SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN, UTTARAKHAND

COMPUTER GRAPHICS

LABORATORY FILE

(2024-2025)

For

Vth Semester

Submitted To:
Mr. Dinesh
Assistant Professor
[Vth Semester]
School of Computer Science

Submitted By:
Akshat Negi
500106533(SAP ID)
R2142220414(Roll No.)
B.Tech. CSF (Batch-1)

LAB EXPERIMENT – 6

Basic 2D & 3D Transformations

# Perform all the experiment for 3-D transformation.

# Take the following values as input from user: Theta (angle of rotation), translation factor, scaling factor and other values. Make necessary assumptions.

a. Write an interactive program for following basic transformation.
b. Translation
c. Rotation
d. Scaling
e. Reflection about axis.
f. Reflection about a line Y=mX+c and aX+bY+c=0.
g. Shear about an edge and about a vertex.

```
#include <GL/freeglut.h>
#include <iostream>
#include <cmath>

float angle = 0.0f;        // Rotation angle
float tx = 0.0f, ty = 0.0f; // Translation factors
float sx = 1.0f, sy = 1.0f; // Scaling factors
float shearX = 0.0f, shearY = 0.0f; // Shear factors
float reflectionX = 1.0f, reflectionY = 1.0f; // Reflection factors

void drawSquare() {
    glBegin(GL_LINE_LOOP);
    glVertex2f(-0.5f, -0.5f);
    glVertex2f(0.5f, -0.5f);
    glVertex2f(0.5f, 0.5f);
    glVertex2f(-0.5f, 0.5f);
    glEnd();
}

// Apply translation transformation
void translate(float x, float y) {
    glTranslatef(x, y, 0.0f);
}

// Apply rotation transformation
void rotate(float angle) {
    glRotatef(angle, 0.0f, 0.0f, 1.0f);
}

// Apply scaling transformation
void scale(float x, float y) {
    glScalef(x, y, 1.0f);
}

// Apply reflection
void reflect(bool x, bool y) {
    reflectionX = x ? -1.0f : 1.0f;
    reflectionY = y ? -1.0f : 1.0f;
    glScalef(reflectionX, reflectionY, 1.0f);
}
```

```
// Apply shear transformation
void shear(float shx, float shy) {
    GLfloat shearMatrix[16] = {
        1.0f, shx, 0.0f, 0.0f,
        shy, 1.0f, 0.0f, 0.0f,
        0.0f, 0.0f, 1.0f, 0.0f,
        0.0f, 0.0f, 0.0f, 1.0f
    };
    glMultMatrixf(shearMatrix);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();

    // Apply transformations in the order: translate, rotate, scale, reflect, shear
    translate(tx, ty);
    rotate(angle);
    scale(sx, sy);
    reflect(reflectionX < 0, reflectionY < 0);
    shear(shearX, shearY);

    drawSquare();
    glutSwapBuffers();
}

// Handle keyboard input for transformations
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 'r': // Rotate
        std::cout << "Enter rotation angle: ";
        std::cin >> angle;
        break;
    case 't': // Translate
        std::cout << "Enter translation factors (tx ty): ";
        std::cin >> tx >> ty;
        break;
    case 's': // Scale
        std::cout << "Enter scaling factors (sx sy): ";
        std::cin >> sx >> sy;
        break;
    case 'x': // Reflect about X-axis
        reflectionX = -reflectionX;
        break;
    case 'y': // Reflect about Y-axis
        reflectionY = -reflectionY;
        break;
    case 'h': // Shear
        std::cout << "Enter shear factors (shearX shearY): ";
        std::cin >> shearX >> shearY;
        break;
    case 27: // Escape key to exit
        exit(0);
    }
    glutPostRedisplay();
}

void init() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-2.0, 2.0, -2.0, 2.0); // Set up a 2D orthogonal projection
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(600, 600);
    glutCreateWindow("2D Transformations - Akshat Negi");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

Interaction:

Keyboard keys:

r: Prompts for rotation angle.

t: Prompts for translation factors (tx, ty).

s: Prompts for scaling factors (sx, sy).

x, y: Reflects about the X or Y axis, respectively.

h: Prompts for shear factors (shearX, shearY).

Esc: Exits the program.

SAMPLE INPUT

Enter rotation angle:

30

Enter translation factors (tx ty):

1.0 0.5

Enter scaling factors (sx sy):

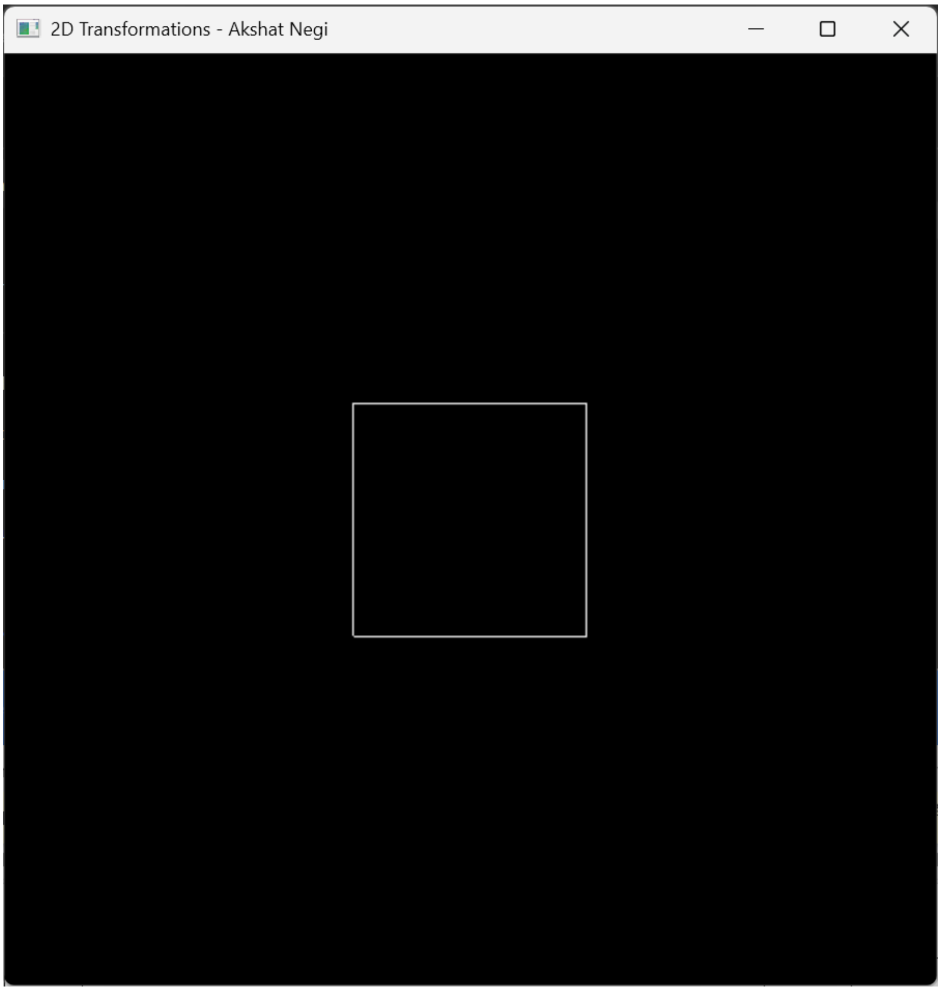
2.0 0.5

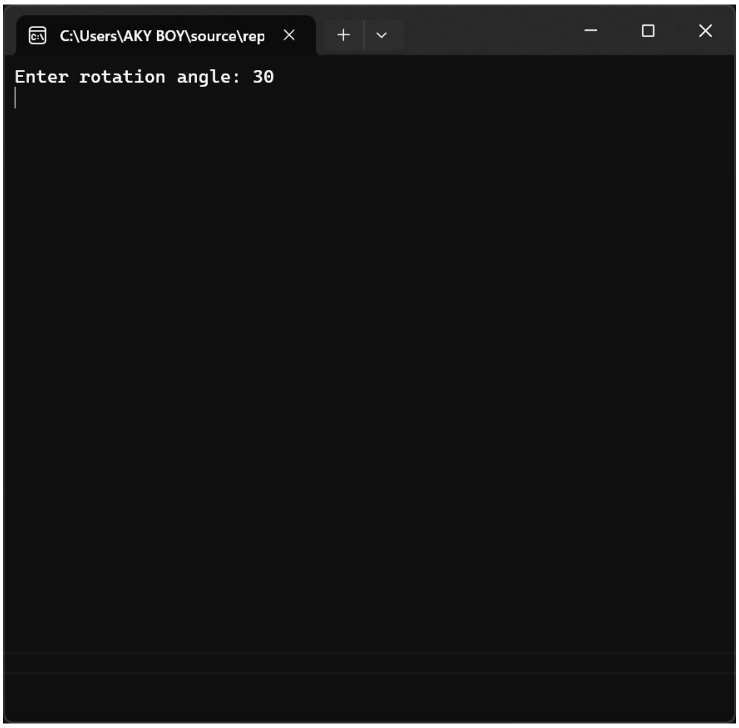
Press 'x' for reflection about X-axis

Press 'h' for shear factors:

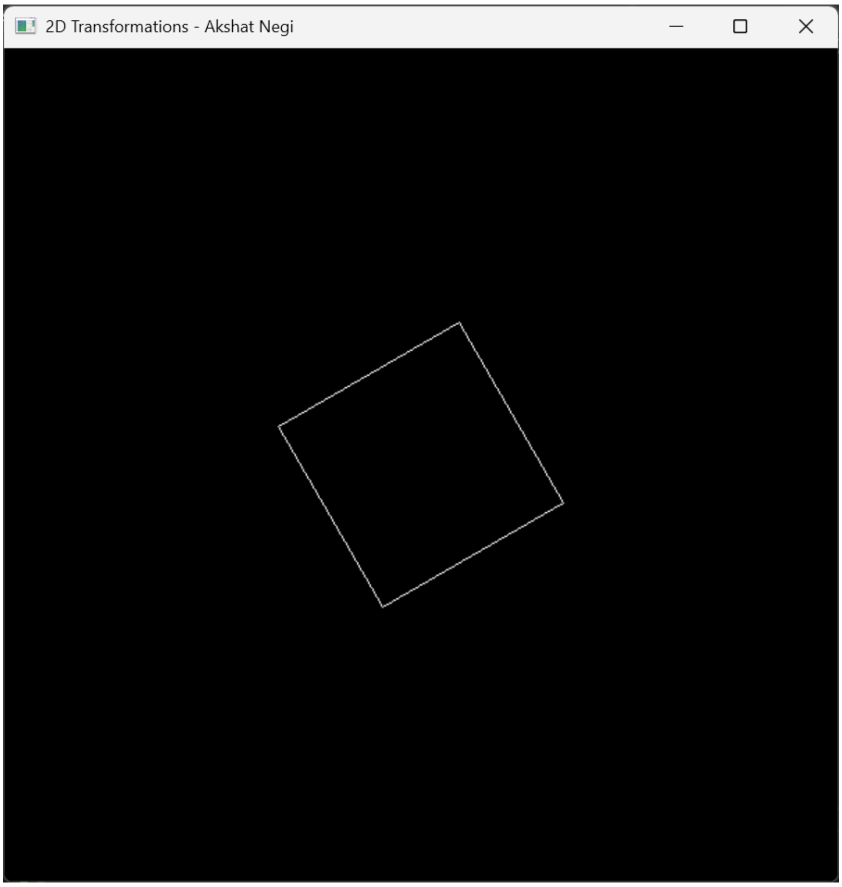
Enter shear factors (shearX shearY):

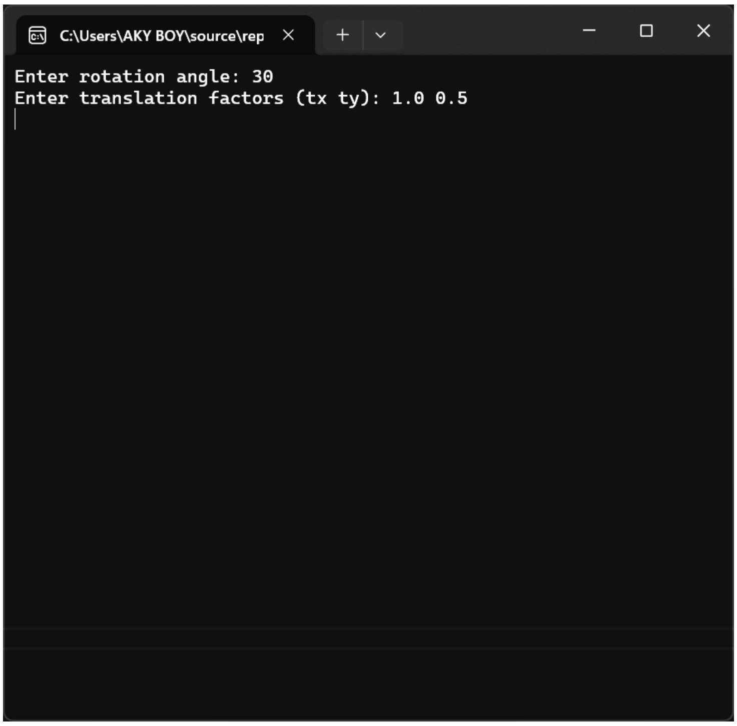
0.2 0.0


2D Transformations - Akshat Negi

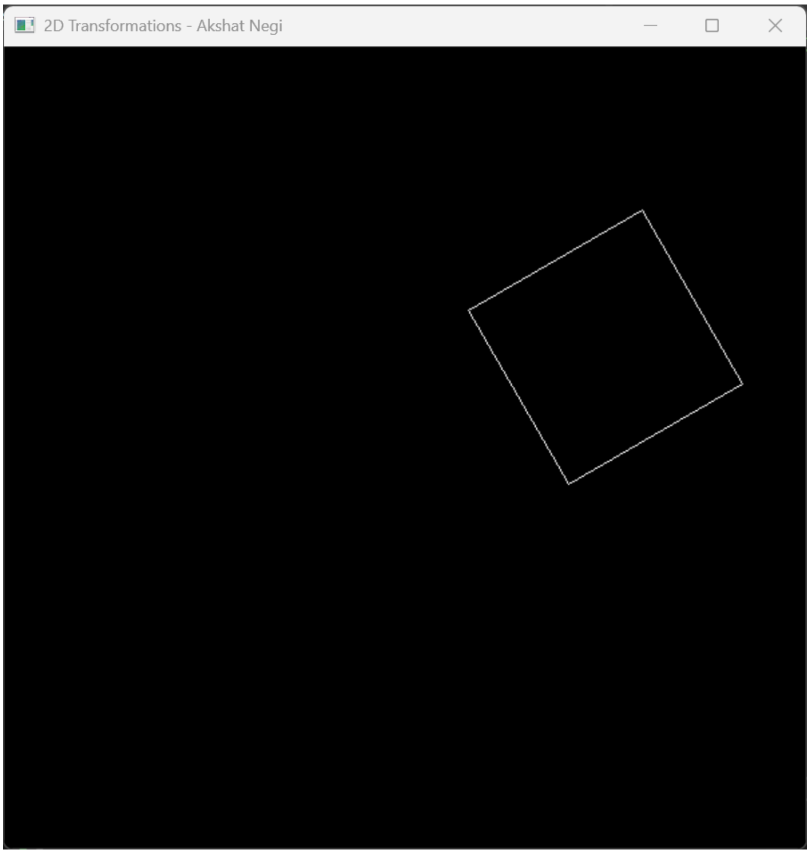


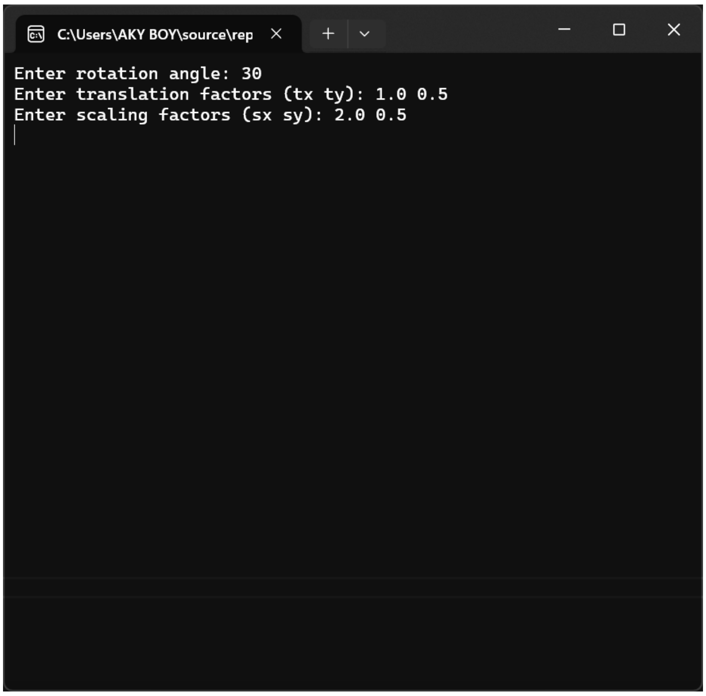
```
Enter rotation angle: 30
```



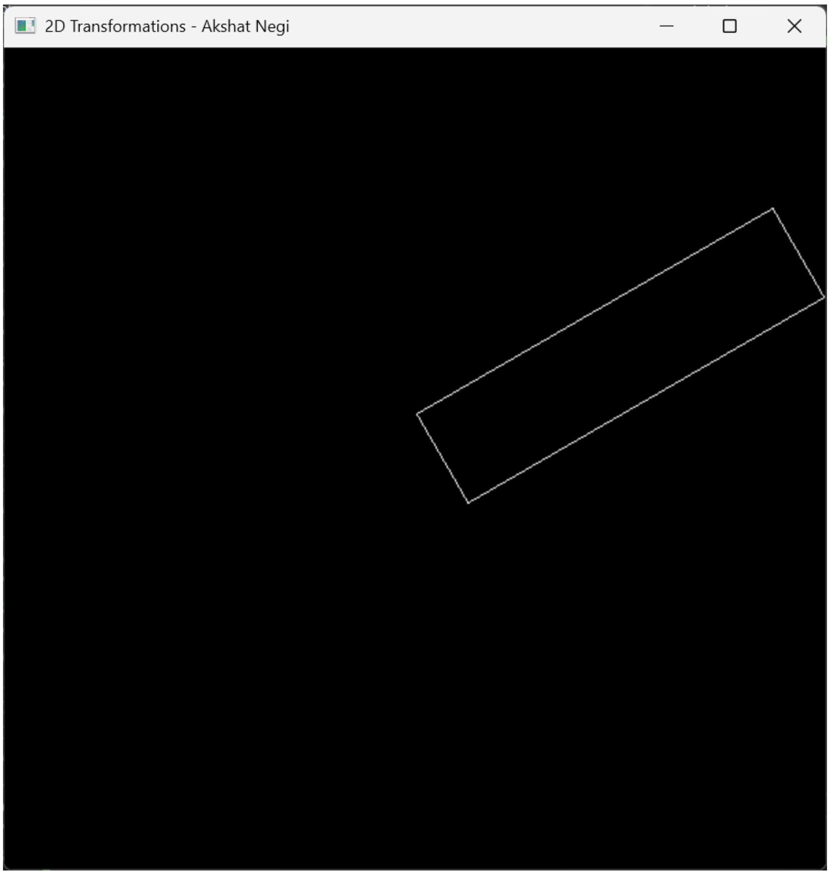


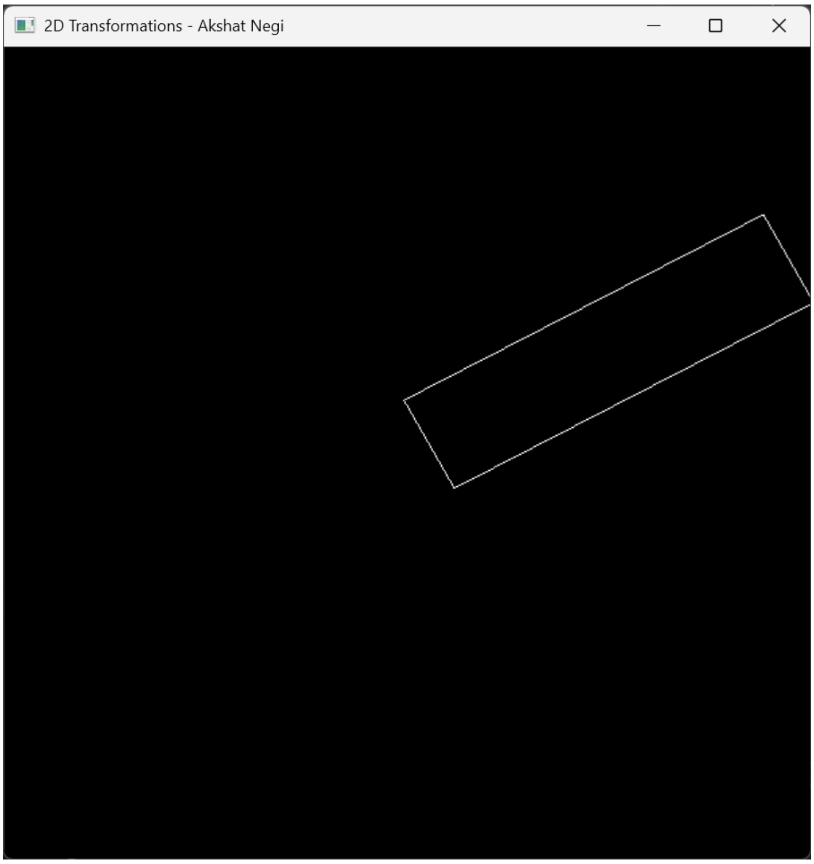
```
Enter rotation angle: 30
Enter translation factors (tx ty): 1.0 0.5
```



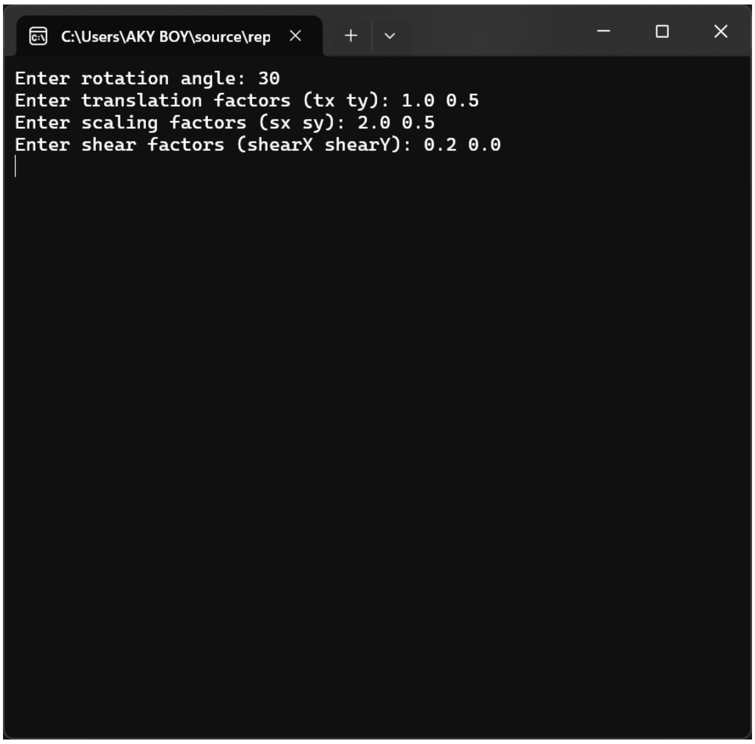


```
Enter rotation angle: 30
Enter translation factors (tx ty): 1.0 0.5
Enter scaling factors (sx sy): 2.0 0.5
```

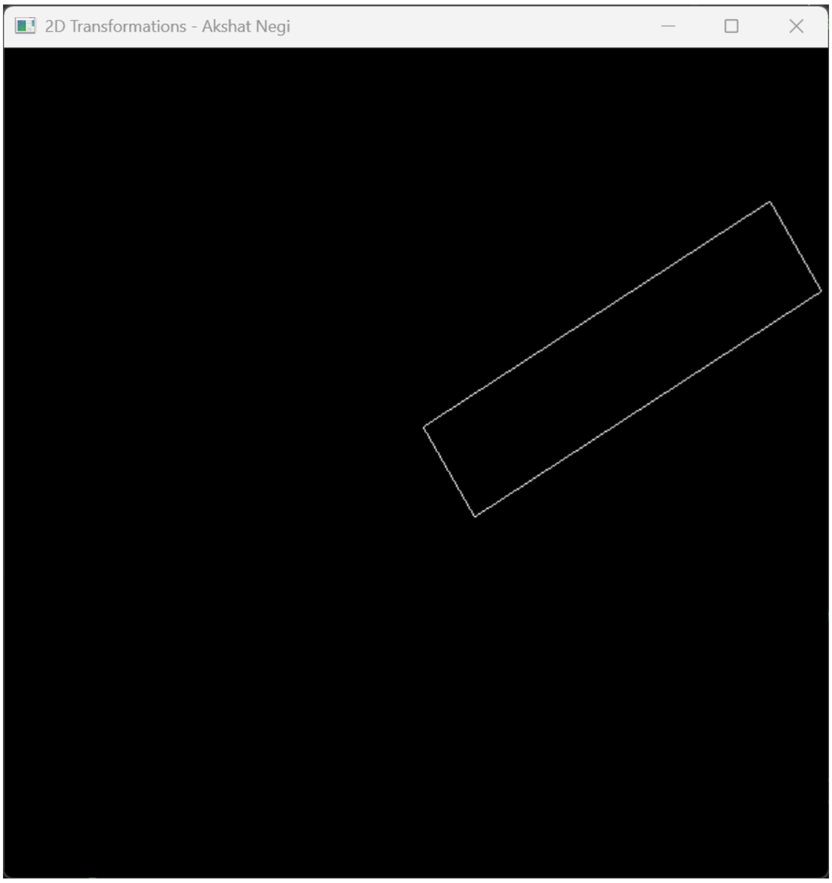




2D Transformations - Akshat Negi



```
Enter rotation angle: 30
Enter translation factors (tx ty): 1.0 0.5
Enter scaling factors (sx sy): 2.0 0.5
Enter shear factors (shearX shearY): 0.2 0.0
```



```
#include <GL/freeglut.h>
#include <iostream>
#include <cmath>

float theta = 0.0f;        // Rotation angle
float tx = 0.0f, ty = 0.0f, tz = 0.0f; // Translation factors
float sx = 1.0f, sy = 1.0f, sz = 1.0f; // Scaling factors
float shearX = 0.0f, shearY = 0.0f, shearZ = 0.0f; // Shear factors
float reflectionX = 1.0f, reflectionY = 1.0f, reflectionZ = 1.0f; // Reflection
factors

void drawCube() {
    glutWireCube(1.0); // Draw a unit cube
}

// Apply translation transformation
void translate(float x, float y, float z) {
    glTranslatef(x, y, z);
}

// Apply rotation transformation
void rotate(float angle, float x, float y, float z) {
    glRotatef(angle, x, y, z);
}

// Apply scaling transformation
void scale(float x, float y, float z) {
    glScalef(x, y, z);
}

// Apply reflection about axis
void reflect(bool x, bool y, bool z) {
    reflectionX = x ? -1.0f : 1.0f;
    reflectionY = y ? -1.0f : 1.0f;
    reflectionZ = z ? -1.0f : 1.0f;
    glScalef(reflectionX, reflectionY, reflectionZ);
}

// Apply shear transformation
void shear(float shx, float shy, float shz) {
    GLfloat shearMatrix[16] = {
        1.0f, shx, 0.0f, 0.0f,
        shy, 1.0f, 0.0f, 0.0f,
        0.0f, shz, 1.0f, 0.0f,
        0.0f, 0.0f, 0.0f, 1.0f
    };
    glMultMatrixf(shearMatrix);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -5.0f); // Position the object

    // Apply transformations in the order: translate, rotate, scale, reflect, shear
    translate(tx, ty, tz);
    rotate(theta, 1.0f, 1.0f, 1.0f);
    scale(sx, sy, sz);
    reflect(reflectionX < 0, reflectionY < 0, reflectionZ < 0);
    shear(shearX, shearY, shearZ);

    drawCube();
    glutSwapBuffers();
}

// Handle keyboard input for transformations
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 'r': // Rotate
        std::cout << "Enter rotation angle: ";
        std::cin >> theta;
```

```
            break;
        case 't': // Translate
            std::cout << "Enter translation factors (tx ty tz): ";
            std::cin >> tx >> ty >> tz;
            break;
        case 's': // Scale
            std::cout << "Enter scaling factors (sx sy sz): ";
            std::cin >> sx >> sy >> sz;
            break;
        case 'x': // Reflect about X-axis
            reflectionX = -reflectionX;
            break;
        case 'y': // Reflect about Y-axis
            reflectionY = -reflectionY;
            break;
        case 'z': // Reflect about Z-axis
            reflectionZ = -reflectionZ;
            break;
        case 'h': // Shear
            std::cout << "Enter shear factors (shearX shearY shearZ): ";
            std::cin >> shearX >> shearY >> shearZ;
            break;
        case 27: // Escape key to exit
            exit(0);
        }
        glutPostRedisplay();
    }

    void init() {
        glEnable(GL_DEPTH_TEST);
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glMatrixMode(GL_PROJECTION);
        gluPerspective(45.0, 1.0, 1.0, 100.0);
        glMatrixMode(GL_MODELVIEW);
    }
```

```
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(600, 600);
    glutCreateWindow("3D Transformations - Akshat Negi");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

Interaction:

Keyboard keys:

r: Prompts for rotation angle.

t: Prompts for translation factors (tx, ty, tz).

s: Prompts for scaling factors (sx, sy, sz).

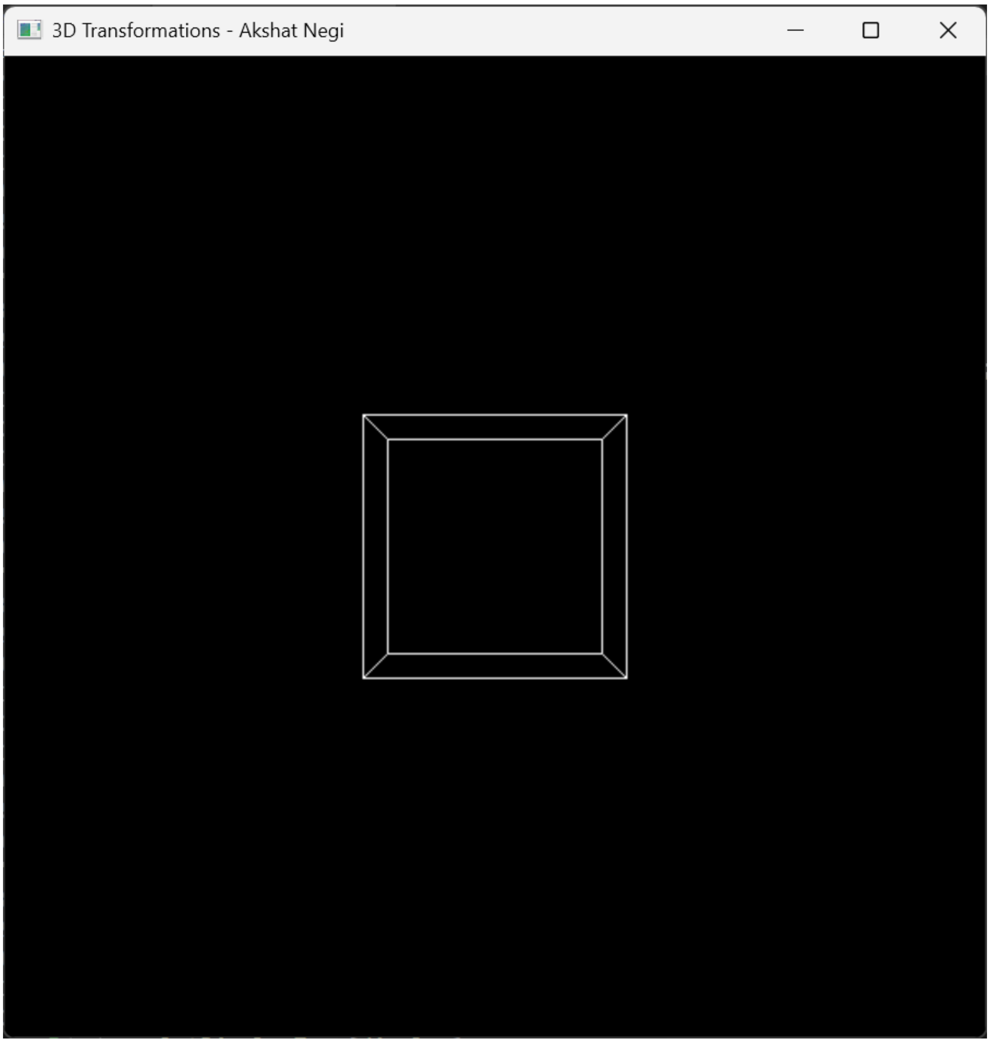
x, y, z: Reflects about X, Y, or Z axes.

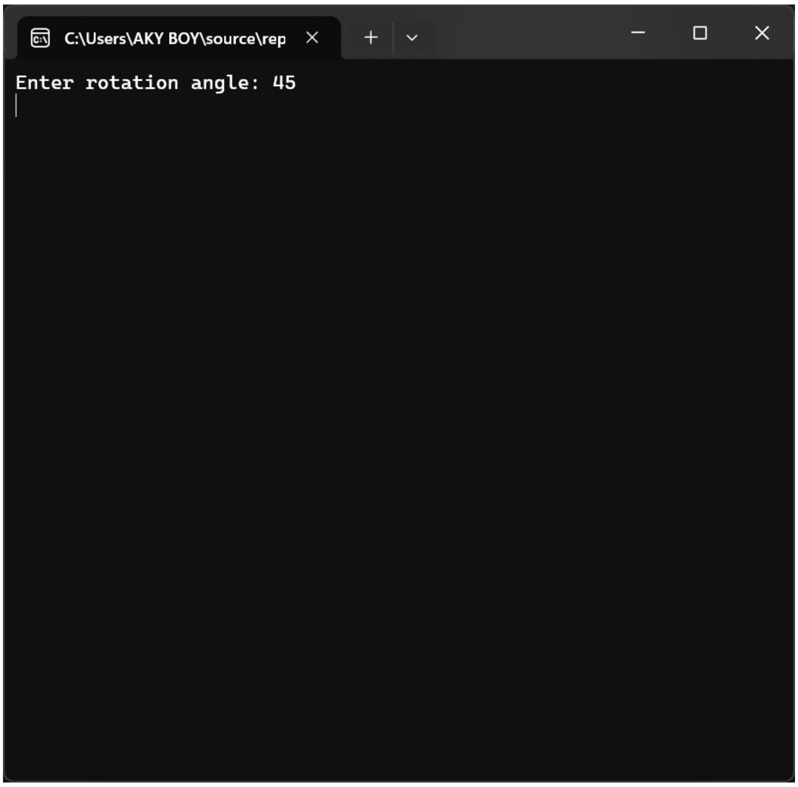
h: Prompts for shear factors (shearX, shearY, shearZ).

Esc: Exits the program.

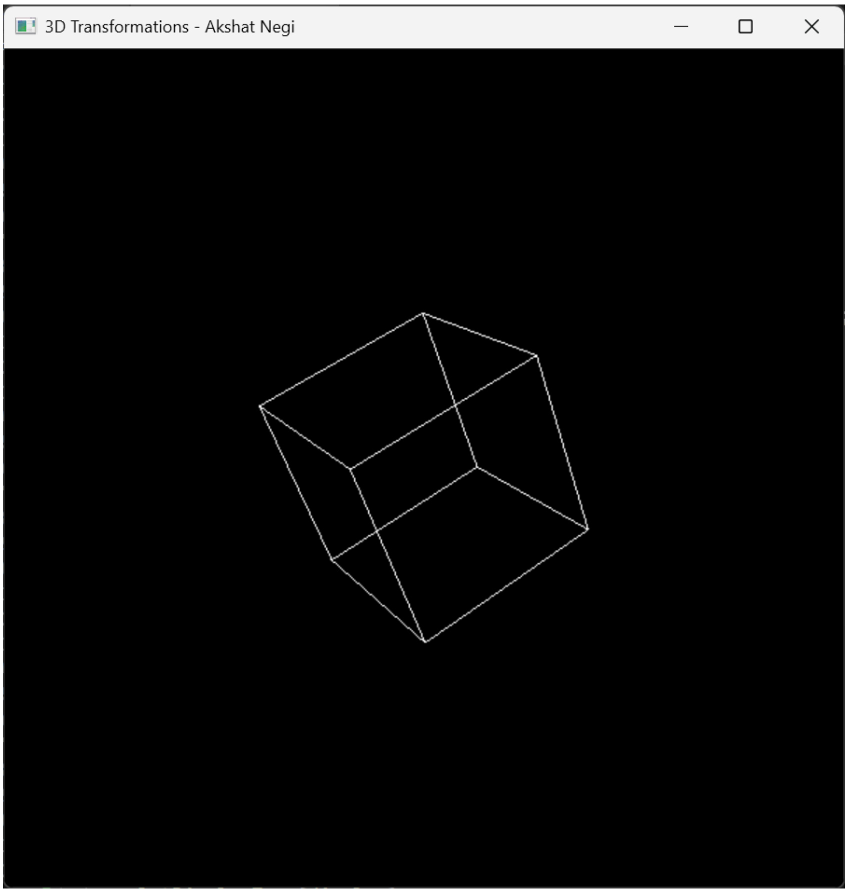
SAMPLE INPUT

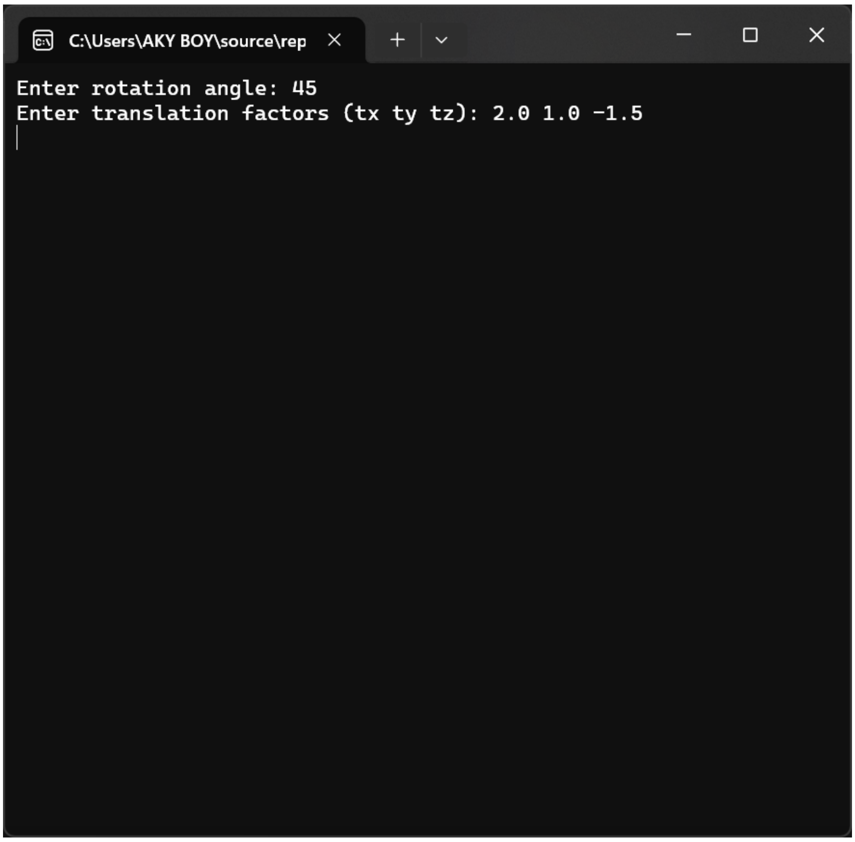
```
Enter rotation angle:
45
Enter translation factors (tx ty tz):
2.0 1.0 -1.5
Enter scaling factors (sx sy sz):
1.5 0.5 2.0
Press 'x' for reflection about X-axis
Press 'h' for shear factors:
Enter shear factors (shearX shearY shearZ):
0.5 0.0 1.0
```


3D Transformations - Akshat Negi

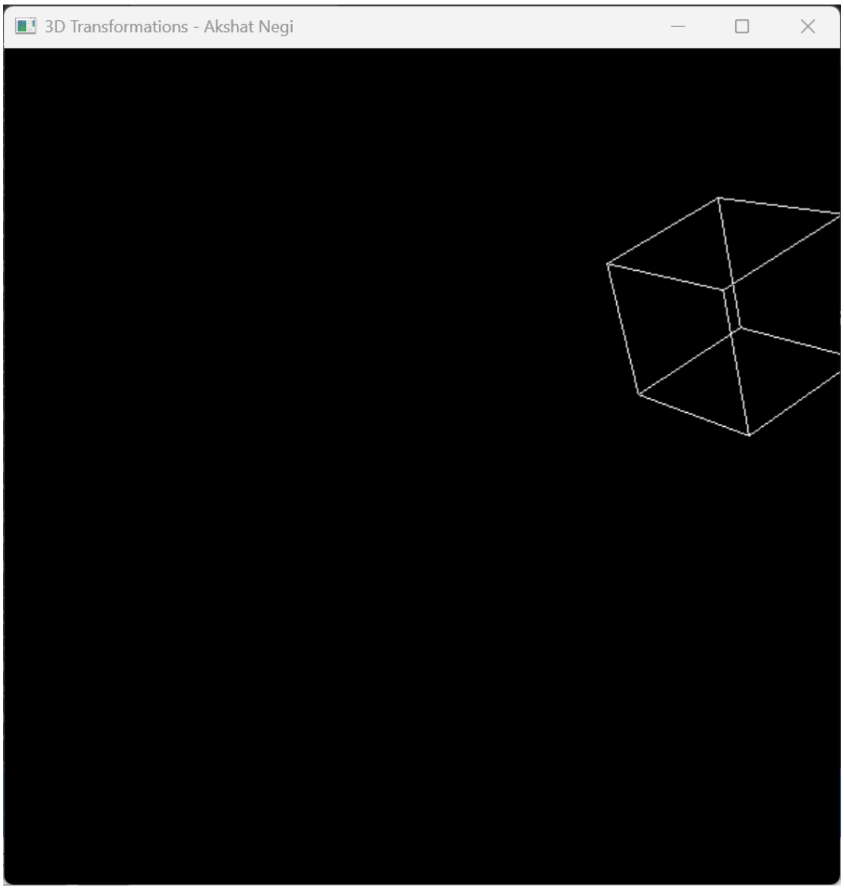


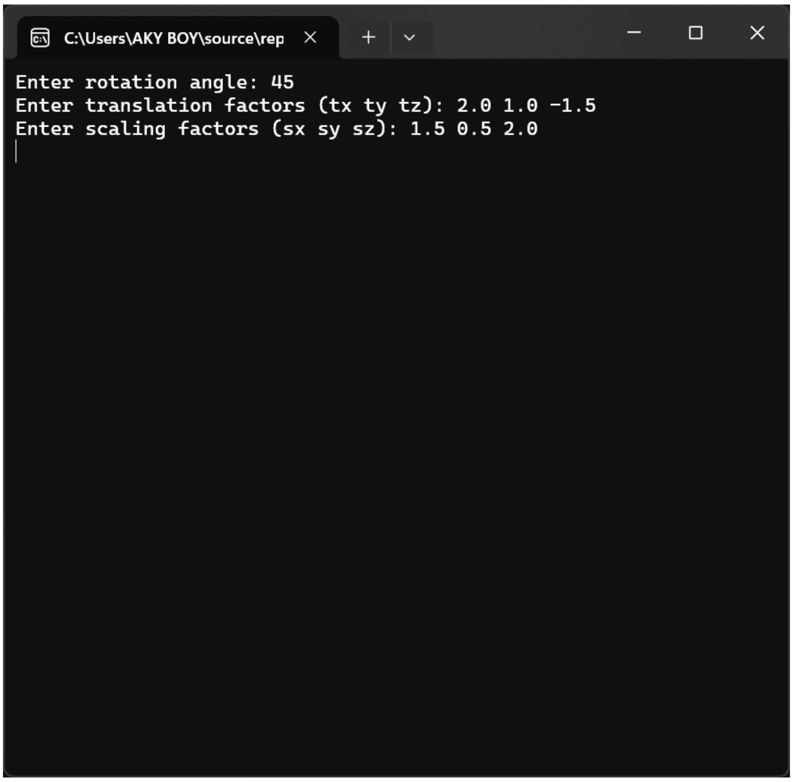
```
Enter rotation angle: 45
```



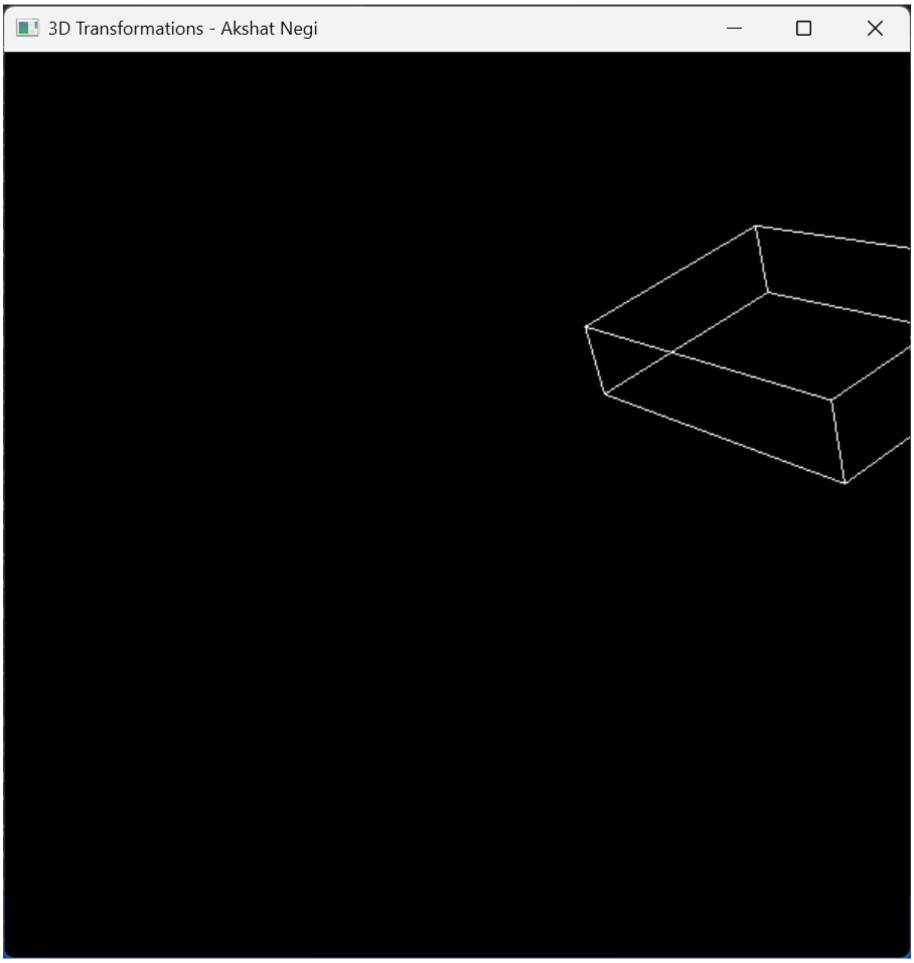


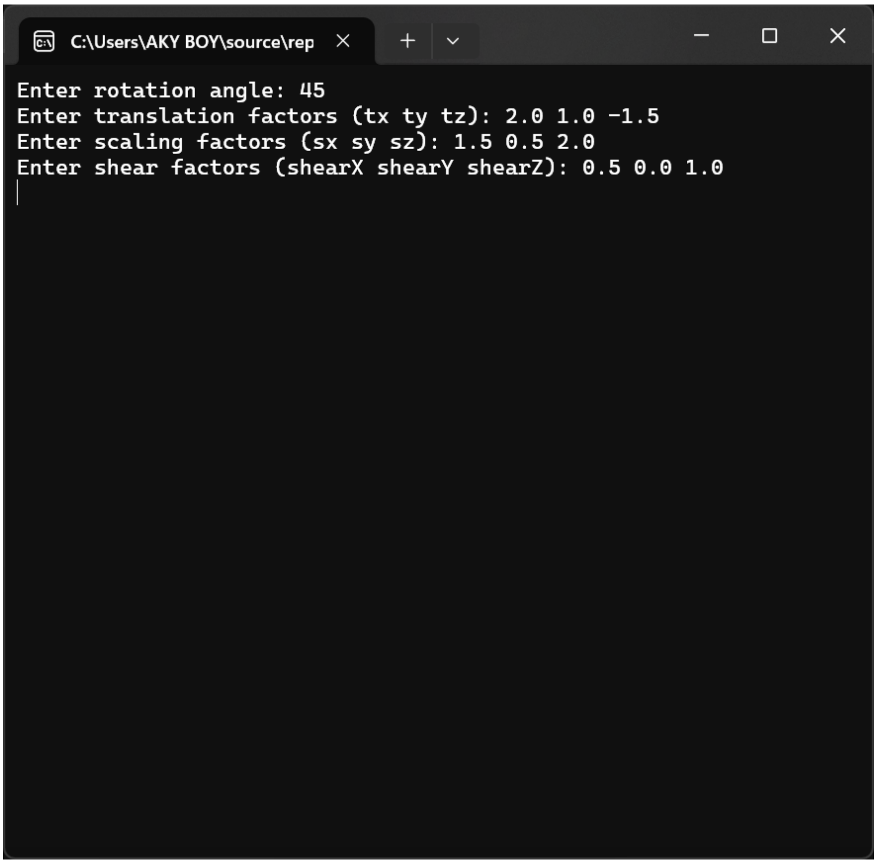
```
Enter rotation angle: 45
Enter translation factors (tx ty tz): 2.0 1.0 -1.5
```



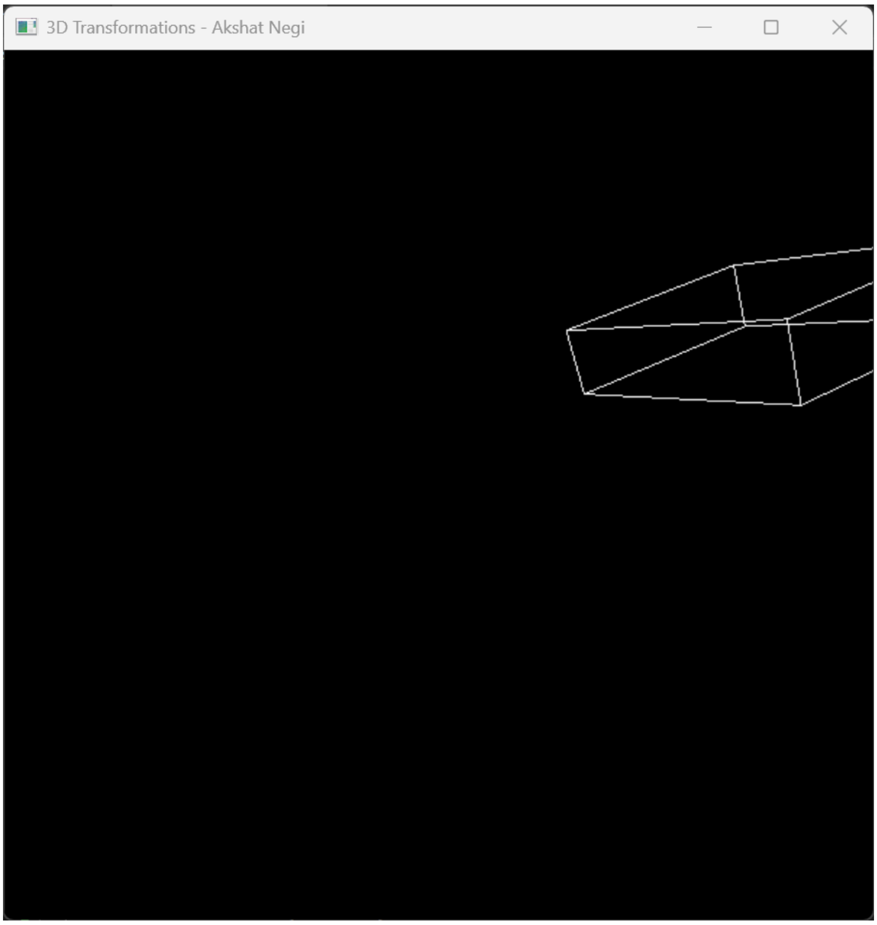


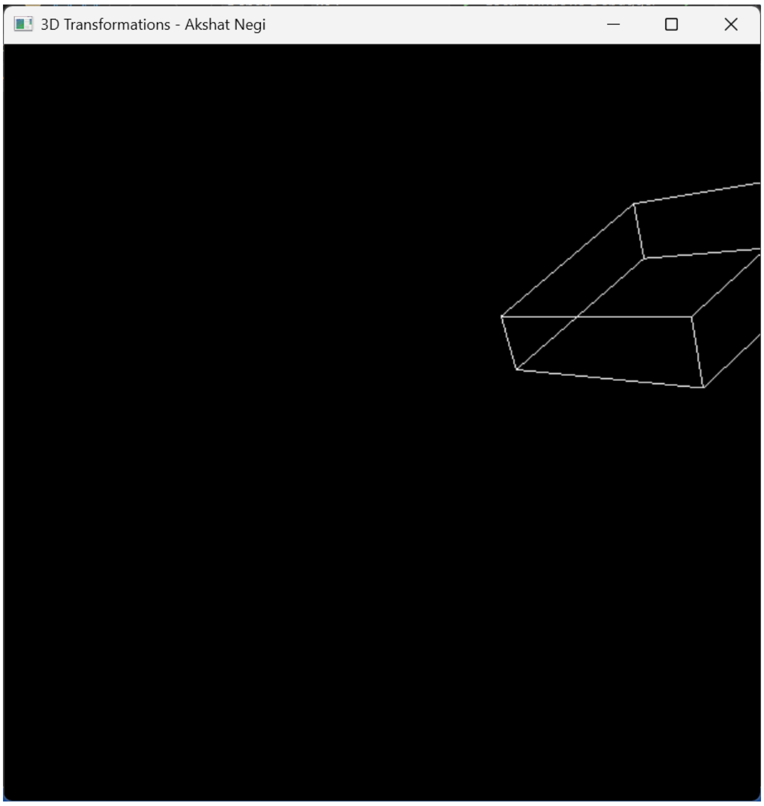
```
Enter rotation angle: 45
Enter translation factors (tx ty tz): 2.0 1.0 -1.5
Enter scaling factors (sx sy sz): 1.5 0.5 2.0
```



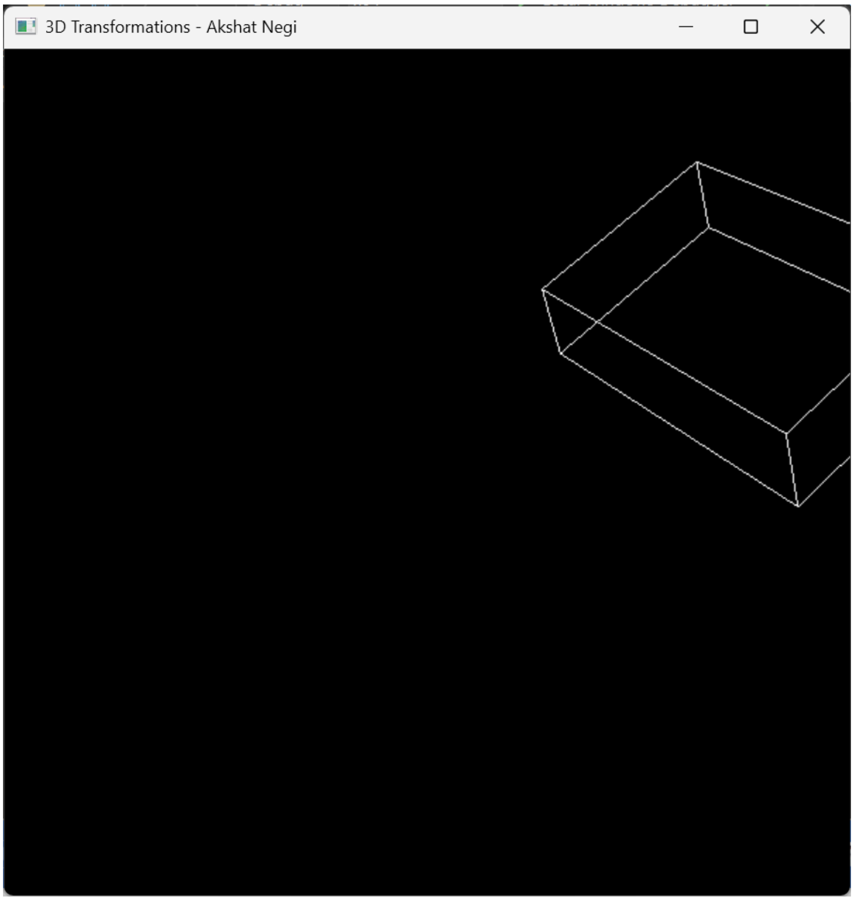


```
Enter rotation angle: 45
Enter translation factors (tx ty tz): 2.0 1.0 -1.5
Enter scaling factors (sx sy sz): 1.5 0.5 2.0
Enter shear factors (shearX shearY shearZ): 0.5 0.0 1.0
```

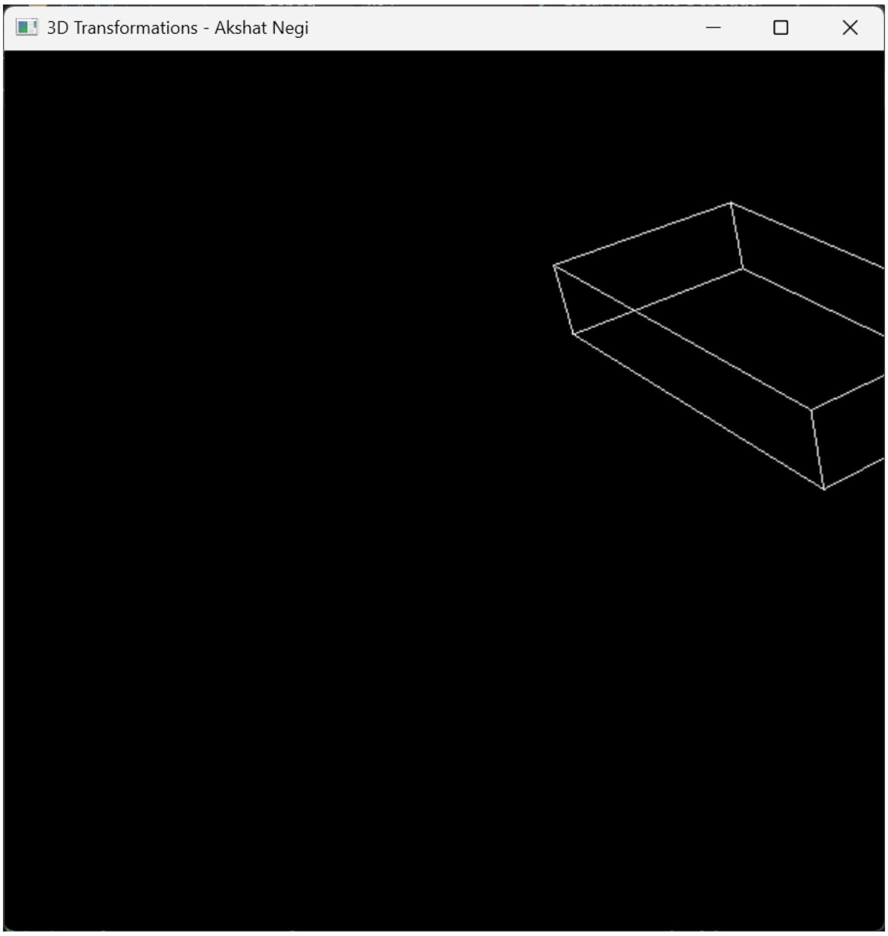




3D Transformations - Akshat Negi



3D Transformations - Akshat Negi


3D Transformations - Akshat Negi