2D Geometric Transformations

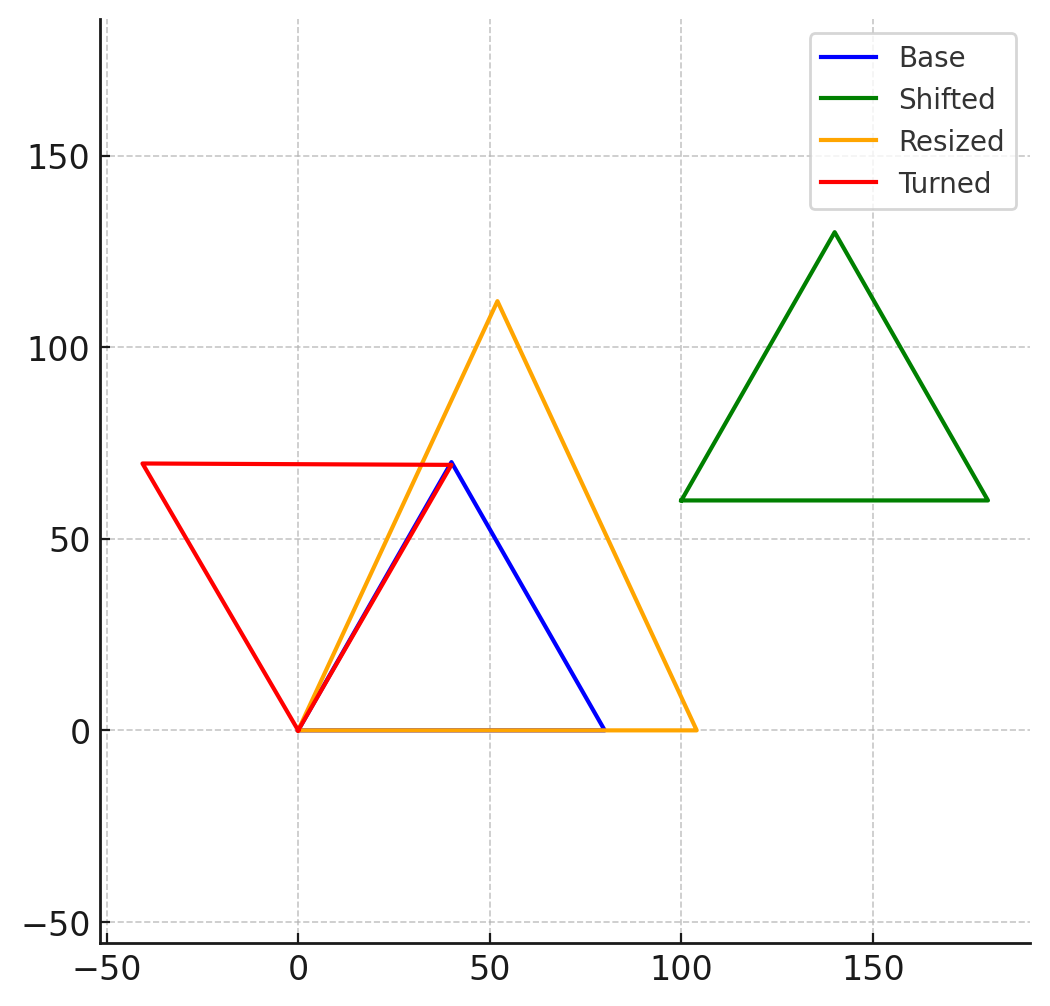
# Aim

To implement 2D geometric transformations (Translation, Scaling, Rotation) on a triangular shape using homogeneous coordinates in Python.

# Code

import numpy as np  
import matplotlib.pyplot as plt  
  
def plot\_shape(pts, name, col):  
 x, y = zip(\*pts)  
 x += (x[0],)  
 y += (y[0],)  
 plt.plot(x, y, color=col, label=name)  
  
def move(pts, dx, dy):  
 M = np.array([[1, 0, dx],  
 [0, 1, dy],  
 [0, 0, 1]])  
 return transform(pts, M)  
  
def resize(pts, fx, fy):  
 S = np.array([[fx, 0, 0],  
 [0, fy, 0],  
 [0, 0, 1]])  
 return transform(pts, S)  
  
def turn(pts, ang):  
 r = np.radians(ang)  
 R = np.array([[np.cos(r), -np.sin(r), 0],  
 [np.sin(r), np.cos(r), 0],  
 [0, 0, 1]])  
 return transform(pts, R)  
  
def transform(pts, mat):  
 new\_pts = []  
 for x, y in pts:  
 vec = np.array([x, y, 1])  
 res = mat @ vec  
 new\_pts.append((res[0], res[1]))  
 return new\_pts  
  
tri = [(0, 0), (80, 0), (40, 70)]  
shifted = move(tri, 100, 60)  
enlarged = resize(tri, 1.3, 1.6)  
angled = turn(tri, 60)  
  
plt.figure(figsize=(8, 8))  
plot\_shape(tri, "Base", 'blue')  
plot\_shape(shifted, "Shifted", 'green')  
plot\_shape(enlarged, "Resized", 'orange')  
plot\_shape(angled, "Turned", 'red')  
plt.legend()  
plt.axis("equal")  
plt.grid(True)  
plt.show()

# Output



# Conclusion

The experiment successfully demonstrates 2D transformations in computer graphics. Using matrix multiplication, shapes can be moved, resized, and rotated efficiently in homogeneous coordinate space. This approach is widely used in modern graphics pipelines such as OpenGL and DirectX.