**Graphics and multimedia**

**3D Transformations on Basic Objects (Cube, Pyramid)**

**EXPERIMENT** : 5

**ROLL NO** : 231701012

**NAME** : DIVYA DHARRSHAN T

**--------------------------------------------------------------------------------------------------------------------------------------**

**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 draw\_edges(ax, vertices, edges, color='b'):

lines = [(vertices[start], vertices[end]) for start, end in edges]

ax.add\_collection3d(Line3DCollection(lines, colors=color))

def translation\_matrix(tx, ty, tz):

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

[0, 1, 0, ty],

[0, 0, 1, tz],

[0, 0, 0, 1]])

def scaling\_matrix(sx, sy, sz):

return np.array([[sx, 0, 0, 0],

[0, sy, 0, 0],

[0, 0, sz, 0],

[0, 0, 0, 1]])

def rotation\_matrix\_z(angle):

rad = np.radians(angle)

return np.array([[np.cos(rad), -np.sin(rad), 0, 0],

[np.sin(rad), np.cos(rad), 0, 0],

[0, 0, 1, 0],

[0, 0, 0, 1]])

def apply\_transform(vertices, matrix):

transformed = []

for v in vertices:

vec = np.array([\*v, 1])

result = matrix @ vec

transformed.append(result[:3])

return transformed

vertices = [(0,0,0), (1,0,0), (1,1,0), (0,1,0),

(0,0,1), (1,0,1), (1,1,1), (0,1,1)]

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)]

T = translation\_matrix(2, 2, 0)

S = scaling\_matrix(1.5, 1.5, 1.5)

R = rotation\_matrix\_z(45)

transformed\_vertices = apply\_transform(vertices, T @ S @ R)

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

draw\_edges(ax, vertices, edges, 'blue')

draw\_edges(ax, transformed\_vertices, edges, 'red')

ax.set\_title("3D Transformation of Cube")

ax.set\_xlabel('X')

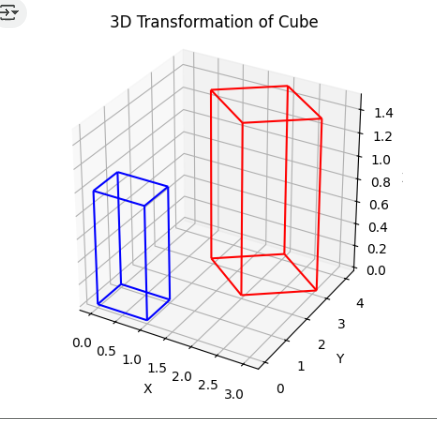
ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.set\_box\_aspect([1,1,1])

plt.show()

**OUTPUT**



**Result**

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