```
//ROTATE

#include <iostream>
#include <GL/gl.h>
#include <GL/glut.h>
#include <windows.h>


using namespace std;


float _angle1 = 0.0f;   // rotation angle


// Timer function for animation
void update(int value) {
   _angle1 += 2.0f;        // increase angle
   if (_angle1 > 360)      // reset after full rotation
      _angle1 -= 360;


   glutPostRedisplay();     // request redraw
   glutTimerFunc(20, update, 0); // call update again after 20 ms
}


void drawScene() {
   glClear(GL_COLOR_BUFFER_BIT);


   // Original object (static)
   glColor3d(1, 0, 0);
   glBegin(GL_QUADS);
      glVertex2f(0.1f, 0.0f);
      glVertex2f(0.5f, 0.0f);
      glVertex2f(0.5f, 0.2f);
      glVertex2f(0.1f, 0.2f);
   glEnd();
```

```
    // Rotating object
    glPushMatrix();
    glRotatef(_angle1, 0.0f, 0.0f, 1.0f);


    glBegin(GL_QUADS);
        glVertex2f(0.1f, 0.0f);
        glVertex2f(0.5f, 0.0f);
        glVertex2f(0.5f, 0.2f);
        glVertex2f(0.1f, 0.2f);
    glEnd();


    glPopMatrix();


    glutSwapBuffers();
}


int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Rotation Animation");


    glutDisplayFunc(drawScene);
    glutTimerFunc(20, update, 0);  // start animation timer


    glutMainLoop();
    return 0;
}


//SCALE
```

```
#include <iostream>
#include <GL/gl.h>
#include <GL/glut.h>
#include <windows.h>


using namespace std;


float _scaleX = 1.5f;   // scale in X direction
float _scaleY = 0.5f;   // scale in Y direction


void drawScene() {
  glClear(GL_COLOR_BUFFER_BIT);


  // Original object (no scaling)
  glColor3f(1, 0, 0);
  glBegin(GL_QUADS);
    glVertex2f(0.1f, 0.0f);
    glVertex2f(0.5f, 0.0f);
    glVertex2f(0.5f, 0.2f);
    glVertex2f(0.1f, 0.2f);
  glEnd();


  // Scaled object
  glPushMatrix();             // save current matrix
  glScalef(_scaleX, _scaleY, 1.0f); // scaling transformation


  glBegin(GL_QUADS);
    glVertex2f(0.1f, 0.0f);
    glVertex2f(0.5f, 0.0f);
    glVertex2f(0.5f, 0.2f);
    glVertex2f(0.1f, 0.2f);
  glEnd();
```

```
    glPopMatrix();              // restore matrix

    glutSwapBuffers();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Scaling Transformation");

    glutDisplayFunc(drawScene);
    glutMainLoop();

    return 0;
}
```

//Shearing

```
#include <iostream>
#include <GL/gl.h>
#include <GL/glut.h>
#include <windows.h>

using namespace std;

// Shear factors
float _shearX = 1.5f;   // X shear
float _shearY = 1.5f;   // Y shear
```

```
void drawScene() {
  glClear(GL_COLOR_BUFFER_BIT);


  // Original object (no shearing)
  glColor3f(1, 0, 0);
  glBegin(GL_QUADS);
    glVertex2f(0.1f, 0.0f);
    glVertex2f(0.5f, 0.0f);
    glVertex2f(0.5f, 0.2f);
    glVertex2f(0.1f, 0.2f);
  glEnd();


  // Sheared object
  glPushMatrix();


  // Shearing matrix (column-major order)
  GLfloat shearMatrix[16] = {
    1.0f,  _shearY, 0.0f, 0.0f,
    _shearX, 1.0f,  0.0f, 0.0f,
    0.0f,  0.0f,  1.0f, 0.0f,
    0.0f,  0.0f,  0.0f, 1.0f
  };


  glMultMatrixf(shearMatrix);


  glBegin(GL_QUADS);
  glColor3f(0,1,0);
    glVertex2f(0.1f, 0.0f);
    glVertex2f(0.5f, 0.0f);
    glVertex2f(0.5f, 0.2f);
    glVertex2f(0.1f, 0.2f);
  glEnd();
```

```
    glPopMatrix();


    glutSwapBuffers();
}


int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Shearing Transformation");


    glutDisplayFunc(drawScene);
    glutMainLoop();


    return 0;
}
```