Lab Report

Week 5

Jitendra Kumar , 1401CS19

20/02/2017

Title

- Implement the 3D Transformations for any primitive.
  1). OpenGL
  2). MatLab

Procedure

OpenGL

1). Choose any primitive and apply the 3D transformations as below :

- Create a C file and name it as transformation_3_D.c
- Following is the final code :

```
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>
#include <stdlib.h>
#define PI 3.14159265

int flag=0, count=0; int i = 0 ; int j = 0 ; int k = 0 ;

double input_pts[8][3] = {{0,0,0},{50,0,0},{50,50,0},{0,50,0},{0,0,10},{50,0,50},{50,50,50},{0,10,10}};
double final_pts[8][3];
double trans_matrix[4][4];

double x = 0;
void displayPolygon()
{

        glClear(GL_COLOR_BUFFER_BIT);
        glLineWidth(3);
        glBegin(GL_LINES);
                glColor3f(1.0f, 1.0f, 1.0f);
                glVertex3f(0.0f,400.0f,0.0f);
                glVertex3f(0.0f,-400.0f,0.0f);
                glVertex3f(400.0f,0.0f,0.0f);
                glVertex3f(-400.0f,0.0f,0.0f);
                glVertex3f(0.0f,0.0f,400.0f);
                glVertex3f(0.0f,0.0f,-400.0f);
        glEnd();

        glBegin(GL_QUADS);

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

        for(i = 0 ; i < 8 ; i++)
        {
                 glVertex3f(input_pts[i][0]/100.0, input_pts[i][1]/100.0, input_pts[i][2]/100.0);
        }
        glEnd();

        glBegin(GL_QUADS);

        glColor3f(0.0f,1.0f,0.0f);
        for(i = 0 ; i < 8 ; i++)
        {
                 glVertex3f(final_pts[i][0]/100.0, final_pts[i][1]/100.0, final_pts[i][2]/100.0);
        }

        glEnd();

        glFlush();
        glutSwapBuffers();
}

void matrix_multiplication()
{
        int i = 0 , j = 0 , k = 0 ;
        double a = 0 ; double b = 0 ; double c = 0;

        for(i = 0 ; i < 8 ; i++)
        {
                a = trans_matrix[0][0]*input_pts[i][0] + trans_matrix[0][1]*input_pts[i][1]
                                + trans_matrix[0][2]*input_pts[i][2] + trans_matrix[0][3];

                b = trans_matrix[1][0]*input_pts[i][0] + trans_matrix[1][1]*input_pts[i][1]
                                + trans_matrix[1][2]*input_pts[i][2] + trans_matrix[1][3];

                c = trans_matrix[2][0]*input_pts[i][0] + trans_matrix[2][1]*input_pts[i][1]
                                + trans_matrix[2][2]*input_pts[i][2] + trans_matrix[2][3];

                final_pts[i][0]=a;
                final_pts[i][1]=b;
                final_pts[i][2]=c;
        }
}
void translate(double x , double y, double z)
{
        trans_matrix[0][0] = 1;
        trans_matrix[0][1] = 0;
        trans_matrix[0][2] = 0;
        trans_matrix[0][3] = x;

        trans_matrix[1][0] = 0;
        trans_matrix[1][1] = 1;
        trans_matrix[1][2] = 0;
        trans_matrix[1][3] = y;

        trans_matrix[2][0] = 0;
        trans_matrix[2][1] = 0;
        trans_matrix[2][2] = 1;
        trans_matrix[2][3] = z;

        trans_matrix[3][0] = 0;
        trans_matrix[3][1] = 0;
        trans_matrix[3][2] = 0;
        trans_matrix[3][3] = 1;
}

void scale_x_y_z(double sx, double sy, double sz)
{
        trans_matrix[0][0] = sx;
        trans_matrix[0][1] = 0;
        trans_matrix[0][2] = 0;
        trans_matrix[0][3] = 0;

        trans_matrix[1][0] = 0;
        trans_matrix[1][1] = sy;
        trans_matrix[1][2] = 0;
        trans_matrix[1][3] = 0;

        trans_matrix[2][0] = 0;
        trans_matrix[2][1] = 0;
        trans_matrix[2][2] = sz;
        trans_matrix[2][3] = 0;

        trans_matrix[3][0] = 0;
        trans_matrix[3][1] = 0;
        trans_matrix[3][2] = 0;
        trans_matrix[3][3] = 1;
}
void reflectAroundXY(void)
{
        trans_matrix[0][0] = 1;
        trans_matrix[0][1] = 0;
        trans_matrix[0][2] = 0;
        trans_matrix[0][3] = 0;

        trans_matrix[1][0] = 0;
        trans_matrix[1][1] = 1;
        trans_matrix[1][2] = 0;
        trans_matrix[1][3] = 0;

        trans_matrix[2][0] = 0;
        trans_matrix[2][1] = 0;
        trans_matrix[2][2] = -1;
        trans_matrix[2][3] = 0;

        trans_matrix[3][0] = 0;
        trans_matrix[3][1] = 0;
        trans_matrix[3][2] = 0;
        trans_matrix[3][3] = 1;
}

void  shear(double shear_factor_x)
{
        trans_matrix[0][0] = 1;
        trans_matrix[0][1] = 0;
        trans_matrix[0][2] = shear_factor_x;
        trans_matrix[0][3] = 0;

        trans_matrix[1][0] = 0;
        trans_matrix[1][1] = 1;
        trans_matrix[1][2] = 0;
        trans_matrix[1][3] = 0;

        trans_matrix[2][0] = 0;
        trans_matrix[2][1] = 0;
        trans_matrix[2][2] = 1;
        trans_matrix[2][3] = 0;

        trans_matrix[3][0] = 0;
        trans_matrix[3][1] = 0;
        trans_matrix[3][2] = 0;
        trans_matrix[3][3] = 1;
}

void rotate(double a)
{
        x = PI / 180 ;
        a = a*x;
        trans_matrix[0][0] = 1;
        trans_matrix[0][1] = 0;
        trans_matrix[0][2] = 0;
        trans_matrix[0][3] = 0;

        trans_matrix[1][0] = 0;
        trans_matrix[1][1] = cos(a);
        trans_matrix[1][2] = -1*sin(a);
        trans_matrix[1][3] = 0;

        trans_matrix[2][0] = 0;
        trans_matrix[2][1] = sin(a);
        trans_matrix[2][2] = cos(a);
        trans_matrix[2][3] = 0;

        trans_matrix[3][0] = 0;
        trans_matrix[3][1] = 0;
        trans_matrix[3][2] = 0;
        trans_matrix[3][3] = 1;
}
void transformPoints()
{
        int choice ;
        printf("\nEnter your choice:\n1. Translation\n2. Scaling\n3. Reflection\n4. Shear\n5. Rotate\n");
        printf("Your choice : ");
        scanf("%d",&choice);

        double x , y , z, scale_factor_x, scale_factor_y, scale_factor_z, shear_factor_x , rotation_angle_x ;
        double shear_factor_y, shear_factor_z;

        if(choice==1)
        {
                printf("Enter translate_x : ");
                scanf("%lf",&x);
                printf("Enter translate_y : ");
                scanf("%lf",&y);
                printf("Enter translate_z : ");
                scanf("%lf",&z);
                translate(x,y,z);
        }

        if(choice==2)
        {
                printf("Enter scale_factor_x : ");
                scanf("%lf",&scale_factor_x);
                printf("Enter scale_factor_y : ");
                scanf("%lf",&scale_factor_y);
                printf("Enter scale_factor_z : ");
                scanf("%lf",&scale_factor_z);
                scale_x_y_z(scale_factor_x,scale_factor_y,scale_factor_z);
        }

        if(choice==3)
        {
                reflectAroundXY();
        }

        if(choice==4)
        {
                printf("Enter  the shear_factor_x :  ");
                scanf("%lf",&shear_factor_x);

                shear(shear_factor_x);
        }

        if(choice==5)
        {
                printf("\nEnter the rotation_angle :  ");
                scanf("%lf",&rotation_angle_x);
                rotate(rotation_angle_x);
        }
}


int main(int argc, char const *argv[])
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGB);
        glutInitWindowSize(800, 800);
        glutInitWindowSize(800, 800);
        transformPoints();
        matrix_multiplication();
        glClearColor(1.0, 1.0, 1.0, 1.0);
        gluOrtho2D(-400, 400, -400, 400);
        glutCreateWindow("\nPolygon transformations");
        glutDisplayFunc(displayPolygon);
        glutMainLoop();
        return 0;
}
```

- Compile and run the executable file in terminal by typing in the following commands :
  (a) *gcc transformation_3_D.c -lGL -lGLU -lglut -ll*
  (b) *./a.out*

MatLab

1). Choose any primitive and apply all 3D transformations :

- Open a new matlab script.
- Following is the final code :

```
function[] = transformations_3_D()

cube = [0 0 0 1; 10 0 0 1; 10 10 0 1; 0 10 0 1; 0 0 0 1;
                0 0 10 1; 10 0 10 1; 10 10 10 1; 0 10 10 1; 0 0 10 1;
                10 0 10 1; 10 0 0 1; 10 10 0 1; 10 10 10 1; 0 10 10 1; 0 10 0 1];


prompt = ("\n\nEnter one of the following options:\n1.Translation\n2.Uniform Scaling\n3.Scaling\n4.Reflection\n5.Rotation\n6.Shee
choice = input(prompt);

trans = eye(4);

switch choice
        case 1
                xt = input("Enter the translation in X:");
                trans(1, 4) =  xt;
                yt = input("Enter the translation in Y:");
                trans(2, 4) =  yt;
                zt = input("Enter the translation in Z:");
                trans(3, 4) =  zt;
        case 2
                s = input("Enter the scale-factor :");
                trans(4, 4) = s;
        case 3
                xs = input("Enter the scale-factor in X:");
                trans(1, 1) =  xs;
                ys = input("Enter the scale-factor in Y:");
                trans(2, 2) =  ys;
                zs = input("Enter the scale-factor in Z:");
                trans(3, 3) =  zs;
        case 4
                disp('Reflection␣wrt␣the␣XY␣Plane␣is:');
                trans(3, 3) =  -1;
        case 5
                theta = input("Enter the angle for rotation about X-Axis:");
                theta = (theta*3.14159265)/180;
                trans(2, 2) = cos(theta);
                trans(2, 3) = -1*sin(theta);
                trans(3, 3) = cos(theta);
                trans(3, 2) = sin(theta);
        case 6
                xsh = input("Enter the sheer in X:");
                ysh = input("Enter the sheer in Y:");
                zsh = input("Enter the sheer in Z:");
                trans(1, 3) = xsh;
                trans(2, 3) = ysh;
                trans(3, 2) = zsh;
        otherwise
                disp('Wrong␣Input.');
end

trans = transpose(trans);
cube3 = cube*trans;
cube3 = cube3*trans(4,4);

axis([-15 15 -15 15 -15 15]);

plot3(cube(:, 1), cube(:, 2), cube(:, 3), 'LineWidth', 2, '-o', 'color', 'r');



hold on;

plot3(cube3(:, 1), cube3(:, 2), cube3(:, 3), 'LineWidth', 2, '-o', 'color', 'g');

%axis([0 20 0 20 0 20]);
```

Output

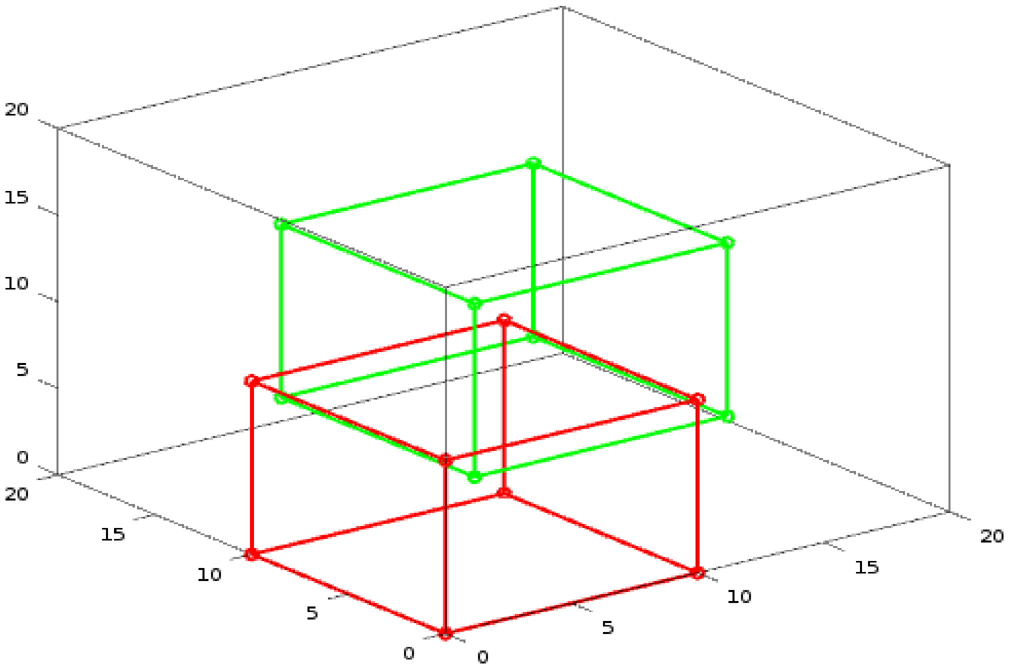
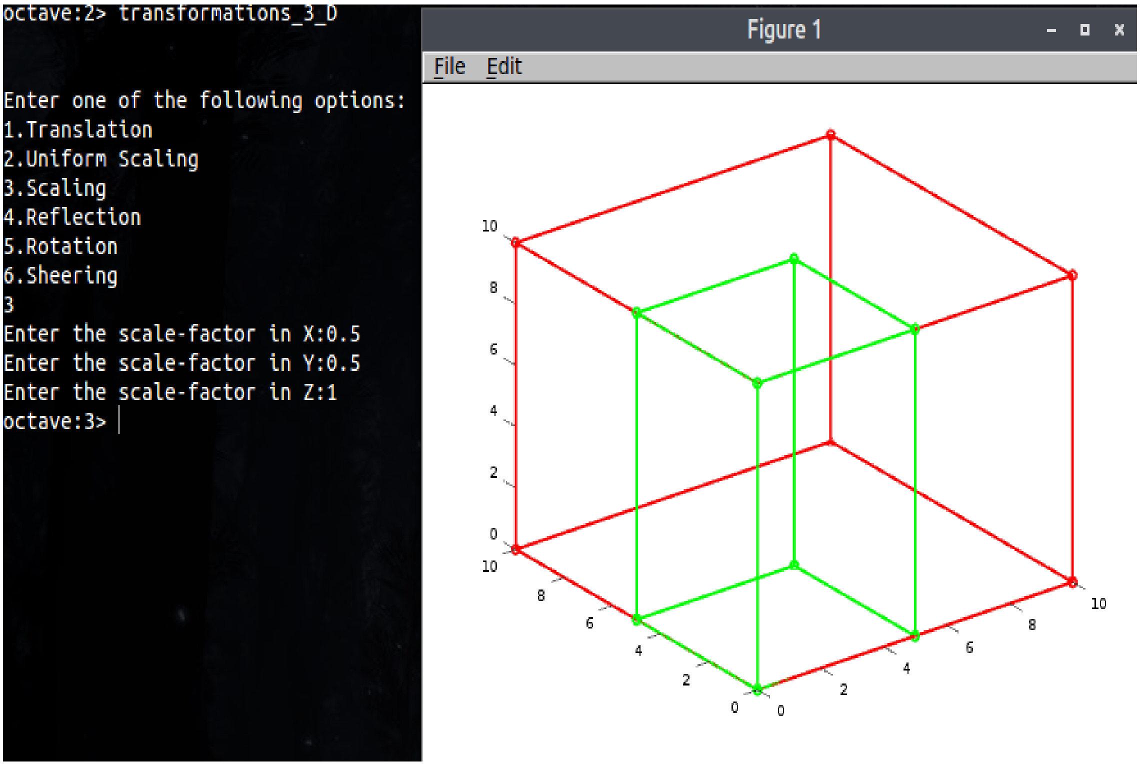


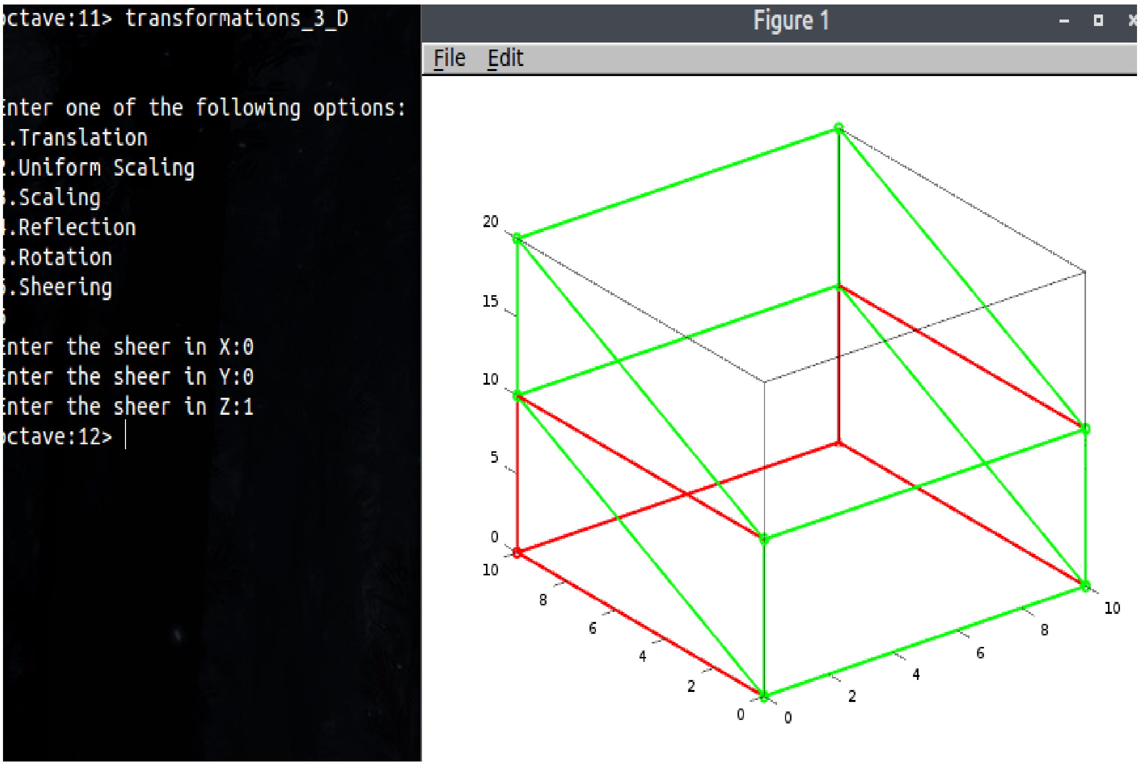
FIGURE 1 – translation



```
octave:2> transformations_3_D

Enter one of the following options:
1.Translation
2.Uniform Scaling
3.Scaling
4.Reflection
5.Rotation
6.Sheering
3
Enter the scale-factor in X:0.5
Enter the scale-factor in Y:0.5
Enter the scale-factor in Z:1
octave:3> |
```

FIGURE 2 – scaling

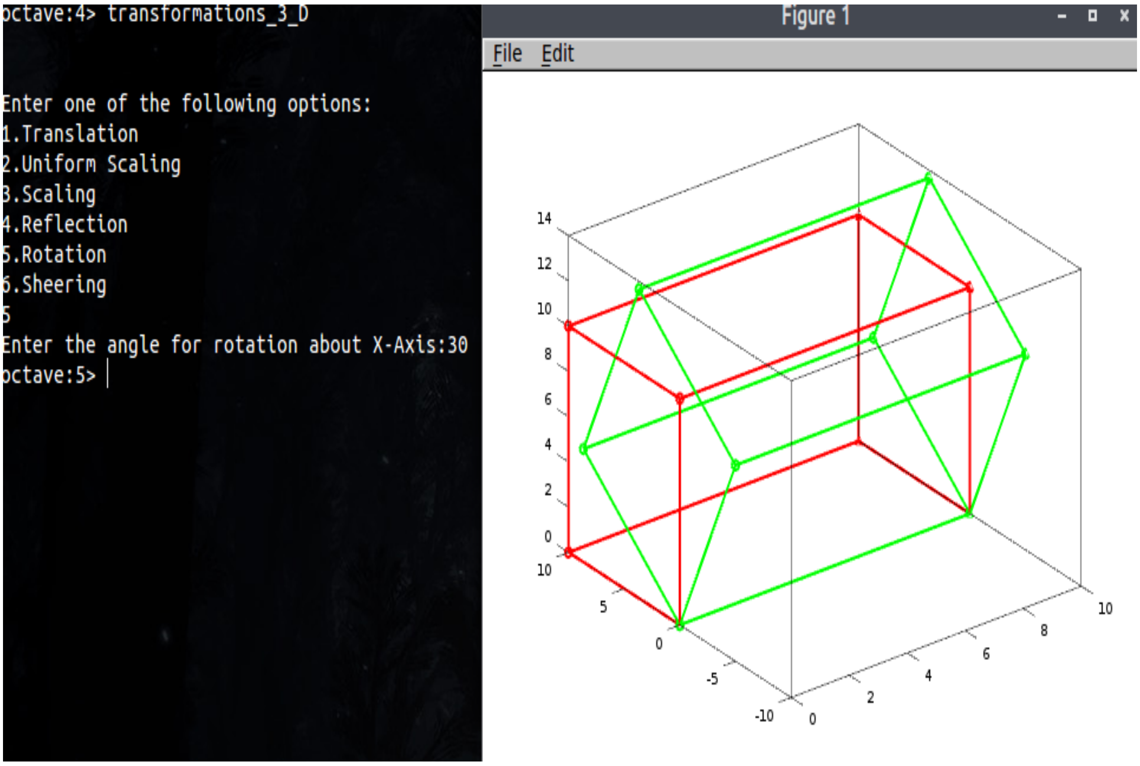


```
ctave:11> transformations_3_D

nter one of the following options:
.Translation
.Uniform Scaling
.Scaling
.Reflection
.Rotation
.Sheering


nter the sheer in X:0
nter the sheer in Y:0
nter the sheer in Z:1
ctave:12>
```

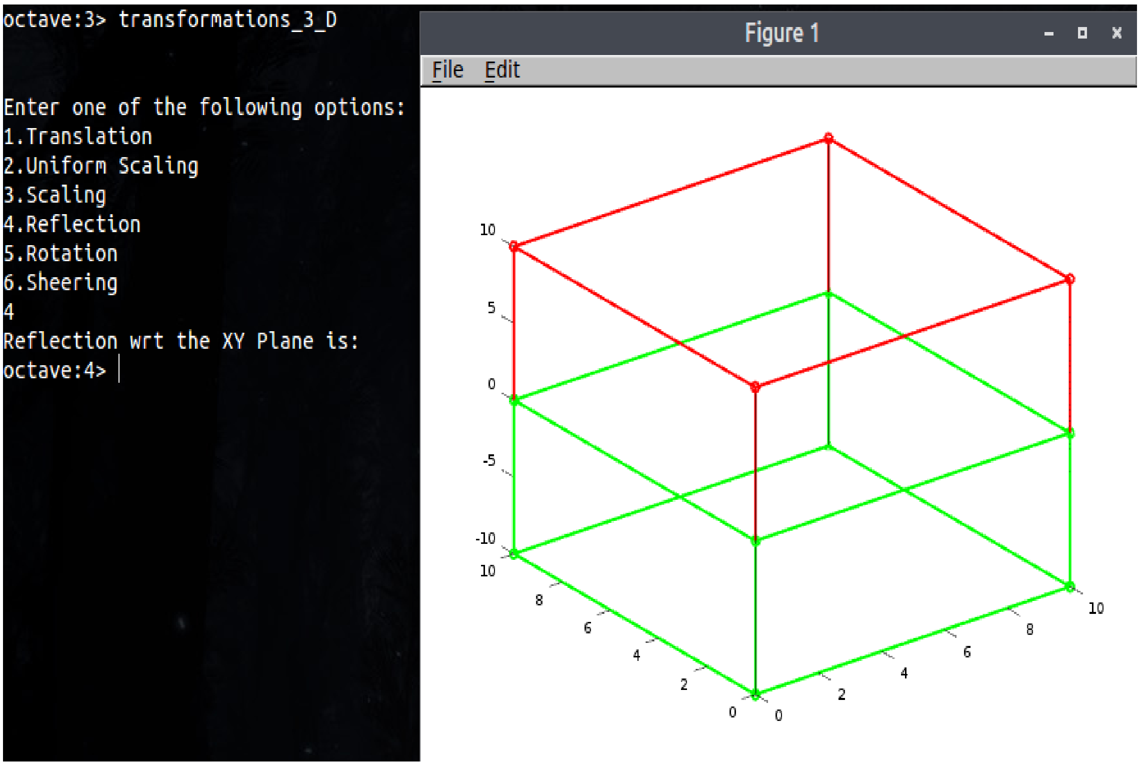
FIGURE 3 – sheer



```
octave:4> transformations_3_D

Enter one of the following options:
1.Translation
2.Uniform Scaling
3.Scaling
4.Reflection
5.Rotation
6.Sheering
5
Enter the angle for rotation about X-Axis:30
octave:5>
```

FIGURE 4 – rotation



```
octave:3> transformations_3_D

Enter one of the following options:
1.Translation
2.Uniform Scaling
3.Scaling
4.Reflection
5.Rotation
6.Sheering
4
Reflection wrt the XY Plane is:
octave:4>
```

FIGURE 5 – reflection