Ex. No. : 5 Date:

Register No.: 231701045 Name: P. Sahaana

.

3D Transformations on Basic Objects (Cube, Pyramid) AIM:

To write a program that allows the user to perform 3D transformations (translation, scaling, rotation) on basic 3D objects like a cube or pyramid, and visualize the results.

Procedure:

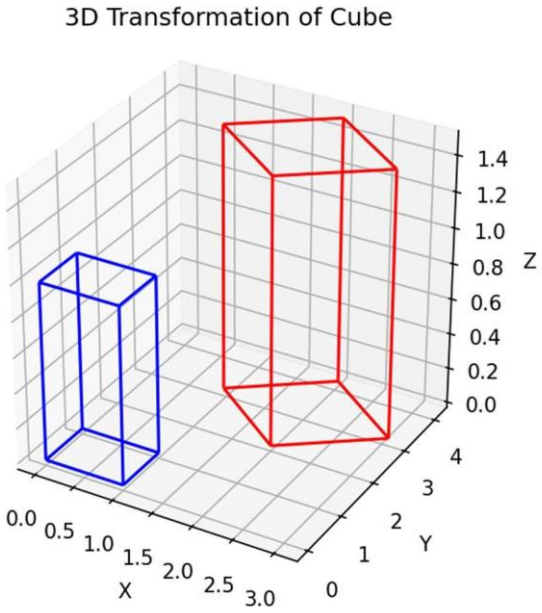
1. Define a 3D object using vertices and edges (cube or pyramid).
2. Use 4×4 homogeneous transformation matrices for:
   - Translation
   - Scaling
   - Rotation (around x, y, z axes)
3. Multiply the object's coordinates with the transformation matrix.
4. Project 3D points to 2D for visualization.
5. Display both original and transformed objects.

Program:

```
import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d.art3d import
Line3DCollection


def connect_edges(ax, pts, pairs, col):

    lines = [(pts[a], pts[b]) for a, b in pairs]

    ax.add_collection3d(Line3DCollection(lines,
colors=col))


def translate(x, y, z):

    return np.array([[1, 0, 0, x],

                     [0, 1, 0, y],

                     [0, 0, 1, z],

                     [0, 0, 0, 1]])
```

```
def scale(x, y, z):
    return np.array([[x, 0, 0, 0],
                     [0, y, 0, 0],
                     [0, 0, z, 0],
                     [0, 0, 0, 1]])

def rotate_z(deg):
    r = np.radians(deg)
    return np.array([[np.cos(r), -np.sin(r), 0, 0],
                     [np.sin(r),  np.cos(r), 0, 0],
                     [0, 0, 1, 0],
                     [0, 0, 0, 1]])

def transform(verts, mat):
    res = []
    for v in verts:
        v4 = np.array([*v, 1])
        t = mat @ v4
        res.append(t[:3])
    return res

cube_points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0),
               (0,0,1), (1,0,1), (1,1,1), (0,1,1)]

cube_edges = [(0,1),(1,2),(2,3),(3,0),
              (4,5),(5,6),(6,7),(7,4),
              (0,4),(1,5),(2,6),(3,7)]
```

```
m_translate = translate(2, 2, 0)

m_scale = scale(1.5, 1.5, 1.5)

m_rotate = rotate_z(45)


final_points = transform(cube_points, m_translate
@ m_scale @ m_rotate)


fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')

connect_edges(ax, cube_points, cube_edges, 'blue')

connect_edges(ax, final_points, cube_edges, 'red')

ax.set_title("3D Cube Transformations")

ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('Z')

ax.set_box_aspect([1,1,1])

plt.show()
```



Result:

The user was able to perform translation, scaling, and rotation on a 3D cube. The transformed cube was successfully rendered and visualized.