Q1. Translate a triangle with translation factors tx and ty

Algorithm: Translation of a Triangle

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the triangle in homogeneous form ((x, y, 1)) and store them in a matrix triangle.
4. Input the translation factors tx (translation in x-direction) and ty (translation in y-direction).
5. Construct the translation matrix
6. Multiply the translation matrix with the triangle matrix to obtain the translated triangle: translated_tri = translation_mat × triangle
7. Plot the original triangle using its coordinate points.
8. Plot the translated triangle using the new coordinates obtained after translation.
9. Add labels, legend, grid, and equal axis scaling for better visualization.
10. Display the plot.
11. Stop.

2. Scale a rectangle about an origin

Algorithm: Scaling of a Rectangle About Origin

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the rectangle in homogeneous form ((x, y, 1)) and store them in a matrix rectangle.
4. Input the scaling factors: sx for scaling in the x-direction and sy for scaling in the y-direction.
5. Construct the scaling matrix (about origin).
6. Multiply the scaling matrix with the rectangle matrix to obtain the scaled rectangle: scaled_rect = scaling_mat × rectangle
7. Plot the original rectangle using its coordinate points.
8. Plot the scaled rectangle using the new coordinates obtained after scaling.
9. Add labels (x-axis, y-axis), title, legend, grid, and set equal axis scaling.

10. Display the plot.
11. Stop.

3. Rotate a triangle about an origin

Algorithm: Rotation of a Triangle About Origin

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the triangle in homogeneous form
   ((x, y, 1)) and store them in a matrix triangle.
4. Input the rotation angle θ in degrees.
5. Convert the angle from degrees to radians: rad = θ × (π / 180) or use a built-in function to convert degrees to radians.
6. Construct the rotation matrix (about origin).
7. Multiply the rotation matrix with the triangle matrix to obtain the rotated triangle: rotated_tri = rotation_mat × triangle
8. Plot the original triangle using its coordinate points.
9. Plot the rotated triangle using the new coordinates obtained after rotation.
10. Add labels (x-axis, y-axis), title, legend, grid, and set equal axis scaling.
11. Display the plot.
12. Stop.

4. Reflect a triangle

Algorithm: Reflection of a Triangle about the X-axis

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the triangle in homogeneous form
   ((x, y, 1)) and store them in a matrix triangle.
4. Construct the reflection matrix about the x-axis:
5. Multiply the reflection matrix with the triangle matrix to obtain the reflected triangle: reflected_tri = ref × triangle

6. Plot the original triangle using its coordinate points.
7. Plot the reflected triangle using the new coordinates obtained after reflection.
8. Add labels (x-axis, y-axis), title, legend, grid, and set equal axis scaling.
9. Display the plot.
10. Stop.

b. about origin

Algorithm: Reflection of a Triangle about the Origin

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the triangle in homogeneous form ((x, y, 1)) and store them in a matrix triangle.
4. Construct the reflection matrix about the origin.
5. Multiply the reflection matrix with the triangle matrix to obtain the reflected triangle: reflected_tri = ref_origin × triangle
6. Plot the original triangle using its coordinate points.
7. Plot the reflected triangle using the new coordinates obtained after reflection.
8. Add labels (x-axis, y-axis), title, legend, grid, and set equal axis scaling.
9. Display the plot.
10. Stop.

c. about y = x

Algorithm: Reflection of a Triangle about the Line (y = x)

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the triangle in homogeneous form ((x, y, 1)) and store them in a matrix triangle.
4. Construct the reflection matrix about the line (y = x).

5. Multiply the reflection matrix with the triangle matrix to obtain the reflected triangle: reflected_tri = reflect_y_eq_x × triangle
6. Plot the original triangle using its coordinate points.
7. Plot the reflected triangle using the new coordinates obtained after reflection.
8. Add labels (x-axis, y-axis), title, legend, grid, and set equal axis scaling.
9. Display the plot.
10. Stop.

d. about y = mx + c

Algorithm: Reflection of Triangle about y=mx+cy=mx+c

1. Start
2. Input the coordinates of the triangle vertices and the line equation y=mx+cy=mx+c.
3. Compute the angle θ of the line with the x-axis: θ = arctan(m).
4. Translate the triangle down by c units (so the line passes through the origin).
5. Rotate the triangle clockwise by θ to align the line with the x-axis.
6. Reflect the triangle about the x-axis.
7. Rotate back counter-clockwise by θ to restore the line's original slope.
8. Translate back vertically by c units to move the line to its original position.
9. Combine all transformations into a single operation (optional for coding efficiency).
10. Apply the combined transformation to each vertex of the triangle.
11. Plot the original and reflected triangles for visualization.
12. Stop

5. Shear a rectangle

a. Towards X-direction

Algorithm: Shearing of a Rectangle in X-Direction

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the rectangle in homogeneous form
   ((x, y, 1)) and store them in a matrix rectangle.

4. Input the shear factor in the x-direction, shx.
5. Construct the shear matrix in x-direction.
6. Multiply the shear matrix with the rectangle matrix to obtain the sheared rectangle: sheared_rect = shear_x × rectangle
7. Plot the original rectangle using its coordinate points.
8. Plot the sheared rectangle using the new coordinates obtained after shearing.
9. Add title, axis labels, legend, grid, and set equal axis scaling.
10. Display the plot.
11. Stop.

b. toward both direction

Algorithm: Shearing of a Rectangle in Both Directions

1. Start
2. Import the required libraries: NumPy for matrix operations and Matplotlib for plotting
3. Define the coordinates of the rectangle in homogeneous form ((x, y, 1)) and store them in a matrix rectangle.
4. Input the shear factors: shx for shearing in the x-direction and shy for shearing in the y-direction.
5. Construct the shear matrix in both directions:
   Multiply the shear matrix with the rectangle matrix to obtain the sheared rectangle: sheared_rect = shear_xy × rectangle
6. Plot the original rectangle using its coordinate points.
7. Plot the sheared rectangle using the new coordinates obtained after shearing.
8. Add title, axis labels, legend, grid, and set equal axis scaling.
9. Display the plot.
10. Stop.