatef(10.0, 10.0, 0.0);

glRotatef(f, 0.0, 0.0, 1.0);

if (f == 120)

{

glScalef(scale, 1.0 / scale, 1.0);

flag = true;

}

else

{

glScalef(1.0, 1.0, 1.0);

flag = false;

}

showAxisMovingFig1();

fig1();

glPopMatrix( );

glFlush();

glutSwapBuffers();

f += 120;

if (f == 360)

f = 0;

Sleep(400);

}

void main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(20, 20);

glutCreateWindow("Lab2\_Prog2");

glutReshapeFunc(reshape);

glutDisplayFunc(scene);

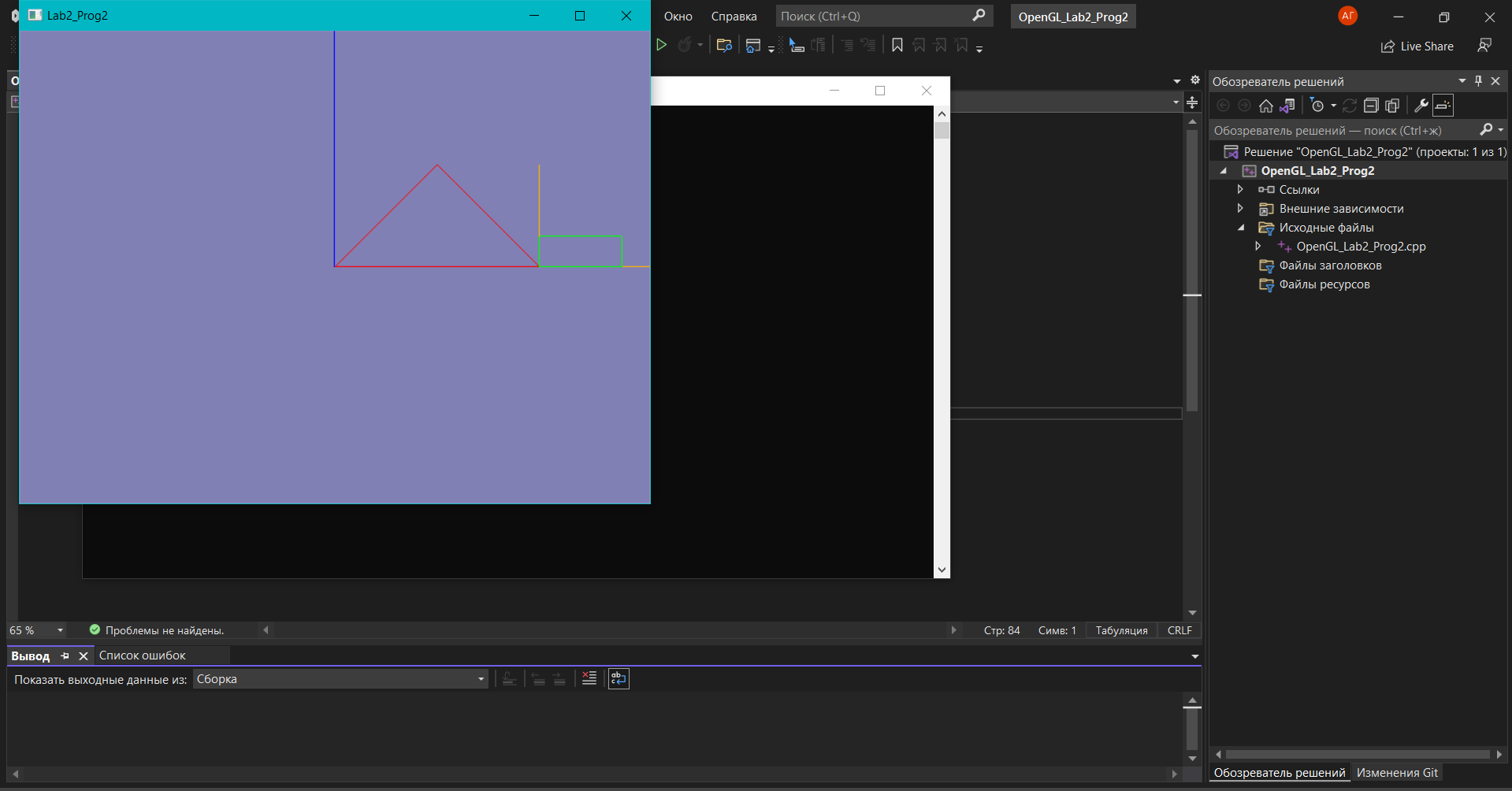
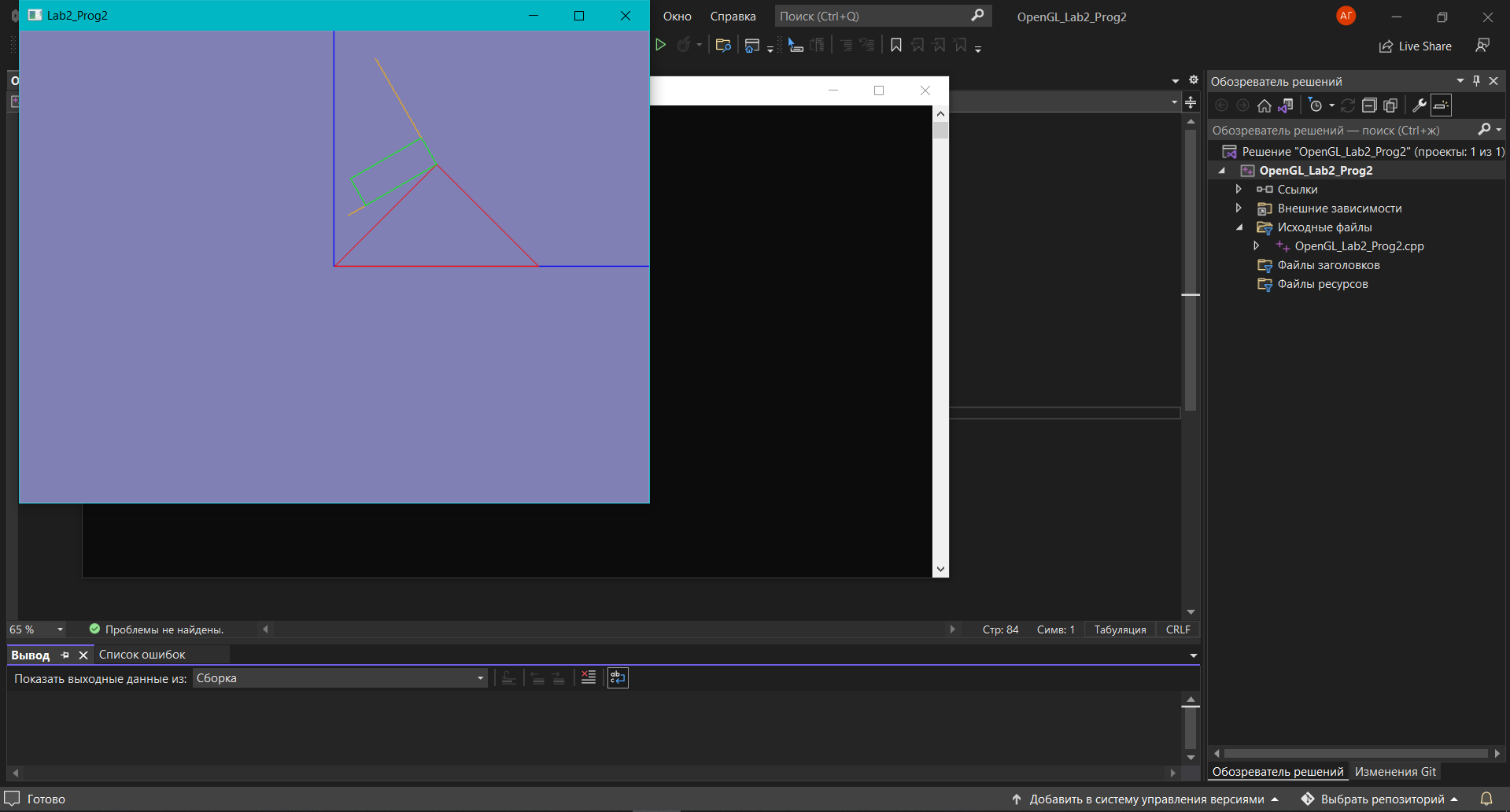
glutIdleFunc(scene);

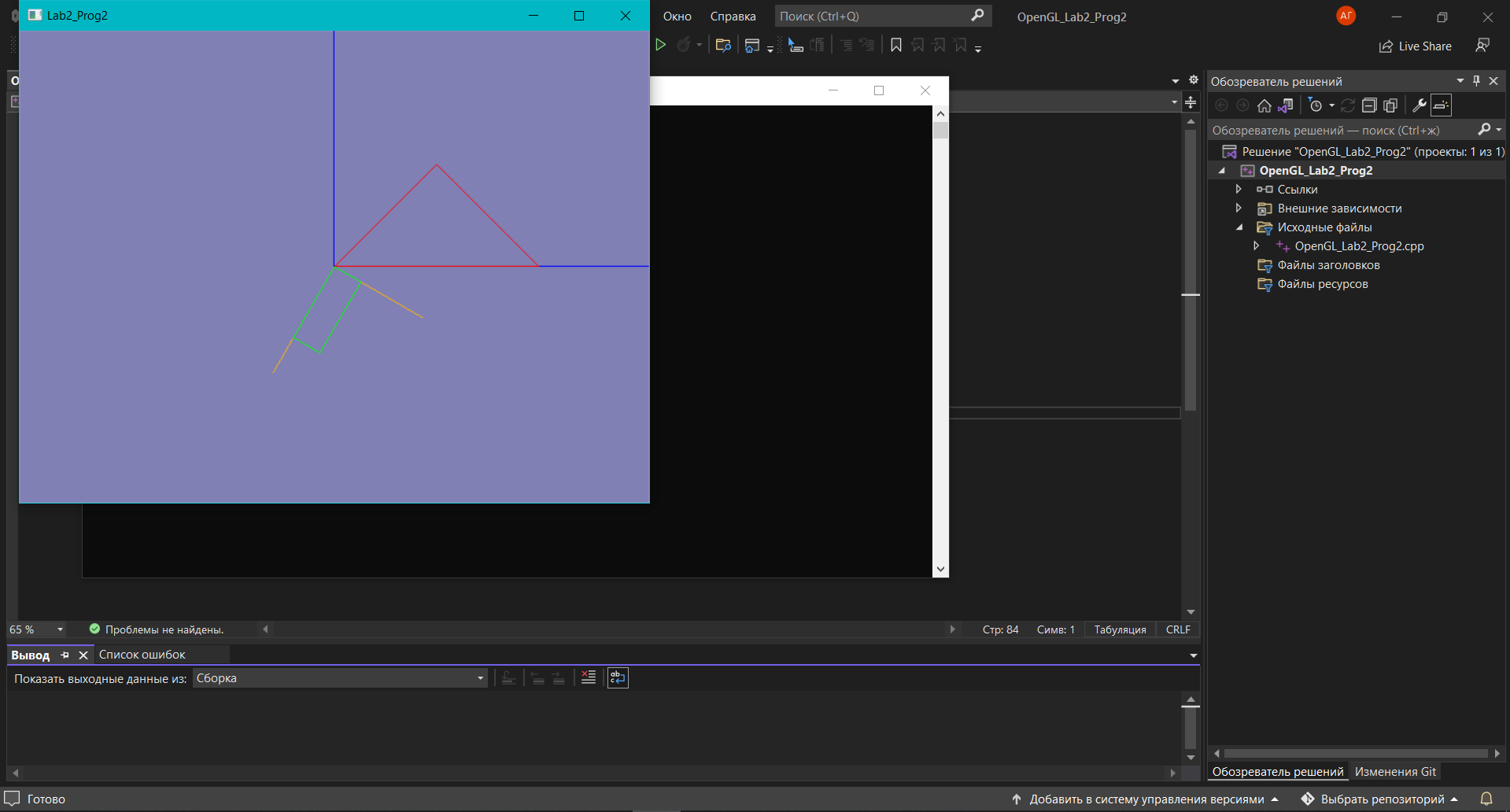
init();

glutMainLoop();

}

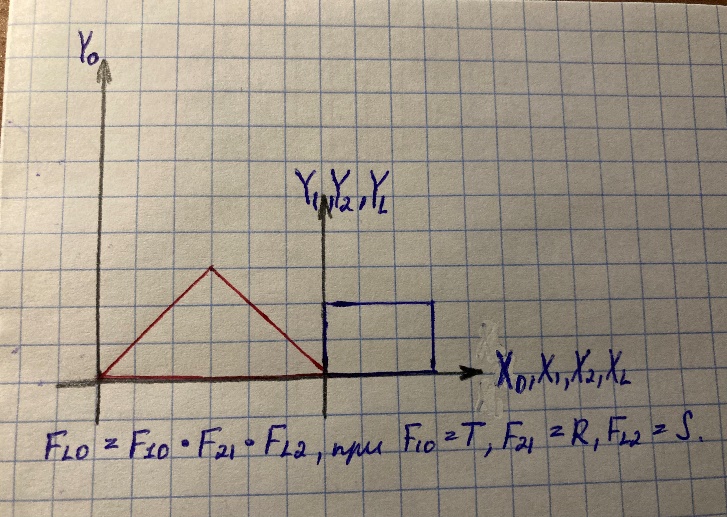
1. **Работа программы:**

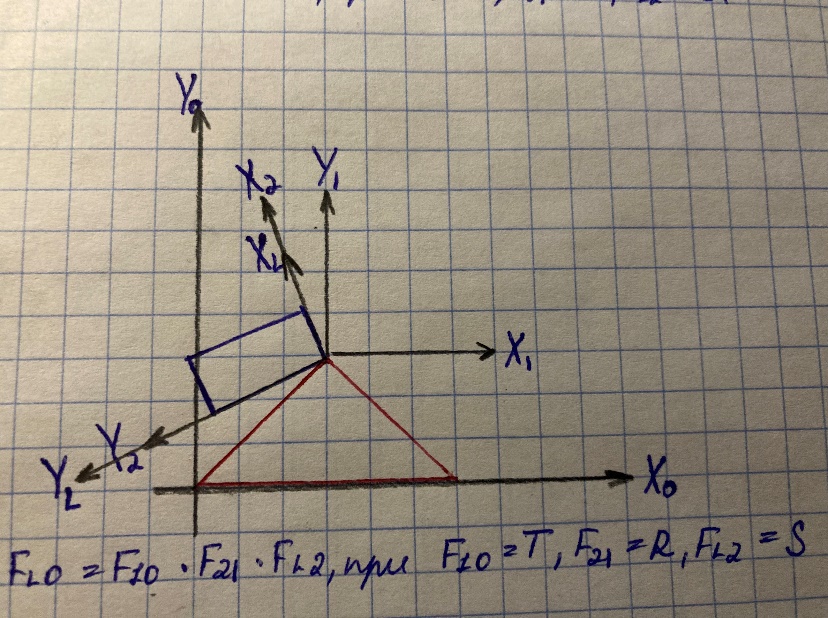


***Примечание (цепочка преобразований систем координат):***

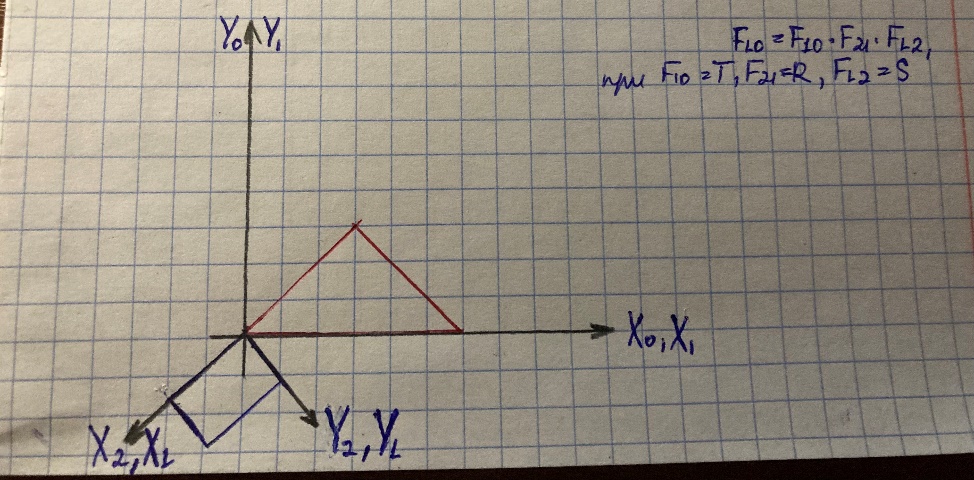
* *Крайнее правое положение имеет след. цепочку (угол поворота = 0):*



* *Крайнее верхнее положение имеет след. цепочку (угол поворота = 120):*



* *Крайнее левое положение имеет след. цепочку (угол поворота = 240):*



*На этих рисунках показаны локальные СКL и все промежуточные в соответствии с преобразованием FL0 =F10\*F21\*FL2, при F10 = T, F21=R, FL2=S. Причем это равносильно такой последовательности преобразований над фигурой: вращение –> перемещение -> масштабирование, хотя вызов функций в программе обратен данной последовательности. Все фигуры (фиг. 1) перемещаются за счет преобразований над системой координат (СК), т.е. перехода от мировой (общей) СК (МСК (CК0)) к локальной СК фиг. 1 (СКL) по указанной выше последовательности.*