Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Lab Report on

"Computer Graphics"

[Code No: COMP342]

LAB- V

Submitted by:

Aakash Thakur

CS III-I

Roll-No: 20

Submitted to:

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date:

2026/1/13

1. Implement the following 3D transformations using the 3D shapes provided by Opengl:

a. Translation

b. Rotation

c. Shearing

d. Scaling

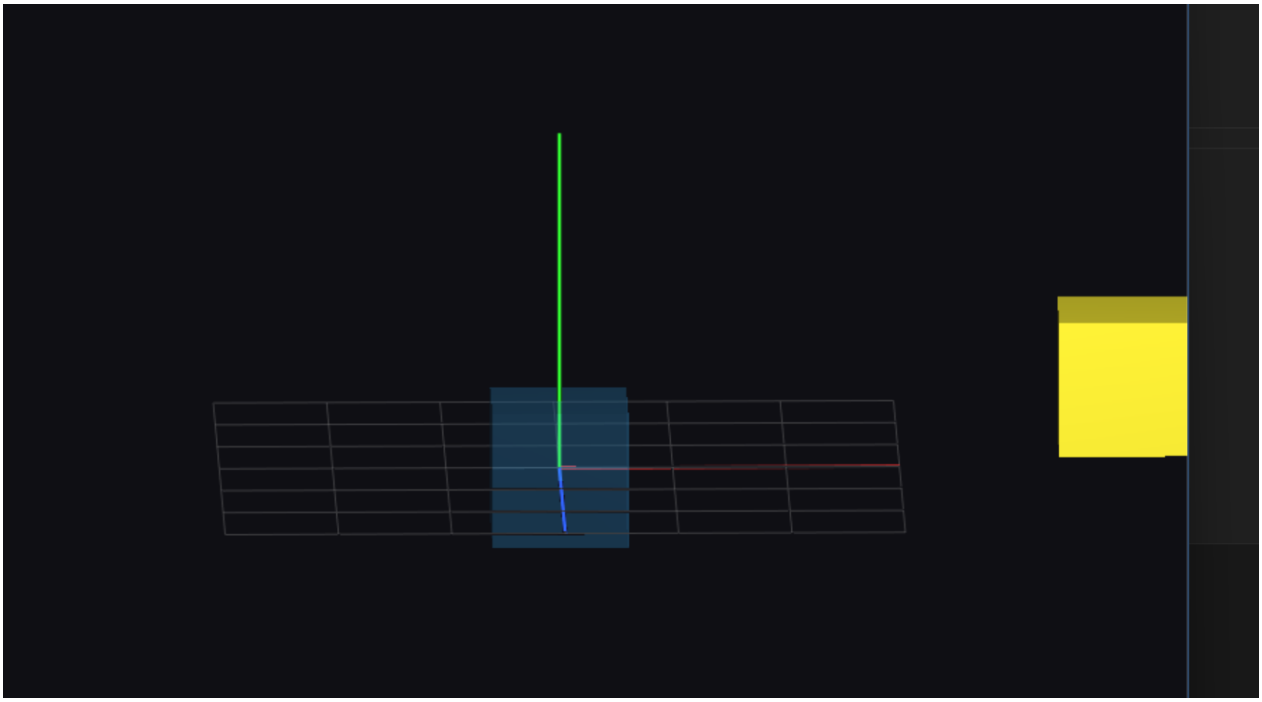
Translation:

```python
import sys
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

TX, TY, TZ = 5, 0.8, 0

rot_x, rot_y = 20.0, -35.0
zoom = 7.0
ortho_size = 4.0

last_x, last_y = 0, 0
left_down = False
right_down = False


def init_gl():
    glClearColor(0.06, 0.06, 0.08, 1.0)
    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LESS)

    glEnable(GL_LINE_SMOOTH)
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)

    glEnable(GL_BLEND)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)

    glLightfv(GL_LIGHT0, GL_POSITION, (4.0, 6.0, 8.0, 1.0))


def draw_axes(length=3.0):
    glDisable(GL_LIGHTING)
    glLineWidth(2.5)
    glBegin(GL_LINES)

    glColor3f(1.0, 0.2, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(length, 0.0, 0.0)

    glColor3f(0.2, 1.0, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, length, 0.0)

    glColor3f(0.2, 0.4, 1.0)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, 0.0, length)

    glEnd()
    glEnable(GL_LIGHTING)


def draw_ground_grid(size=6, step=1.0):
    glDisable(GL_LIGHTING)
    glLineWidth(1.0)
    glColor4f(0.6, 0.6, 0.6, 0.35)

    half = size * step * 0.5
    glBegin(GL_LINES)
    for i in range(size + 1):
        x = -half + i * step
        glVertex3f(x, 0.0, -half)
        glVertex3f(x, 0.0, +half)
        z = -half + i * step
        glVertex3f(-half, 0.0, z)
        glVertex3f(+half, 0.0, z)
    glEnd()
    glEnable(GL_LIGHTING)


def draw_cube(size=1.2):
    s = size * 0.5

    glBegin(GL_QUADS)

    glNormal3f(0, 0, 1)
    glVertex3f(-s, -s, +s)
    glVertex3f(+s, -s, +s)
    glVertex3f(+s, +s, +s)
    glVertex3f(-s, +s, +s)

    glNormal3f(0, 0, -1)
    glVertex3f(-s, -s, -s)
    glVertex3f(-s, +s, -s)
    glVertex3f(+s, +s, -s)
    glVertex3f(+s, -s, -s)

    glNormal3f(-1, 0, 0)
    glVertex3f(-s, -s, -s)
    glVertex3f(-s, -s, +s)
    glVertex3f(-s, +s, +s)
    glVertex3f(-s, +s, -s)

    glNormal3f(1, 0, 0)
    glVertex3f(+s, -s, -s)
    glVertex3f(+s, +s, -s)
    glVertex3f(+s, +s, +s)
    glVertex3f(+s, -s, +s)
```

```
    glNormal3f(0, 1, 0)
    glVertex3f(-s, +s, -s)
    glVertex3f(-s, +s, +s)
    glVertex3f(+s, +s, +s)
    glVertex3f(+s, +s, -s)

    glNormal3f(0, -1, 0)
    glVertex3f(-s, -s, -s)
    glVertex3f(+s, -s, -s)
    glVertex3f(+s, -s, +s)
    glVertex3f(-s, -s, +s)

    glEnd()


def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    gluLookAt(0.0, 2.2, zoom, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0)

    glRotatef(rot_x, 1.0, 0.0, 0.0)
    glRotatef(rot_y, 0.0, 1.0, 0.0)

    draw_ground_grid()
    draw_axes()

    glPushMatrix()
    glTranslatef(TX, TY, TZ)
    glColor4f(0.95, 0.90, 0.20, 1.0)
    draw_cube()
    glPopMatrix()

    glDepthMask(GL_FALSE)
    glPushMatrix()
    glColor4f(0.25, 0.70, 1.00, 0.28)
    draw_cube()
    glPopMatrix()
    glDepthMask(GL_TRUE)

    glutSwapBuffers()


def reshape(w, h):
    global ortho_size
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    aspect = w / float(h)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()

    if aspect >= 1.0:
        glOrtho(-ortho_size * aspect, ortho_size * aspect, -ortho_size, ortho_size, 0.1, 100.0)
    else:
        glOrtho(-ortho_size, ortho_size, -ortho_size / aspect, ortho_size / aspect, 0.1, 100.0)

    glMatrixMode(GL_MODELVIEW)


def mouse(button, state, x, y):
    global left_down, right_down, last_x, last_y
    if button == GLUT_LEFT_BUTTON:
        left_down = (state == GLUT_DOWN)
    elif button == GLUT_RIGHT_BUTTON:
        right_down = (state == GLUT_DOWN)
    last_x, last_y = x, y


def motion(x, y):
    global rot_x, rot_y, ortho_size, last_x, last_y
    dx = x - last_x
    dy = y - last_y

    if left_down:
        rot_y += dx * 0.5
        rot_x += dy * 0.5
        rot_x = max(-89.0, min(89.0, rot_x))
    elif right_down:
        ortho_size += dy * 0.02
        ortho_size = max(1.2, min(12.0, ortho_size))
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)

    last_x, last_y = x, y
    glutPostRedisplay()
```

```python
def keyboard(key, x, y):
    global rot_x, rot_y, zoom, ortho_size
    k = key.decode("utf-8").lower()
    if k == "r":
        rot_x, rot_y = 20.0, -35.0
        zoom = 7.0
        ortho_size = 4.0
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)
        glutPostRedisplay()
    elif k == "q" or ord(key) == 27:
        sys.exit(0)


def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(900, 650)
    glutCreateWindow(b"3D Translation: Original (Translucent) vs Translated (Solid) [ORTHO]")

    init_gl()

    glutDisplayFunc(display)
    glutReshapeFunc(reshape)
    glutMouseFunc(mouse)
    glutMotionFunc(motion)
    glutKeyboardFunc(keyboard)

    glutMainLoop()


if __name__ == "__main__":
    main()
```

Output:



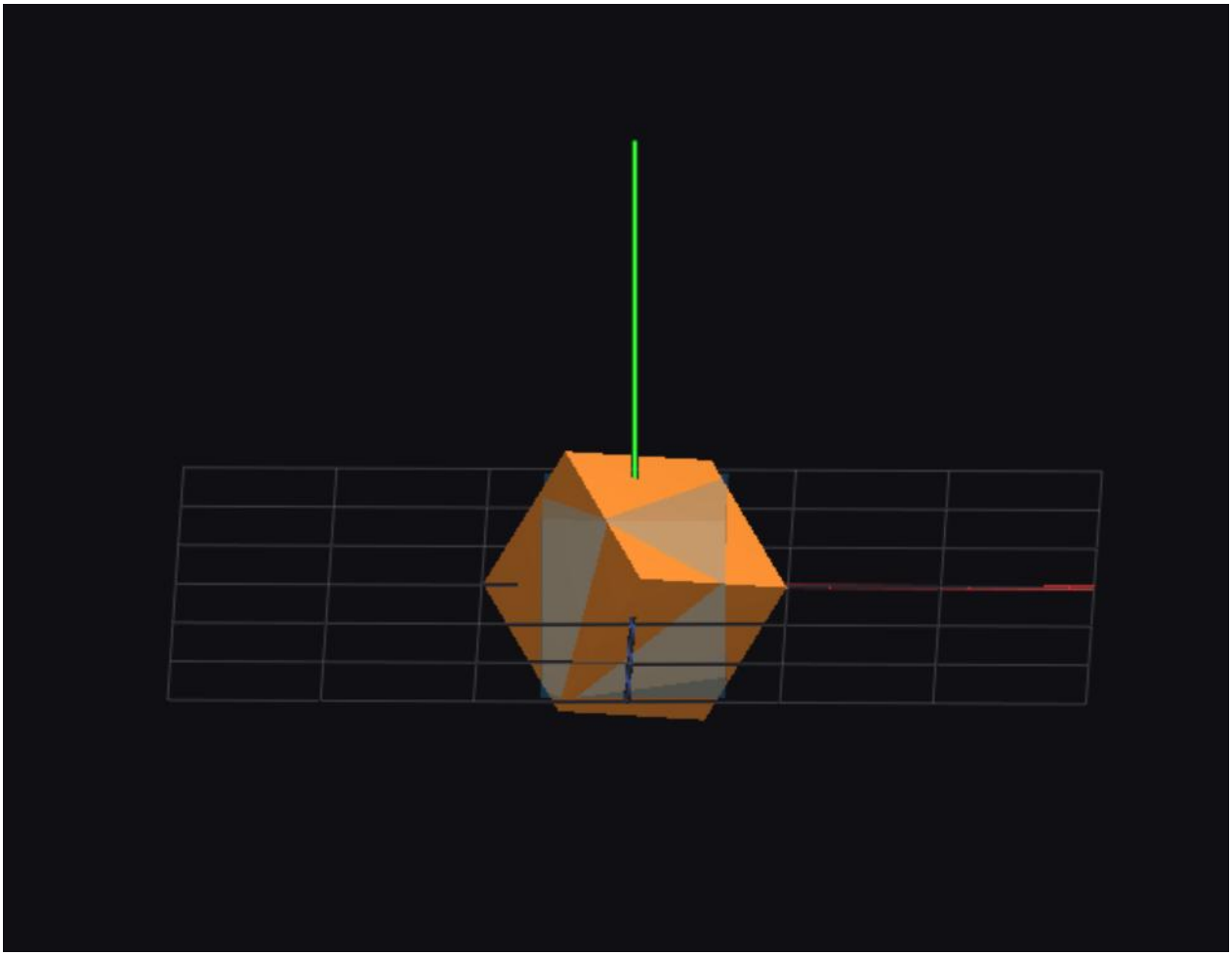
Rotation:

```python
import sys
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

RX, RY, RZ = 35.0, 25.0, 60.0

rot_x, rot_y = 20.0, -35.0
zoom = 7.0
ortho_size = 4.0

last_x, last_y = 0, 0
left_down = False
right_down = False


def init_gl():
    glClearColor(0.06, 0.06, 0.08, 1.0)
    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LESS)

    glEnable(GL_LINE_SMOOTH)
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)

    glEnable(GL_BLEND)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)
    glLightfv(GL_LIGHT0, GL_POSITION, (4.0, 6.0, 8.0, 1.0))


def draw_axes(length=3.0):
    glDisable(GL_LIGHTING)
    glLineWidth(2.5)
    glBegin(GL_LINES)

    glColor3f(1.0, 0.2, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(length, 0.0, 0.0)

    glColor3f(0.2, 1.0, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, length, 0.0)

    glColor3f(0.2, 0.4, 1.0)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, 0.0, length)

    glEnd()
    glEnable(GL_LIGHTING)


def draw_ground_grid(size=6, step=1.0):
    glDisable(GL_LIGHTING)
    glLineWidth(1.0)
    glColor4f(0.6, 0.6, 0.6, 0.35)

    half = size * step * 0.5
    glBegin(GL_LINES)
    for i in range(size + 1):
        x = -half + i * step
        glVertex3f(x, 0.0, -half)
        glVertex3f(x, 0.0, +half)

        z = -half + i * step
        glVertex3f(-half, 0.0, z)
        glVertex3f(+half, 0.0, z)
    glEnd()
    glEnable(GL_LIGHTING)


def draw_cube(size=1.2):
    s = size * 0.5
    glBegin(GL_QUADS)

    glNormal3f(0, 0, 1)
    glVertex3f(-s, -s, +s); glVertex3f(+s, -s, +s); glVertex3f(+s, +s, +s); glVertex3f(-s, +s, +s)

    glNormal3f(0, 0, -1)
    glVertex3f(-s, -s, -s); glVertex3f(-s, +s, -s); glVertex3f(+s, +s, -s); glVertex3f(+s, -s, -s)

    glNormal3f(-1, 0, 0)
    glVertex3f(-s, -s, -s); glVertex3f(-s, -s, +s); glVertex3f(-s, +s, +s); glVertex3f(-s, +s, -s)

    glNormal3f(1, 0, 0)
    glVertex3f(+s, -s, -s); glVertex3f(+s, +s, -s); glVertex3f(+s, +s, +s); glVertex3f(+s, -s, +s)

    glNormal3f(0, 1, 0)
    glVertex3f(-s, +s, -s); glVertex3f(-s, +s, +s); glVertex3f(+s, +s, +s); glVertex3f(+s, +s, -s)

    glNormal3f(0, -1, 0)
    glVertex3f(-s, -s, -s); glVertex3f(+s, -s, -s); glVertex3f(+s, -s, +s); glVertex3f(-s, -s, +s)

    glEnd()
```

```
def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    gluLookAt(0.0, 2.2, zoom, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0)

    glRotatef(rot_x, 1.0, 0.0, 0.0)
    glRotatef(rot_y, 0.0, 1.0, 0.0)

    draw_ground_grid()
    draw_axes()

    glPushMatrix()
    glRotatef(RX, 1.0, 0.0, 0.0)
    glRotatef(RY, 0.0, 1.0, 0.0)
    glRotatef(RZ, 0.0, 0.0, 1.0)
    glColor4f(0.95, 0.55, 0.20, 1.0)
    draw_cube()
    glPopMatrix()

    glDepthMask(GL_FALSE)
    glPushMatrix()
    glColor4f(0.25, 0.70, 1.00, 0.28)
    draw_cube()
    glPopMatrix()
    glDepthMask(GL_TRUE)

    glutSwapBuffers()


def reshape(w, h):
    global ortho_size
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    aspect = w / float(h)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()

    if aspect >= 1.0:
        glOrtho(-ortho_size * aspect, ortho_size * aspect, -ortho_size, ortho_size, 0.1, 100.0)
    else:
        glOrtho(-ortho_size, ortho_size, -ortho_size / aspect, ortho_size / aspect, 0.1, 100.0)

    glMatrixMode(GL_MODELVIEW)


def mouse(button, state, x, y):
    global left_down, right_down, last_x, last_y
    if button == GLUT_LEFT_BUTTON:
        left_down = (state == GLUT_DOWN)
    elif button == GLUT_RIGHT_BUTTON:
        right_down = (state == GLUT_DOWN)
    last_x, last_y = x, y


def motion(x, y):
    global rot_x, rot_y, zoom, ortho_size, last_x, last_y
    dx = x - last_x
    dy = y - last_y

    if left_down:
        rot_y += dx * 0.5
        rot_x += dy * 0.5
        rot_x = max(-89.0, min(89.0, rot_x))
    elif right_down:
        ortho_size += dy * 0.02
        ortho_size = max(1.2, min(12.0, ortho_size))
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)

    last_x, last_y = x, y
    glutPostRedisplay()


def keyboard(key, x, y):
    global rot_x, rot_y, zoom, ortho_size
    k = key.decode("utf-8").lower()
    if k == "r":
        rot_x, rot_y = 20.0, -35.0
        zoom = 7.0
        ortho_size = 4.0
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)
        glutPostRedisplay()
    elif k == "q" or ord(key) == 27:
        sys.exit(0)
```

```python
def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(900, 650)
    glutCreateWindow(b"3D Rotation: Original (Translucent) vs Rotated (Solid) [ORTHO]")

    init_gl()

    glutDisplayFunc(display)
    glutReshapeFunc(reshape)
    glutMouseFunc(mouse)
    glutMotionFunc(motion)
    glutKeyboardFunc(keyboard)

    glutMainLoop()


if __name__ == "__main__":
    main()
```

Output:



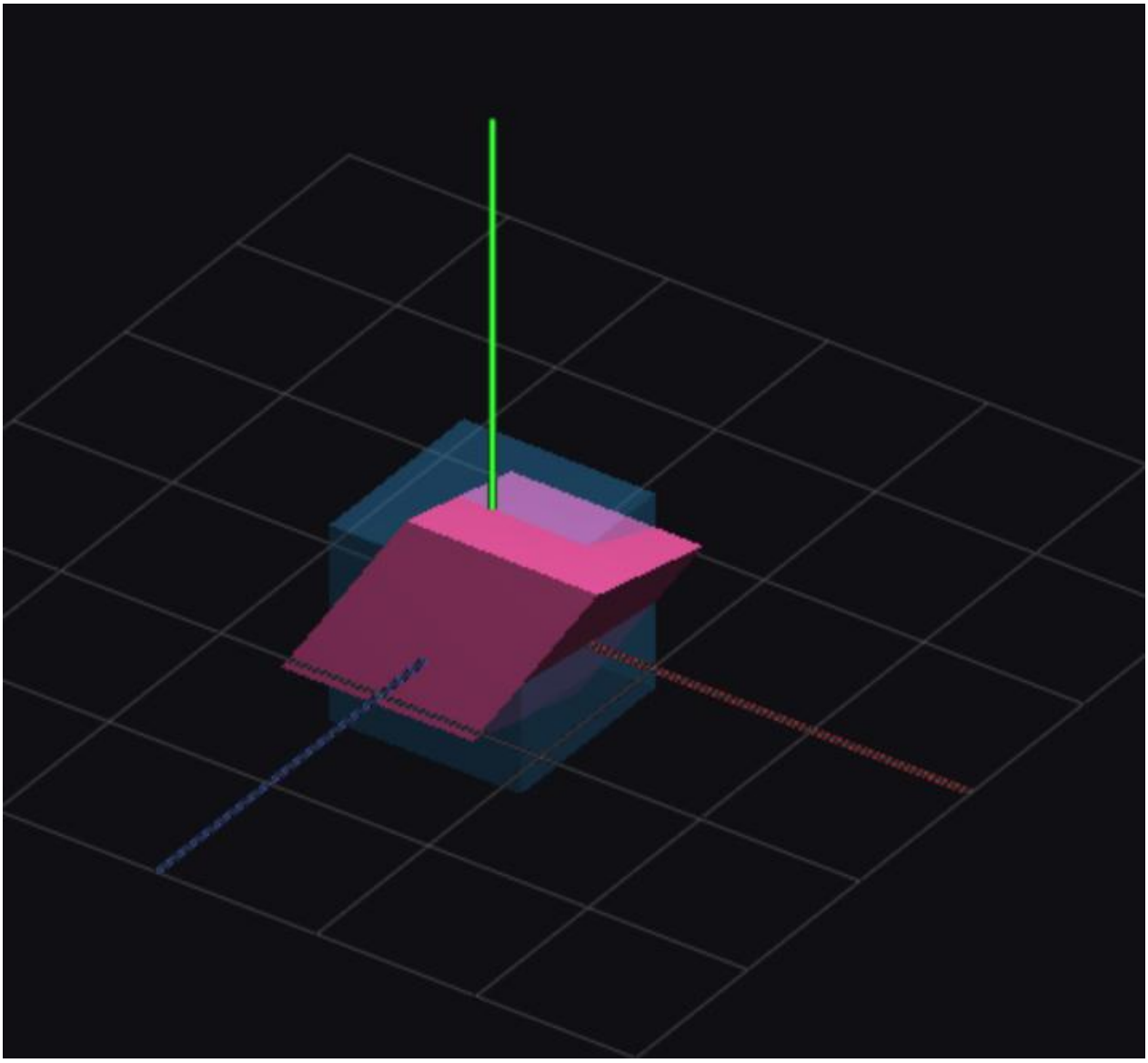
Shearing

```
Orthogonal > shearing-3D.py > reshape
import sys
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

shXY, shXZ = 0.65, 0.15
shYX, shYZ = 0.00, 0.35
shZX, shZY = 0.00, 0.00

rot_x, rot_y = 20.0, -35.0
zoom = 7.0
ortho_size = 4.0

last_x, last_y = 0, 0
left_down = False
right_down = False


def init_gl():
    glClearColor(0.06, 0.06, 0.08, 1.0)
    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LESS)

    glEnable(GL_LINE_SMOOTH)
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)

    glEnable(GL_BLEND)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)
    glLightfv(GL_LIGHT0, GL_POSITION, (4.0, 6.0, 8.0, 1.0))


def draw_axes(length=3.0):
    glDisable(GL_LIGHTING)
    glLineWidth(2.5)
    glBegin(GL_LINES)

    glColor3f(1.0, 0.2, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(length, 0.0, 0.0)

    glColor3f(0.2, 1.0, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, length, 0.0)

    glColor3f(0.2, 0.4, 1.0)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, 0.0, length)

    glEnd()
    glEnable(GL_LIGHTING)


def draw_ground_grid(size=6, step=1.0):
    glDisable(GL_LIGHTING)
    glLineWidth(1.0)
    glColor4f(0.6, 0.6, 0.6, 0.35)

    half = size * step * 0.5
    glBegin(GL_LINES)
    for i in range(size + 1):
        x = -half + i * step
        glVertex3f(x, 0.0, -half)
        glVertex3f(x, 0.0, +half)

        z = -half + i * step
        glVertex3f(-half, 0.0, z)
        glVertex3f(+half, 0.0, z)
    glEnd()
    glEnable(GL_LIGHTING)


def draw_cube(size=1.2):
    s = size * 0.5
    glBegin(GL_QUADS)

    glNormal3f(0, 0, 1)
    glVertex3f(-s, -s, +s); glVertex3f(+s, -s, +s); glVertex3f(+s, +s, +s); glVertex3f(-s, +s, +s)

    glNormal3f(0, 0, -1)
    glVertex3f(-s, -s, -s); glVertex3f(-s, +s, -s); glVertex3f(+s, +s, -s); glVertex3f(+s, -s, -s)

    glNormal3f(-1, 0, 0)
    glVertex3f(-s, -s, -s); glVertex3f(-s, -s, +s); glVertex3f(-s, +s, +s); glVertex3f(-s, +s, -s)

    glNormal3f(1, 0, 0)
    glVertex3f(+s, -s, -s); glVertex3f(+s, +s, -s); glVertex3f(+s, +s, +s); glVertex3f(+s, -s, +s)

    glNormal3f(0, 1, 0)
    glVertex3f(-s, +s, -s); glVertex3f(-s, +s, +s); glVertex3f(+s, +s, +s); glVertex3f(+s, +s, -s)

    glNormal3f(0, -1, 0)
    glVertex3f(-s, -s, -s); glVertex3f(+s, -s, -s); glVertex3f(+s, -s, +s); glVertex3f(-s, -s, +s)

    glEnd()
```

```
def apply_shear_matrix():
    m_col_major = [
        1.0,  shYX, shZX, 0.0,
        shXY, 1.0,  shZY, 0.0,
        shXZ, shYZ, 1.0,  0.0,
        0.0,  0.0,  0.0,  1.0
    ]
    glMultMatrixf(m_col_major)


def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    gluLookAt(0.0, 2.2, zoom, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0)

    glRotatef(rot_x, 1.0, 0.0, 0.0)
    glRotatef(rot_y, 0.0, 1.0, 0.0)

    draw_ground_grid()
    draw_axes()

    glPushMatrix()
    apply_shear_matrix()
    glColor4f(0.95, 0.35, 0.65, 1.0)
    draw_cube()
    glPopMatrix()

    glDepthMask(GL_FALSE)
    glPushMatrix()
    glColor4f(0.25, 0.70, 1.00, 0.28)
    draw_cube()
    glPopMatrix()
    glDepthMask(GL_TRUE)

    glutSwapBuffers()


def reshape(w, h):
    global ortho_size
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    aspect = w / float(h)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()

    if aspect >= 1.0:
        glOrtho(-ortho_size * aspect, ortho_size * aspect, -ortho_size, ortho_size, 0.1, 100.0)
    else:
        glOrtho(-ortho_size, ortho_size, -ortho_size / aspect, ortho_size / aspect, 0.1, 100.0)

    glMatrixMode(GL_MODELVIEW)


def mouse(button, state, x, y):
    global left_down, right_down, last_x, last_y
    if button == GLUT_LEFT_BUTTON:
        left_down = (state == GLUT_DOWN)
    elif button == GLUT_RIGHT_BUTTON:
        right_down = (state == GLUT_DOWN)
    last_x, last_y = x, y


def motion(x, y):
    global rot_x, rot_y, ortho_size, last_x, last_y
    dx = x - last_x
    dy = y - last_y

    if left_down:
        rot_y += dx * 0.5
        rot_x += dy * 0.5
        rot_x = max(-89.0, min(89.0, rot_x))
    elif right_down:
        ortho_size += dy * 0.02
        ortho_size = max(1.2, min(12.0, ortho_size))
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)

    last_x, last_y = x, y
    glutPostRedisplay()


def keyboard(key, x, y):
    global rot_x, rot_y, zoom, ortho_size
    k = key.decode("utf-8").lower()
    if k == "r":
        rot_x, rot_y = 20.0, -35.0
        zoom = 7.0
        ortho_size = 4.0
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)
        glutPostRedisplay()
    elif k == "q" or ord(key) == 27:
        sys.exit(0)
```

```
def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(900, 650)
    glutCreateWindow(b"3D Shearing: Original (Translucent) vs Sheared (Solid) [ORTHO]")

    init_gl()

    glutDisplayFunc(display)
    glutReshapeFunc(reshape)
    glutMouseFunc(mouse)
    glutMotionFunc(motion)
    glutKeyboardFunc(keyboard)
    glutMainLoop()


if __name__ == "__main__":
    main()
```

Output:



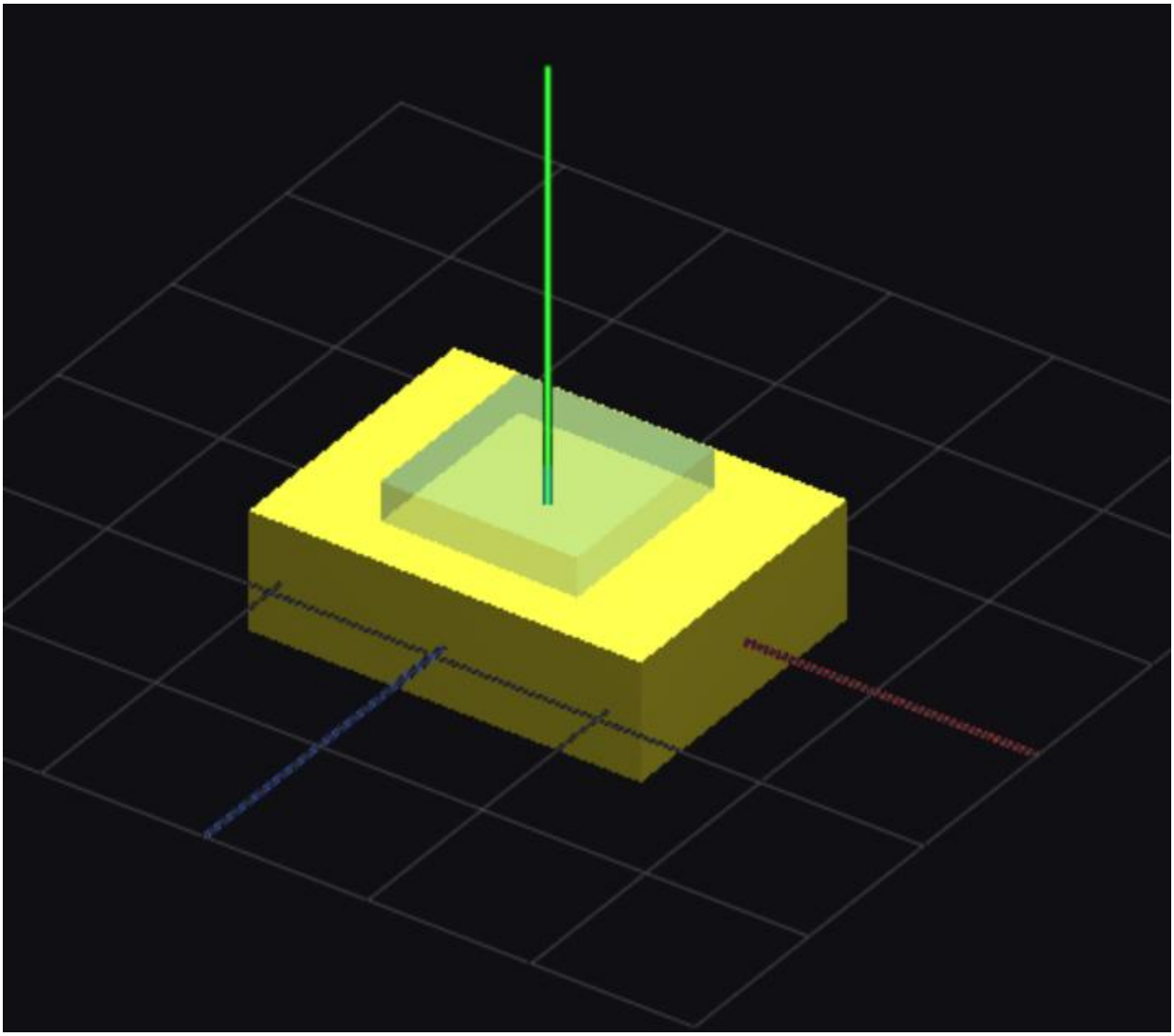
Scaling

```python
import sys
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

SX, SY, SZ = 2.0, 0.6, 1.5

rot_x, rot_y = 20.0, -35.0
zoom = 7.0
ortho_size = 4.0

last_x, last_y = 0, 0
left_down = False
right_down = False


def init_gl():
    glClearColor(0.06, 0.06, 0.08, 1.0)
    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LESS)

    glEnable(GL_LINE_SMOOTH)
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)

    glEnable(GL_BLEND)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)

    glLightfv(GL_LIGHT0, GL_POSITION, (4.0, 6.0, 8.0, 1.0))


def draw_axes(length=3.0):
    glDisable(GL_LIGHTING)
    glLineWidth(2.5)
    glBegin(GL_LINES)

    glColor3f(1.0, 0.2, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(length, 0.0, 0.0)

    glColor3f(0.2, 1.0, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, length, 0.0)

    glColor3f(0.2, 0.4, 1.0)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, 0.0, length)

    glEnd()
    glEnable(GL_LIGHTING)


def draw_ground_grid(size=6, step=1.0):
    glDisable(GL_LIGHTING)
    glLineWidth(1.0)
    glColor4f(0.6, 0.6, 0.6, 0.35)

    half = size * step * 0.5
    glBegin(GL_LINES)
    for i in range(size + 1):
        x = -half + i * step
        glVertex3f(x, 0.0, -half)
        glVertex3f(x, 0.0, +half)

        z = -half + i * step
        glVertex3f(-half, 0.0, z)
        glVertex3f(+half, 0.0, z)
    glEnd()
    glEnable(GL_LIGHTING)


def draw_cube(size=1.2):
    s = size * 0.5
    glBegin(GL_QUADS)

    glNormal3f(0, 0, 1)
    glVertex3f(-s, -s, +s)
    glVertex3f(+s, -s, +s)
    glVertex3f(+s, +s, +s)
    glVertex3f(-s, +s, +s)

    glNormal3f(0, 0, -1)
    glVertex3f(-s, -s, -s)
    glVertex3f(-s, +s, -s)
    glVertex3f(+s, +s, -s)
    glVertex3f(+s, -s, -s)

    glNormal3f(-1, 0, 0)
    glVertex3f(-s, -s, -s)
    glVertex3f(-s, -s, +s)
    glVertex3f(-s, +s, +s)
    glVertex3f(-s, +s, -s)

    glNormal3f(1, 0, 0)
    glVertex3f(+s, -s, -s)
    glVertex3f(+s, +s, -s)
    glVertex3f(+s, +s, +s)
    glVertex3f(+s, -s, +s)
```

```
    glNormal3f(0, 1, 0)
    glVertex3f(-s, +s, -s)
    glVertex3f(-s, +s, +s)
    glVertex3f(+s, +s, +s)
    glVertex3f(+s, +s, -s)

    glNormal3f(0, -1, 0)
    glVertex3f(-s, -s, -s)
    glVertex3f(+s, -s, -s)
    glVertex3f(+s, -s, +s)
    glVertex3f(-s, -s, +s)

    glEnd()


def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    gluLookAt(0.0, 2.2, zoom, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0)

    glRotatef(rot_x, 1.0, 0.0, 0.0)
    glRotatef(rot_y, 0.0, 1.0, 0.0)

    draw_ground_grid()
    draw_axes()

    glPushMatrix()
    glScalef(SX, SY, SZ)
    glColor4f(0.95, 0.90, 0.20, 1.0)
    draw_cube()
    glPopMatrix()

    glDepthMask(GL_FALSE)
    glPushMatrix()
    glColor4f(0.25, 0.70, 1.00, 0.28)
    draw_cube()
    glPopMatrix()
    glDepthMask(GL_TRUE)

    glutSwapBuffers()


def reshape(w, h):
    global ortho_size
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    aspect = w / float(h)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()

    if aspect >= 1.0:
        glOrtho(-ortho_size * aspect, ortho_size * aspect, -ortho_size, ortho_size, 0.1, 100.0)
    else:
        glOrtho(-ortho_size, ortho_size, -ortho_size / aspect, ortho_size / aspect, 0.1, 100.0)

    glMatrixMode(GL_MODELVIEW)


def mouse(button, state, x, y):
    global left_down, right_down, last_x, last_y
    if button == GLUT_LEFT_BUTTON:
        left_down = (state == GLUT_DOWN)
    elif button == GLUT_RIGHT_BUTTON:
        right_down = (state == GLUT_DOWN)
    last_x, last_y = x, y


def motion(x, y):
    global rot_x, rot_y, ortho_size, last_x, last_y
    dx = x - last_x
    dy = y - last_y

    if left_down:
        rot_y += dx * 0.5
        rot_x += dy * 0.5
        rot_x = max(-89.0, min(89.0, rot_x))
    elif right_down:
        ortho_size += dy * 0.02
        ortho_size = max(1.2, min(12.0, ortho_size))
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)

    last_x, last_y = x, y
    glutPostRedisplay()
```

```
def keyboard(key, x, y):
    global rot_x, rot_y, zoom, ortho_size
    k = key.decode("utf-8").lower()
    if k == "r":
        rot_x, rot_y = 20.0, -35.0
        zoom = 7.0
        ortho_size = 4.0
        w = glutGet(GLUT_WINDOW_WIDTH)
        h = glutGet(GLUT_WINDOW_HEIGHT)
        reshape(w, h)
        glutPostRedisplay()
    elif k == "q" or ord(key) == 27:
        sys.exit(0)


def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(900, 650)
    glutCreateWindow(b"3D Scaling: Original (Translucent) vs Scaled (Solid) [ORTHO]")

    init_gl()

    glutDisplayFunc(display)
    glutReshapeFunc(reshape)
    glutMouseFunc(mouse)
    glutMotionFunc(motion)
    glutKeyboardFunc(keyboard)

    glutMainLoop()


if __name__ == "__main__":
    main()
```

Output:



2. Implement the Perspective Projection

perspective > combined-3D.py > ...

```python
import sys
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

# Translation
TX, TY, TZ = 1.4, 0.6, -0.3

# Rotation angles (degrees)
RX, RY, RZ = 35.0, 25.0, 60.0

# Scaling
SX, SY, SZ = 1.7, 0.7, 1.2

# Shear factors:
shXY, shXZ = 0.55, 0.15
shYX, shYZ = 0.00, 0.30
shZX, shZY = 0.00, 0.00

rot_x, rot_y = 20.0, -35.0
zoom = 7.0
last_x, last_y = 0, 0
left_down = False
right_down = False


def init_gl():
    glClearColor(0.06, 0.06, 0.08, 1.0)
    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LESS)

    glEnable(GL_LINE_SMOOTH)
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)

    glEnable(GL_BLEND)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)
    glLightfv(GL_LIGHT0, GL_POSITION, (4.0, 6.0, 8.0, 1.0))


def draw_axes(length=3.0):
    glDisable(GL_LIGHTING)
    glLineWidth(2.5)
    glBegin(GL_LINES)

    glColor3f(1.0, 0.2, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(length, 0.0, 0.0)

    glColor3f(0.2, 1.0, 0.2)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, length, 0.0)

    glColor3f(0.2, 0.4, 1.0)
    glVertex3f(0.0, 0.0, 0.0)
    glVertex3f(0.0, 0.0, length)

    glEnd()
    glEnable(GL_LIGHTING)


def draw_ground_grid(size=6, step=1.0):
    glDisable(GL_LIGHTING)
    glLineWidth(1.0)
    glColor4f(0.6, 0.6, 0.6, 0.35)

    half = size * step * 0.5
    glBegin(GL_LINES)
    for i in range(size + 1):
        x = -half + i * step
        glVertex3f(x, 0.0, -half)
        glVertex3f(x, 0.0, +half)

        z = -half + i * step
        glVertex3f(-half, 0.0, z)
        glVertex3f(+half, 0.0, z)
    glEnd()
    glEnable(GL_LIGHTING)
```

```python
def draw_cube(size=1.2):
    s = size * 0.5
    glBegin(GL_QUADS)

    glNormal3f(0, 0, 1)
    glVertex3f(-s, -s, +s); glVertex3f(+s, -s, +s); glVertex3f(+s, +s, +s); glVertex3f(-s, +s, +s)

    glNormal3f(0, 0, -1)
    glVertex3f(-s, -s, -s); glVertex3f(-s, +s, -s); glVertex3f(+s, +s, -s); glVertex3f(+s, -s, -s)

    glNormal3f(-1, 0, 0)
    glVertex3f(-s, -s, -s); glVertex3f(-s, -s, +s); glVertex3f(-s, +s, +s); glVertex3f(-s, +s, -s)

    glNormal3f(1, 0, 0)
    glVertex3f(+s, -s, -s); glVertex3f(+s, +s, -s); glVertex3f(+s, +s, +s); glVertex3f(+s, -s, +s)

    glNormal3f(0, 1, 0)
    glVertex3f(-s, +s, -s); glVertex3f(-s, +s, +s); glVertex3f(+s, +s, +s); glVertex3f(+s, +s, -s)

    glNormal3f(0, -1, 0)
    glVertex3f(-s, -s, -s); glVertex3f(+s, -s, -s); glVertex3f(+s, -s, +s); glVertex3f(-s, -s, +s)

    glEnd()


def apply_shear_matrix():
    """
    Row-major (for reference):
      [ 1   shXY shXZ 0 ]
      [ shYX 1   shYZ 0 ]
      [ shZX shZY 1   0 ]
      [ 0    0    0   1 ]

    OpenGL expects column-major array.
    """
    m_col_major = [
        1.0,  shYX, shZX, 0.0,
        shXY, 1.0,  shZY, 0.0,
        shXZ, shYZ, 1.0,  0.0,
        0.0,  0.0,  0.0,  1.0
    ]
    glMultMatrixf(m_col_major)


def apply_combined_transform():
    """
    Effective on vertices: v' = T * Rz * Ry * Rx * Sh * S * v

    (Because OpenGL composes transforms into the current matrix; order matters.)
    """
    glTranslatef(TX, TY, TZ)

    glRotatef(RZ, 0.0, 0.0, 1.0)
    glRotatef(RY, 0.0, 1.0, 0.0)
    glRotatef(RX, 1.0, 0.0, 0.0)

    apply_shear_matrix()

    glScalef(SX, SY, SZ)
```

```python
def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    gluLookAt(
        0.0, 2.2, zoom,
        0.0, 0.5, 0.0,
        0.0, 1.0, 0.0
    )

    # Scene rotation (mouse)
    glRotatef(rot_x, 1.0, 0.0, 0.0)
    glRotatef(rot_y, 0.0, 1.0, 0.0)

    draw_ground_grid()
    draw_axes()

    # Solid cube with combined transform
    glPushMatrix()
    apply_combined_transform()
    glColor4f(0.95, 0.55, 0.20, 1.0)
    draw_cube()
    glPopMatrix()

    # Translucent original cube
    glDepthMask(GL_FALSE)
    glPushMatrix()
    glColor4f(0.25, 0.70, 1.00, 0.28)
    draw_cube()
    glPopMatrix()
    glDepthMask(GL_TRUE)

    glutSwapBuffers()


def reshape(w, h):
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(60.0, w / float(h), 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)


def mouse(button, state, x, y):
    global left_down, right_down, last_x, last_y
    if button == GLUT_LEFT_BUTTON:
        left_down = (state == GLUT_DOWN)
    elif button == GLUT_RIGHT_BUTTON:
        right_down = (state == GLUT_DOWN)
    last_x, last_y = x, y


def motion(x, y):
    global rot_x, rot_y, zoom, last_x, last_y
    dx = x - last_x
    dy = y - last_y

    if left_down:
        rot_y += dx * 0.5
        rot_x += dy * 0.5
        rot_x = max(-89.0, min(89.0, rot_x))
    elif right_down:
        zoom += dy * 0.03
        zoom = max(2.5, min(20.0, zoom))

    last_x, last_y = x, y
    glutPostRedisplay()


def keyboard(key, x, y):
    global rot_x, rot_y, zoom
    k = key.decode("utf-8").lower()
    if k == "r":
        rot_x, rot_y = 20.0, -35.0
        zoom = 7.0
        glutPostRedisplay()
    elif k == "q" or ord(key) == 27:
        sys.exit(0)
```

```
def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(900, 650)
    glutCreateWindow(b"3D Combined Transform: T + R + Shear + Scale")

    init_gl()

    print("Combined Transform Parameters")
    print("  Translation (TX, TY, TZ):", (TX, TY, TZ))
    print("  Rotation    (RX, RY, RZ):", (RX, RY, RZ))
    print("  Scaling     (SX, SY, SZ):", (SX, SY, SZ))
    print("  Shear factors:")
    print("    x' = x + shXY*y + shXZ*z :", (shXY, shXZ))
    print("    y' = y + shYX*x + shYZ*z :", (shYX, shYZ))
    print("    z' = z + shZX*x + shZY*y :", (shZX, shZY))
    print("Mouse: Left-drag rotate, Right-drag zoom | R reset | Q/ESC quit")

    glutDisplayFunc(display)
    glutReshapeFunc(reshape)
    glutMouseFunc(mouse)
    glutMotionFunc(motion)
    glutKeyboardFunc(keyboard)
    glutMainLoop()


if __name__ == "__main__":
    main()
```

Output: