**Translation:**

#include <windows.h> // for MS Windows

#include <GL/glut.h> // GLUT, include glu.h and gl.h

/\* Initialize OpenGL Graphics \*/

void initGL() {

// Set "clearing" or background color

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque

}

/\* Handler for window-repaint event. Call back when the window first appears and

whenever the window needs to be re-painted. \*/

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer

glLoadIdentity(); // Reset the model-view matrix

glTranslatef(+0.0f, 0.0f, 0.0f);

glBegin(GL\_QUADS); // Each set of 4 vertices form a quad

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

glVertex2f(-0.1f, -0.1f); // Define vertices in counter-clockwise (CCW) order

glVertex2f( 0.1f, -0.1f); // so that the normal (front-face) is facing you

glVertex2f( 0.1f, 0.1f);

glVertex2f(-0.1f, 0.1f);

glEnd();

glTranslatef(+0.3f, 0.1f, 0.0f); // Translate x right and y up

glBegin(GL\_QUADS); // Each set of 4 vertices form a quad

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

glVertex2f(-0.1f, -0.1f); // Define vertices in counter-clockwise (CCW) order

glVertex2f( 0.1f, -0.1f); // so that the normal (front-face) is facing you

glVertex2f( 0.1f, 0.1f);

glVertex2f(-0.1f, 0.1f);

glEnd();

glLoadIdentity(); // Reset the model-view matrix

glTranslatef(-0.3f, -0.1f, 0.0f); // Translate x left and y down

glBegin(GL\_QUADS); // Each set of 4 vertices form a quad

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

glVertex2f(-0.1f, -0.1f); // Define vertices in counter-clockwise (CCW) order

glVertex2f( 0.1f, -0.1f); // so that the normal (front-face) is facing you

glVertex2f( 0.1f, 0.1f);

glVertex2f(-0.1f, 0.1f);

glEnd();

glFlush(); // Render now

}

/\* Main function: GLUT runs as a console application starting at main() \*/

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

glutInit(&argc, argv); // Initialize GLUT

glutInitWindowSize(320, 320); // Set the window's initial width & height - non-square

glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

glutCreateWindow("Model Transform"); // Create window with the given title

glutDisplayFunc(display); // Register callback handler for window re-paint event

//glutReshapeFunc(reshape); // Register callback handler for window re-size event

initGL(); // Our own OpenGL initialization

glutMainLoop(); // Enter the infinite event-processing loop

return 0;

}

**Scaling:**

#include <windows.h>  // for MS Windows

#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/\* Initialize OpenGL Graphics \*/

void initGL() {

   // Set "clearing" or background color

   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque

}

/\* Handler for window-repaint event. Call back when the window first appears and

   whenever the window needs to be re-painted. \*/

void display() {

   glClear(GL\_COLOR\_BUFFER\_BIT);    // Clear the color buffer

   //glMatrixMode(GL\_MODELVIEW);      // To operate on Model-View matrix

   glLoadIdentity();                // Reset the model-view matrix

glTranslatef(+0.0f, 0.0f, 0.0f);

   glBegin(GL\_QUADS);               // Each set of 4 vertices form a quad

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

      glVertex2f(-0.1f, -0.1f);     // Define vertices in counter-clockwise (CCW) order

      glVertex2f( 0.1f, -0.1f);     //  so that the normal (front-face) is facing you

      glVertex2f( 0.1f,  0.1f);

      glVertex2f(-0.1f,  0.1f);

   glEnd();

   glScalef(2,2,0); // increase x right and y

glTranslatef(+0.3f, 0.1f, 0.0f); // Translate x right and y up

   glBegin(GL\_QUADS);               // Each set of 4 vertices form a quad

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

      glVertex2f(-0.1f, -0.1f);     // Define vertices in counter-clockwise (CCW) order

      glVertex2f( 0.1f, -0.1f);     //  so that the normal (front-face) is facing you

      glVertex2f( 0.1f,  0.1f);

      glVertex2f(-0.1f,  0.1f);

   glEnd();

   glFlush();   // Render now

}

/\* Main function: GLUT runs as a console application starting at main() \*/

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

   glutInit(&argc, argv);          // Initialize GLUT

   glutInitWindowSize(320, 320);   // Set the window's initial width & height - non-square

   glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

   glutCreateWindow("Model Transform");  // Create window with the given title

   glutDisplayFunc(display);       // Register callback handler for window re-paint event

   //glutReshapeFunc(reshape);       // Register callback handler for window re-size event

   initGL();                       // Our own OpenGL initialization

   glutMainLoop();                 // Enter the infinite event-processing loop

   return 0;

}

**Rotation:**

#include <windows.h>

#include <GL/glut.h>

GLfloat i = 0.0f;

void initGL()

{

    glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque

}

void Idle()

{

    glutPostRedisplay();//// marks the current window as needing to be redisplayed

}

void display()

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();//Reset the current matrix

    ///glTranslatef(0.4,0.4,0);

    glPushMatrix(); //glPushMatrix copies the tp matrix and pushes it onto the stack, while glPopMatrix pops the top matrix off the stack

        glRotatef(i,0.0,1.0,0.0);//i=how many degree you want to rotate around an axis

        //glLineWidth(3.0);

        // glTranslatef(0.f,0.0f,0.f);

        glBegin(GL\_LINES);

        glColor3f(1.0f, 0.0f, 0.0f);

        glVertex2f(0.0f, 0.0f);

        glVertex2f( 0.6f, 0.0f);

        glVertex2f(0.0f, 0.0f);

        glVertex2f( 0.0f, 0.6f);

        glVertex2f(0.0f, 0.0f);

        glVertex2f( -0.6f, 0.0f);

        //glVertex2f(.5f, .3f);

        // glVertex2f( .7f, 0.3f);

        glEnd();

    glPopMatrix();

    i+=0.02f;

    glFlush();

}

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

{

    glutInit(&argc, argv);          // Initialize GLUT

    glutInitWindowSize(320, 320);

    glutCreateWindow("Model Transform");

    glutDisplayFunc(display);//

    initGL();

    glutIdleFunc(Idle);//glutIdleFunc sets the global idle callback to be func so a GLUT program can perform background processing tasks or continuous animation when window system events are not being received.

    glutMainLoop();

    return 0;

}