Ex. No. : 5 Date:

Register No. : 231701041 Name: R.Priyanka

Write a program that allows the user to perform 3D transformations on basic 3D objects (cube, pyramid) and view the results.

AIM:

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

Procedure:

1. Define the 3D object (cube or pyramid) using vertices and edges/faces.
2. Apply **transformation matrices** for:
    a. Translation (shifting along x, y, z).
    b. Scaling (enlarging or shrinking).
    c. Rotation (around x, y, z axes).
3. Multiply the original vertices by the chosen transformation matrix.
4. Use **perspective or orthographic projection** to convert 3D coordinates into 2D screen coordinates.
5. Draw the transformed object on the screen.
6. Allow the user to select object type (cube/pyramid) and the transformation to apply.

Program:

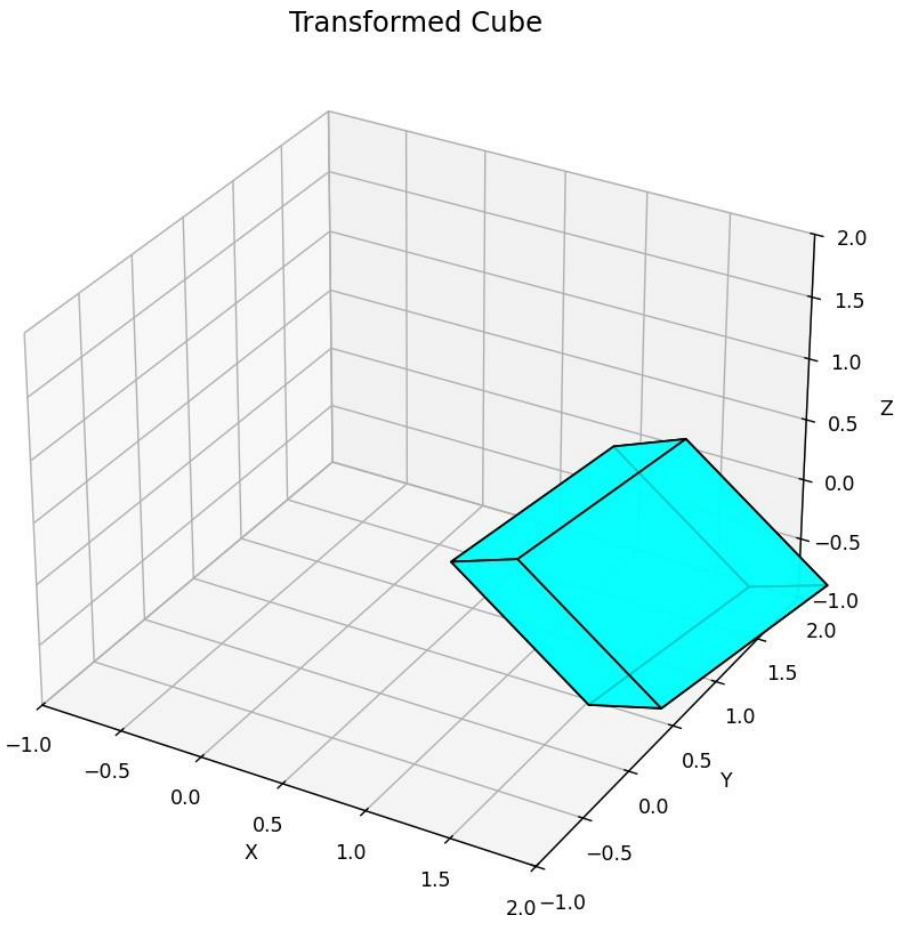
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d
import Poly3DCollection
def create_cube():
```

```
    return np.array([
        [0, 0, 0],
        [1, 0, 0],
        [1, 1, 0],
        [0, 1, 0],
        [0, 0, 1],
        [1, 0, 1],
        [1, 1, 1],
        [0, 1, 1]
    ])

def create_pyramid():
    return np.array([
        [0, 0, 0],
        [1, 0, 0],
        [1, 1, 0],
        [0, 1, 0],
        [0.5, 0.5, 1]
    ])

# ---------- Transformations ----------
def translate(vertices, tx, ty, tz):
    T = np.array([tx, ty, tz])
    return vertices + T

def scale(vertices, sx, sy, sz):
    S = np.diag([sx, sy, sz])
    return np.dot(vertices, S)

def rotate_x(vertices, angle):
    rad = np.radians(angle)
    R = np.array([
        [1, 0, 0],
        [0, np.cos(rad), -np.sin(rad)],
```

```
        [0, np.sin(rad), np.cos(rad)]
    ])
    return np.dot(vertices, R)

def rotate_y(vertices, angle):
    rad = np.radians(angle)
    R = np.array([
        [np.cos(rad), 0, np.sin(rad)],
        [0, 1, 0],
        [-np.sin(rad), 0, np.cos(rad)]
    ])
    return np.dot(vertices, R)

def rotate_z(vertices, angle):
    rad = np.radians(angle)
    R = np.array([
        [np.cos(rad), -np.sin(rad), 0],
        [np.sin(rad), np.cos(rad), 0],
        [0, 0, 1]
    ])
    return np.dot(vertices, R)

# - - - - - - - Draw - - - - - - -
def draw_object(vertices, faces,
title):
    fig = plt.figure()
    ax = fig.add_subplot(111,
projection='3d')

    poly3d = [[vertices[vert] for vert
in face] for face in faces]
```

```
ax.add_collection3d(Poly3DCollecti
on(poly3d, facecolors='cyan',
linewidths=1, edgecolors='black',
alpha=0.8))

    ax.set_title(title)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.auto_scale_xyz([0, 2], [0, 2],
[0, 2])
    plt.show()

# - - - - - - - Main - - - - - - -
choice = input("Choose object
(cube/pyramid): ").lower()

if choice == "cube":
    vertices = create_cube()
    faces = [[0,1,2,3], [4,5,6,7],
[0,1,5,4], [2,3,7,6], [1,2,6,5],
[0,3,7,4]]
elif choice == "pyramid":
    vertices = create_pyramid()
    faces = [[0,1,2,3], [0,1,4], [1,2,4],
[2,3,4], [3,0,4]]
else:
    print("Invalid choice.")
    exit()

# Apply transformations
vertices = translate(vertices, 0.5, 0.5,
0)
vertices = scale(vertices, 1.2, 1.2,
1.2)
```

```
vertices = rotate_x(vertices, 30)
vertices = rotate_y(vertices, 45)

# Draw
draw_object(vertices, faces,
f"Transformed
{choice.capitalize()}")
```



Final vertices for cube:

```
[[ 0.6363961   0.51961524 -0.21213203]
 [ 1.48492424  0.51961524 -1.06066017]
 [ 1.90918831  1.55884573 -0.6363961 ]
 [ 1.06066017  1.55884573  0.21213203]
 [ 1.37124303 -0.08038476  0.52271489]
 [ 2.21977116 -0.08038476 -0.32581325]
 [ 2.64403523  0.95884573  0.09845082]
 [ 1.79550709  0.95884573  0.94697896]]
```

Result:

The program successfully creates **3D objects** (cube/pyramid).