```
/*
 * OpenGL Blending Exersise, a simple/sample OpenGL program.
 *
 * Usage:
 *    g++ alpha3d.c -framework GLUT -framework OpenGL -framework Foundation
 *
 * This code places the following three rectangles in 3-D space;
 *    B: Opaque black rectangle that is parallel to the XY plane.
 *    R: Translucent red rectangle that is also parallel to XY.
 *    G: Translucent green rectangle that is parallel to YZ
 * The rectangle G crosses through B and R.
 * The points here (or in OpenGL blending in general) are:
 * (1) Place opaque objects first.
 * (2) Freeze the depth buffer by glDepthMask(GL_FALSE).
 * (3) But keep applying the depth test for hidden surface removal.
 *
 *   coded by Akira Kageyama, Kobe Univ.
 *         on 2012.05.07
 *         at Kobe
 */

#include <GLUT/glut.h>
#include <stdlib.h>

static int opaqueFirst = GL_TRUE;

/* initialize alpha blending function */
static void init(void)
{
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glShadeModel(GL_FLAT);
    glClearColor(1.0, 1.0, 1.0, 0.0);
}

static void drawRectangleXY(float r, float g, float b, float a, float z)
{
    glPushMatrix();
        glTranslatef(0.0, 0.0, z);
        glBegin(GL_QUADS);
            glColor4f(r, g, b, a);
            glVertex3f(0.0, 0.0, 0.0);
            glVertex3f(0.7, 0.0, 0.0);
            glVertex3f(0.7, 0.3, 0.0);
            glVertex3f(0.0, 0.3, 0.0);
        glEnd();
    glPopMatrix();
}

static void drawRectangleYZ(float r, float g, float b, float a, float x)
{
    glPushMatrix();
        glTranslatef(x, 0.0, 0.0);
        glBegin(GL_QUADS);
            glColor4f(r, g, b, a);
            glVertex3f(0.0, 0.0, 0.0);
            glVertex3f(0.0, 0.3, 0.0);
            glVertex3f(0.0, 0.3,-5.0);
            glVertex3f(0.0, 0.0,-5.0);
        glEnd();
    glPopMatrix();
}

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

```
    glEnable(GL_DEPTH_TEST);

    if (opaqueFirst) {
        drawRectangleXY(0.0, 0.0, 0.0, 1.0, -2.0);
        glDepthMask(GL_FALSE);
        drawRectangleXY(1.0, 0.0, 0.0, 0.3, -1.0);
        drawRectangleYZ(0.0, 1.0, 0.0, 0.3, 0.35);
        glDepthMask(GL_TRUE);
    }
    else {
        glDepthMask(GL_FALSE);
        drawRectangleXY(1.0, 0.0, 0.0, 0.3, -1.0);
        drawRectangleYZ(0.0, 1.0, 0.0, 0.3, 0.35);
        glDepthMask(GL_TRUE);
        drawRectangleXY(0.0, 0.0, 0.0, 1.0, -2.0);
    }
    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, w, h);

    glLoadIdentity();
    gluPerspective(15.0, (double)w / (double)h, 1.0, 100.0);
    gluLookAt(-1.0, -0.6, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case ’t’:
    case ’T’:
        opaqueFirst = !opaqueFirst;
        glutPostRedisplay();
        break;
    case 27:  /* Escape key */
        exit(0);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(400, 400);
    glutCreateWindow(argv[0]);
    init();
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```