**EXP:03** Write a program that performs translation, scaling, and rotation on basic 2D shapes (e.g.. triangle, rectangle) using matrices. give code in python simple

1. Translation

import numpy as np

import matplotlib.pyplot as plt

def transform(points, matrix):

return np.dot(matrix, points)

triangle = np.array([[0, 1, 0.5, 0],

[0, 0, 1, 0],

[1, 1, 1, 1]])

T = np.array([[1, 0, 2],

[0, 1, 1],

[0, 0, 1]]

triangle\_translated = transform(triangle, T.T)

plt.plot(triangle[0], triangle[1], 'b-', label='Original')

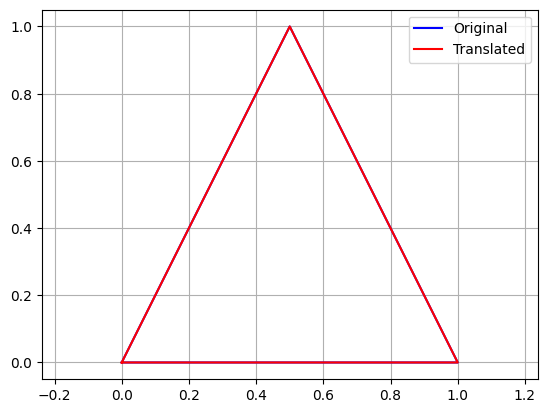
plt.plot(triangle\_translated[0], triangle\_translated[1], 'r-', label='Translated')

plt.legend()

plt.axis('equal')

plt.grid(True)

plt.show()



1. Scaling

import numpy as np

import matplotlib.pyplot as plt

def transform(points, matrix):

return np.dot(matrix, points)

triangle = np.array([[0, 1, 0.5, 0],

[0, 0, 1, 0],

[1, 1, 1, 1]])

# Scaling matrix (sx=2, sy=1.5)

S = np.array([[2, 0, 0],

[0, 1.5, 0],

[0, 0, 1]])

triangle\_scaled = transform(triangle, S.T)

plt.plot(triangle[0], triangle[1], 'b-', label='Original')

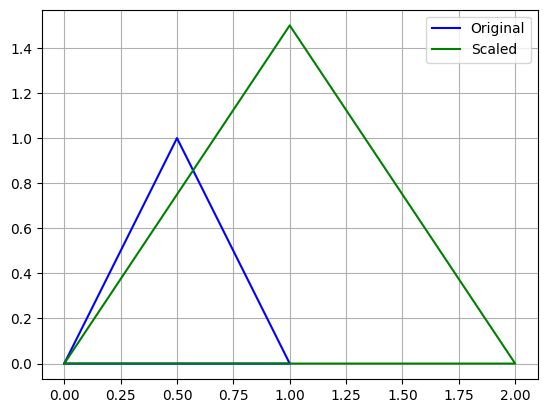
plt.plot(triangle\_scaled[0], triangle\_scaled[1], 'g-', label='Scaled')

plt.legend()

plt.axis('equal')

plt.grid(True)

plt.show()



1. Rotation

import numpy as np

import matplotlib.pyplot as plt

def transform(points, matrix):

return np.dot(matrix, points)

triangle = np.array([[0, 1, 0.5, 0],

[0, 0, 1, 0],

[1, 1, 1, 1]])

theta = np.radians(45)

R = np.array([[np.cos(theta), -np.sin(theta), 0],

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

[0, 0, 1]])

triangle\_rotated = transform(triangle, R.T)

plt.plot(triangle[0], triangle[1], 'b-', label='Original')

plt.plot(triangle\_rotated[0], triangle\_rotated[1], 'm-', label='Rotated 45°')

plt.legend()

plt.axis('equal')

plt.grid(True)

plt.show()

