1. Задание

Включить 3 источника света, задать отражающие свойства поверхностей, положение и цвет источников света. Рекомендуется цвет поверхностей сделать одинаковым, а отражающие свойства – разными. Вывести несколько объемных объектов. Источники света должны иметь различный цвет. Необходимо организовать вращение сцены, управляемое с клавиатуры (сцена с объектами объекты вращается, источники неподвижны)

2. Текст программы

```
#include <glut.h>

GLfloat  rx = 0;          // Угол поворта сцены вокруг оси X
GLfloat  ry = 0;          // Угол поворта сцены вокруг оси Y

GLfloat mat_amb_diff[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 50.0 };

GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light_position[] = { 1.0, 0.0, 0.0, 0.0 };

void init(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glShadeModel(GL_SMOOTH);

   static float lmodel_twoside[] = { GL_TRUE };
   static float lmodel_oneside[] = { GL_FALSE };
```

```
static float ambient[] = { 0.2, 0.2, 0.2, 1.0 };
static float diffuse[] = { 1.0f, 0.0, 0.0, 1.0 };
static float position[] = { -2.0, 1.0, 1.0f, 0.0 };

static float ambient1[] = { 0.3, 0.3, 0.3, 1.0 };
static float diffuse1[] = { 0.0, 1.0, 0.0, 1.0 };
static float position1[] = { 1.0,1.0, 1.0f, 0.0 };

static float ambient2[] = { 0.4, 0.4, 0.4, 1.0 };
static float diffuse2[] = { 0.0, 0.0, 1.0, 1.0 };
static float position2[] = { 2.0,1.0, 1.0f, 0.0 };


static float front_mat_shininess[] =
{ 60.0 };
static float front_mat_specular[] =
{ 0.2, 0.2, 0.2, 1.0 };
static float front_mat_diffuse[] =
{ 0.5, 0.5, 0.28, 1.0 };
static float back_mat_shininess[] =
{ 60.0 };
static float back_mat_specular[] =
{ 0.5, 0.5, 0.2, 1.0 };
static float back_mat_diffuse[] =
{ 1.0, 0.9, 0.2, 1.0 };

static float lmodel_ambient[] =
{ 1.0, 1.0, 1.0, 1.0 };
```

```
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);

glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
glLightfv(GL_LIGHT0, GL_POSITION, position);

glLightfv(GL_LIGHT1, GL_AMBIENT, ambient1);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuse1);
glLightfv(GL_LIGHT1, GL_POSITION, position1);

glLightfv(GL_LIGHT2, GL_AMBIENT, ambient2);
glLightfv(GL_LIGHT2, GL_DIFFUSE, diffuse2);
glLightfv(GL_LIGHT2, GL_POSITION, position2);

glMaterialfv(GL_FRONT, GL_SHININESS, front_mat_shininess);
glMaterialfv(GL_FRONT, GL_SPECULAR, front_mat_specular);
glMaterialfv(GL_FRONT, GL_DIFFUSE, front_mat_diffuse);
glMaterialfv(GL_BACK, GL_SHININESS, back_mat_shininess);
glMaterialfv(GL_BACK, GL_SPECULAR, back_mat_specular);
glMaterialfv(GL_BACK, GL_DIFFUSE, back_mat_diffuse);

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
glLightModelfv(GL_LIGHT_MODEL_TWO_SIDE, lmodel_twoside);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
```

```
    glEnable(GL_LIGHT2);
    glShadeModel(GL_SMOOTH);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glDisable(GL_LIGHTING);
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHT1);
    glDisable(GL_LIGHT2);



    glColor3f(1.0f, 0.0f, 0.0f);
    glBegin(GL_LINES);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0f, 1.0f, 0.0f);
    glBegin(GL_LINES);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
    glEnd();

    glColor3f(0.0f, 0.0f, 1.0f);
    glBegin(GL_LINES);
    glVertex3f(0.0, 0.0, 0.0);
```

```
glVertex3f(0.0, 0.0, 1.0);
glEnd();

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
glEnable(GL_LIGHT2);
glEnable(GL_DEPTH_TEST);



glPushMatrix();
glTranslatef(0.0f, -0.7f, 0.0f);
glutSolidCube(0.5);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f, 0.0f, 0.0f);
glutSolidCube(0.5);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f, 0.7f, 0.0f);
glutSolidCube(0.5);
glPopMatrix();

glRotatef(rx, 1, 0, 0);
glRotatef(ry, 0, 1, 0);

glutSwapBuffers();
```

```
}

void KeysMove(unsigned char key, int x, int y)
{
    if (key == 'a' || key == 'A')
    {
        ry = 0;
        rx = 3;
    }
    else if (key == 'd' || key == 'D')
    {
        ry = 0;
        rx = -3;
    }
    else if (key == 119)
    {
        ry = 3;
        rx = 0;
    }
    else if (key == 115)
    {
        ry = -3;
        rx = 0;
    }
    else
    {
        ry = 0;
        rx = 0;
    }
```

```
    glutPostRedisplay();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(
        2.0f, 0.0f, 2.0f,
        0.0f, 0.0f, 0.0f,
        0.0f, 0.0f, 1.0f
    );
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(100, 100);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
```

```
    glutKeyboardFunc(KeysMove);                    //обработка нажатия
клавиш
    glutMainLoop();
    return 0;
}
```

4. Приложение

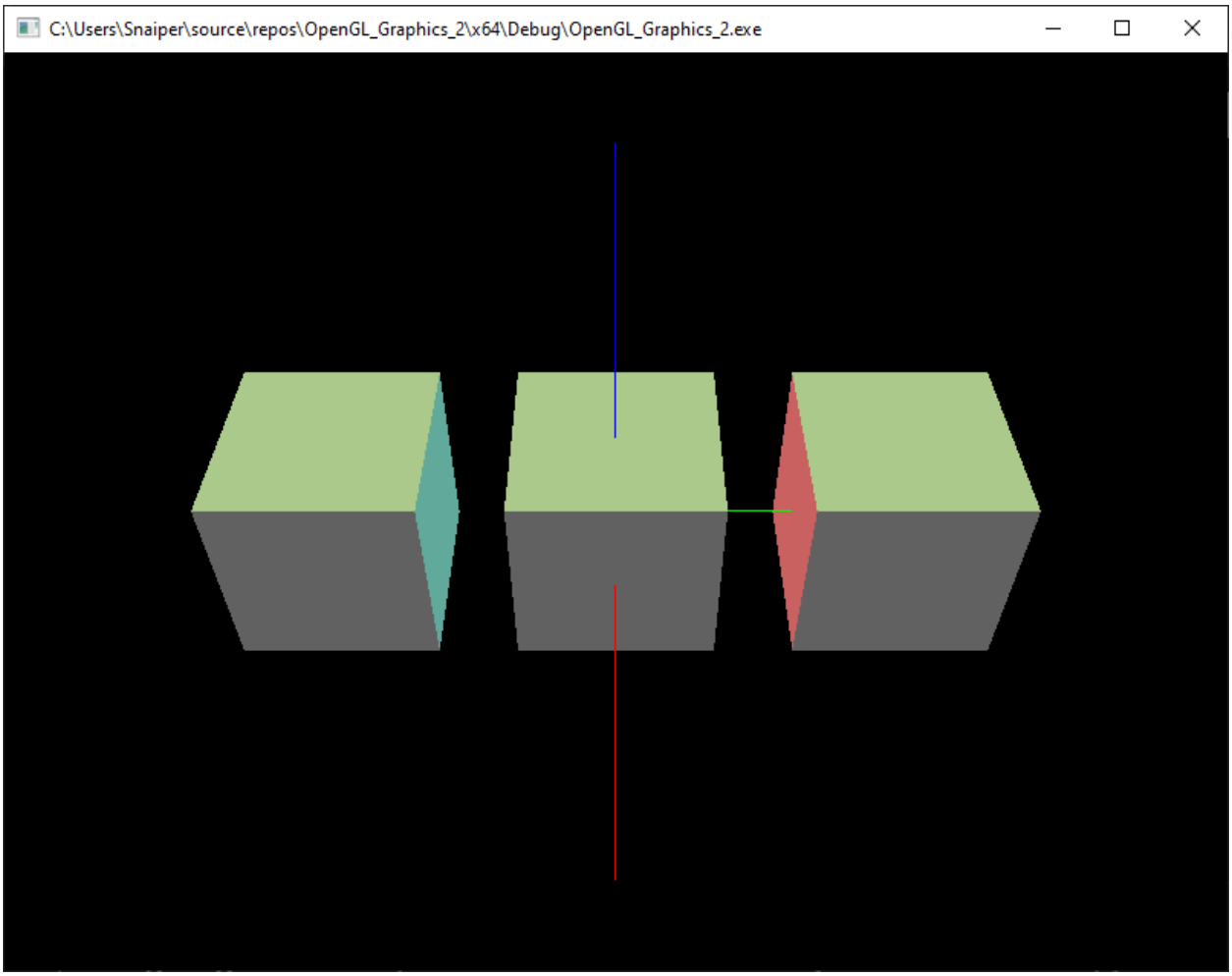

Рисунок 1 – пример выполнения программы

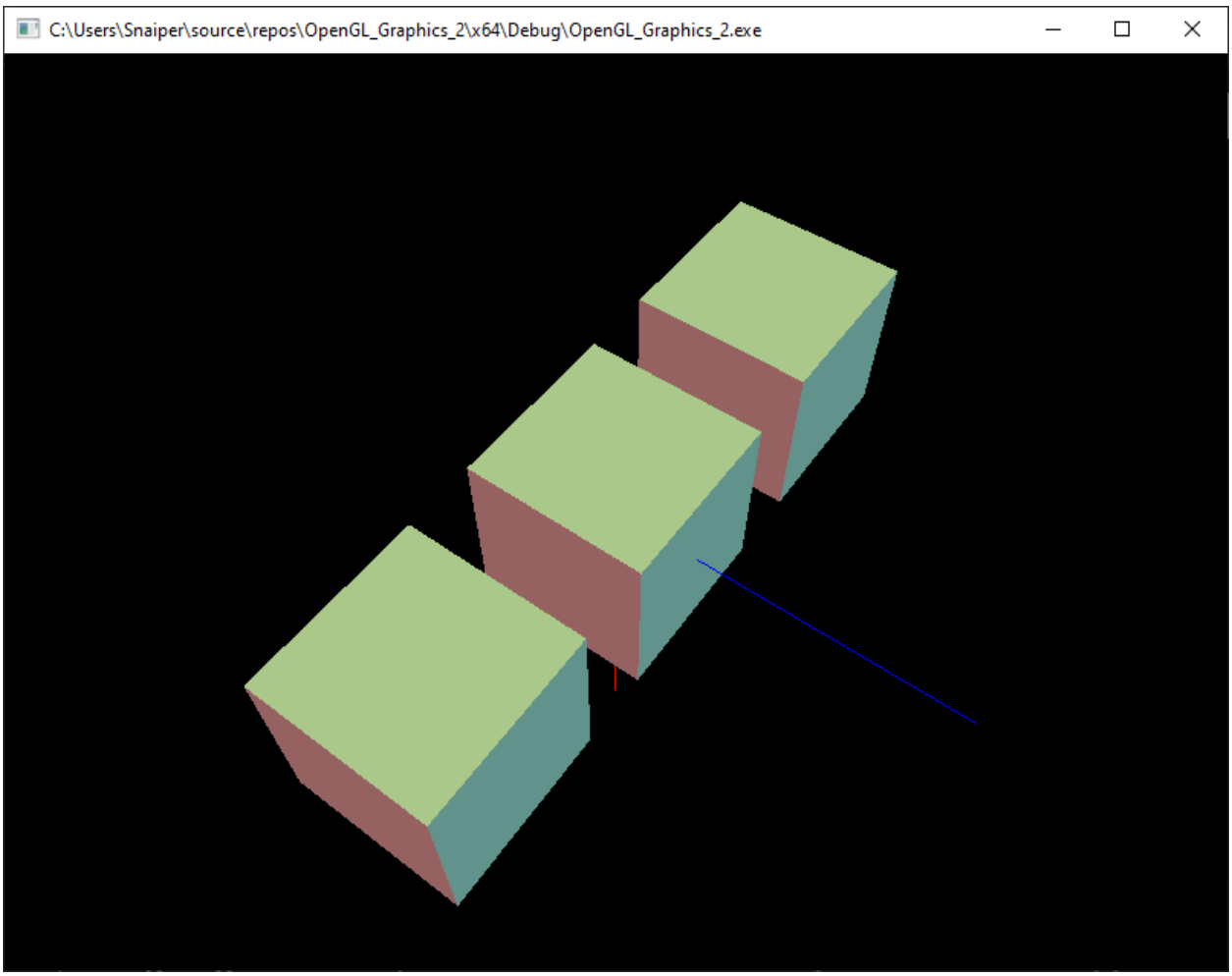

Рисунок 2 – пример выполнения программы

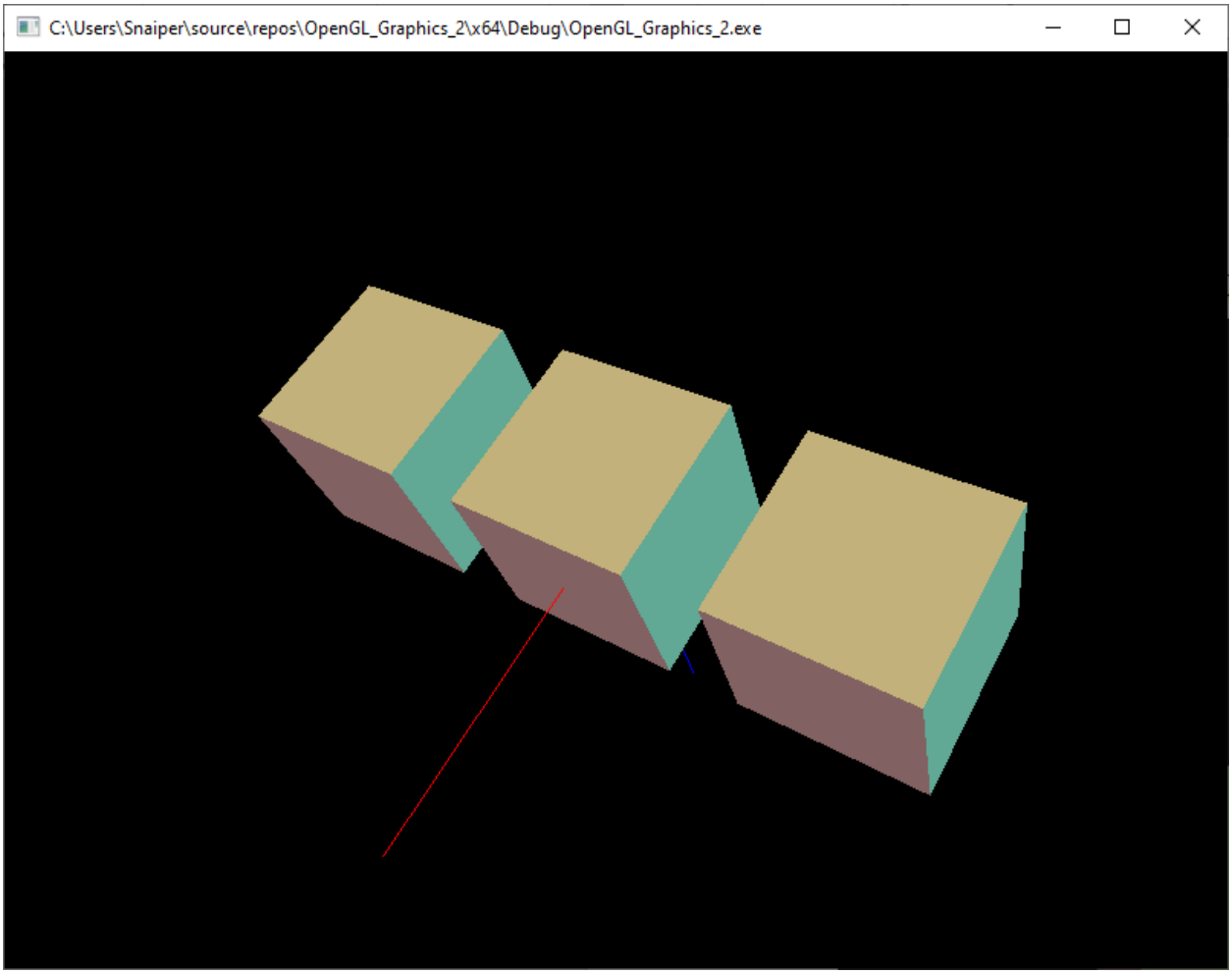

Рисунок 3 – пример выполнения программы