**Lab 10**

**AIM:**  
Implement 3D transformations of a object.  
**ALGORITHM:**   
1. Enter the choice for transformation.  
2. Perform the translation, rotation and scaling of 3d object.  
3. Get the needed parameters for the transformation from the user.  
4. Incase of rotation, object can be rotated about x or y axis.  
5. Display the transmitted object in the screen along with new generated coordinates.

**CODE:**

*import numpy as np*

*import matplotlib.pyplot as plt*

*from mpl\_toolkits.mplot3d import Axes3D*

*# Function for translation*

*def translate(x, y, z, Tx, Ty, Tz):*

*return x + Tx, y + Ty, z + Tz*

*# Function for rotation about X-axis*

*def rotate\_x(x, y, z, angle):*

*angle\_rad = np.radians(angle)*

*y\_new = y \* np.cos(angle\_rad) - z \* np.sin(angle\_rad)*

*z\_new = y \* np.sin(angle\_rad) + z \* np.cos(angle\_rad)*

*return x, y\_new, z\_new*

*# Function for rotation about Y-axis*

*def rotate\_y(x, y, z, angle):*

*angle\_rad = np.radians(angle)*

*x\_new = x \* np.cos(angle\_rad) + z \* np.sin(angle\_rad)*

*z\_new = -x \* np.sin(angle\_rad) + z \* np.cos(angle\_rad)*

*return x\_new, y, z\_new*

*# Function for scaling*

*def scale(x, y, z, Sx, Sy, Sz):*

*return x \* Sx, y \* Sy, z \* Sz*

*# Display the object*

*def display\_object(initial\_x, initial\_y, initial\_z, final\_x, final\_y, final\_z):*

*fig = plt.figure()*

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

*# Plot initial position (in blue)*

*ax.scatter(initial\_x, initial\_y, initial\_z, c='blue', label='Initial Position', marker='o')*

*# Plot final position (in red)*

*ax.scatter(final\_x, final\_y, final\_z, c='red', label='Final Position', marker='^')*

*# Set labels and title*

*ax.set\_xlabel('X')*

*ax.set\_ylabel('Y')*

*ax.set\_zlabel('Z')*

*ax.set\_title('3D Transformation: Initial vs Final Position')*

*# Show legend*

*ax.legend()*

*plt.show()*

*# Main logic*

*def main():*

*# Step 1: Enter the choice for transformation*

*print("Choose transformation:")*

*print("1. Translation")*

*print("2. Rotation")*

*print("3. Scaling")*

*choice = int(input("Enter choice (1/2/3): "))*

*# Initial coordinates of the 3D object (example: cube with 8 vertices)*

*x = np.array([1, -1, -1, 1, 1, -1, -1, 1])*

*y = np.array([1, 1, -1, -1, 1, 1, -1, -1])*

*z = np.array([1, 1, 1, 1, -1, -1, -1, -1])*

*# Step 2: Perform the transformation*

*if choice == 1: # Translation*

*Tx = float(input("Enter translation value for Tx: "))*

*Ty = float(input("Enter translation value for Ty: "))*

*Tz = float(input("Enter translation value for Tz: "))*

*final\_x, final\_y, final\_z = translate(x, y, z, Tx, Ty, Tz)*

*initial\_x, initial\_y, initial\_z = x, y, z*

*elif choice == 2: # Rotation*

*axis = input("Enter rotation axis (X/Y): ")*

*angle = float(input("Enter rotation angle in degrees: "))*

*if axis.lower() == 'x':*

*final\_x, final\_y, final\_z = rotate\_x(x, y, z, angle)*

*elif axis.lower() == 'y':*

*final\_x, final\_y, final\_z = rotate\_y(x, y, z, angle)*

*initial\_x, initial\_y, initial\_z = x, y, z*

*elif choice == 3: # Scaling*

*Sx = float(input("Enter scaling factor for Sx: "))*

*Sy = float(input("Enter scaling factor for Sy: "))*

*Sz = float(input("Enter scaling factor for Sz: "))*

*final\_x, final\_y, final\_z = scale(x, y, z, Sx, Sy, Sz)*

*initial\_x, initial\_y, initial\_z = x, y, z*

*# Step 3: Display the transformed object*

*display\_object(initial\_x, initial\_y, initial\_z, final\_x, final\_y, final\_z)*

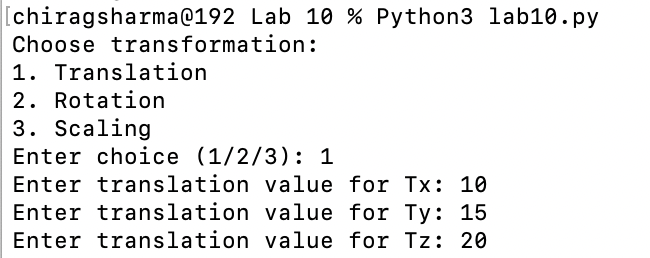
*# Run the program*

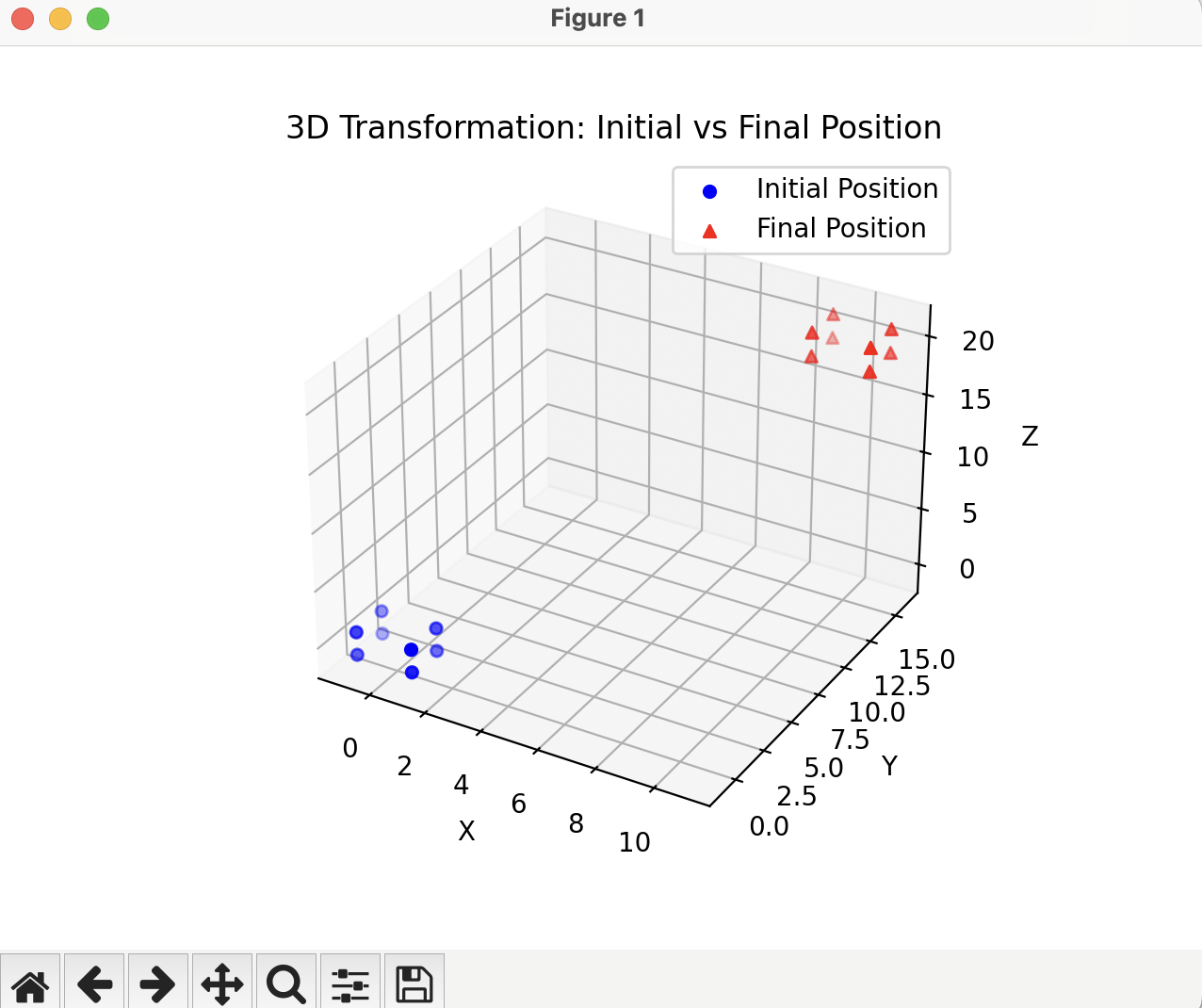
*if \_\_name\_\_ == "\_\_main\_\_":*

*main()*

**OUTPUT:**

**Translation:**

****

****

**Rotation:**

**A black text on a white background

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Scaling:**

**A white background with black text

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**