Tutorial No 5

Name : Sarthak Shrikant Bagul

Branch : ENTC

Div :- A

PRN :- 12410555

Roll No :- 25

Subject :- Computer Graphics (MDM)

Title / Problem Statement:

2D Transformations in Computer Graphics – Translation, Scaling, Rotation, Reflection, and Shearing

Understand and apply 2D transformations on graphical objects such as points, lines, and polygons. Explore different types of transformations and their effects on the shape, position, and orientation of objects in 2D space.

Objectives: By the end of this tutorial, students should be able to:

- Define and explain the different types of 2D transformations.
- Apply transformations such as **translation, scaling, rotation, reflection, and shearing** to 2D objects.
- Represent transformations mathematically using **homogeneous matrices**.
- Understand and analyze the effects of transformations on 2D objects in computer graphics.

Prerequisites:

- Knowledge of coordinate geometry (points, lines, polygons).
- Basic linear algebra (matrices, matrix multiplication).
- Basic trigonometry (sine, cosine for rotation).
- Understanding of computer graphics basics (pixels, raster displays).

Theory Overview:

1. **2D Transformations:**
   Transformations are operations that alter the position, size, orientation, or shape of graphical objects in 2D space.
2. **Types of Transformations:**
   (a) **Translation:** Moves an object from one location to another without changing its shape or orientation.

- Formula:

$$x' = x + tx, \quad y' = y + ty$$

(b) **Scaling:** Changes the size of an object relative to a fixed point (usually origin).

- Formula:

$$x' = Sx \cdot x, \quad y' = Sy \cdot y$$

- Scaling around a pivot requires moving the object to origin, scaling, and moving back.

(c) **Rotation:** Rotates an object around a fixed point (usually origin) by angle θ.

- Formula:

$$x' = x\cos\theta - y\sin\theta, \quad y' = x\sin\theta + y\cos\theta$$

(d) **Reflection:** Produces a mirror image of an object about a line (x-axis, y-axis, y=x, etc.).

(e) **Shearing:** Slants the shape of an object along x or y axis.

- X-shear: $x' = x + sh_x \cdot y$
- Y-shear: $y' = y + sh_y \cdot x$

3. **Homogeneous Coordinates:**

- Transformations can be represented by 3×3 matrices to combine multiple operations conveniently:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & tx \\ c & d & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- This allows **composite transformations** like scaling + rotation + translation in a single operation.

4. **Order of Transformations:**

- The sequence of transformations matters:
    - **Rotation then translation ≠ Translation then rotation**
- Composite transforms often use a **pivot point** to control rotation or scaling.

Algorithm / Steps:

A. Translation:

1. For each point (x, y), calculate:

$$x' = x + tx,\ y' = y + ty$$

2. Plot the new points.

B. Scaling:

1. Select scale factors Sx,Sy.
2. For each point (x, y), calculate:

$$x' = x \cdot Sx,\ \ y' = y \cdot Sy$$

3. Plot scaled points.

C. Rotation:

1. Select rotation angle θ.
2. For each point (x, y), calculate:

$$x' = x\cos\theta - y\sin\theta,\ y' = x\sin\theta + y\cos\theta$$

3. Plot rotated points.

D. Reflection:

1. Choose the axis of reflection (x-axis, y-axis, y=x).
2. Apply the corresponding reflection formula.

E. Shearing:

1. Select shear factors sh_x or sh_y.
2. For each point (x, y), apply:

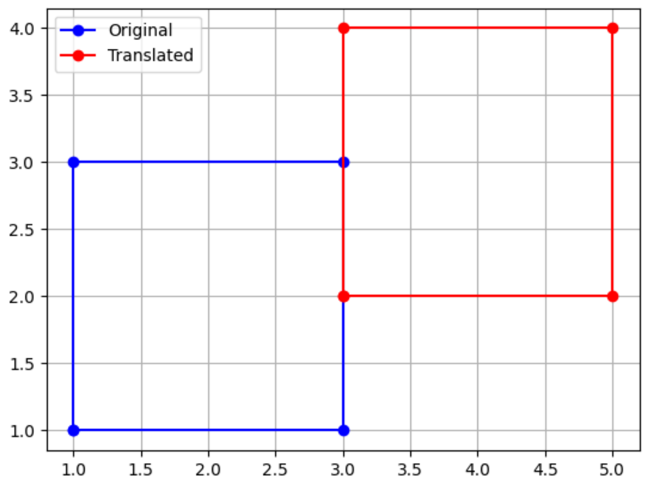
$$x' = x + sh_x \cdot y, \quad y' = y + sh_y \cdot x$$

3. Plot sheared points.

Code Implementation (Python + Matplotlib)

(i) Translation:

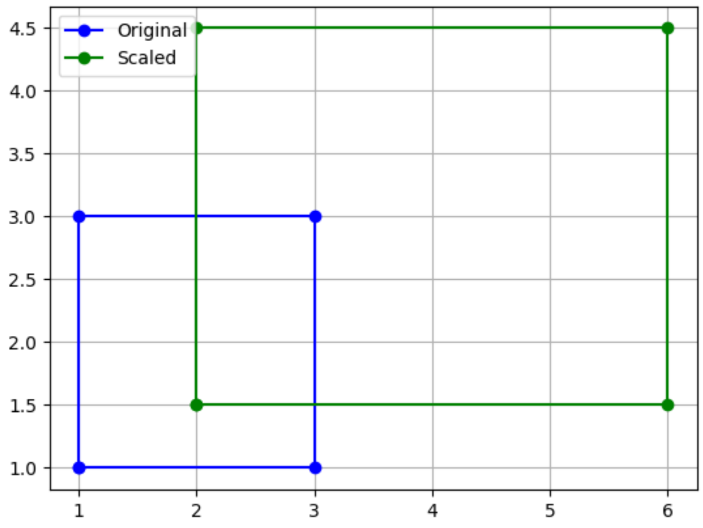
```
import matplotlib.pyplot as plt

def translate(points, tx, ty):

return [(x + tx, y + ty) for x, y in points]

# Original square

square = [(1,1),(1,3),(3,3),(3,1),(1,1)]

translated_square = translate(square, 2, 1)

x_orig, y_orig = zip(*square)

x_trans, y_trans = zip(*translated_square)

plt.plot(x_orig, y_orig, 'b-o', label='Original')

plt.plot(x_trans, y_trans, 'r-o', label='Translated')

plt.legend()

plt.grid(True)

plt.show()
```



(ii) Scaling:

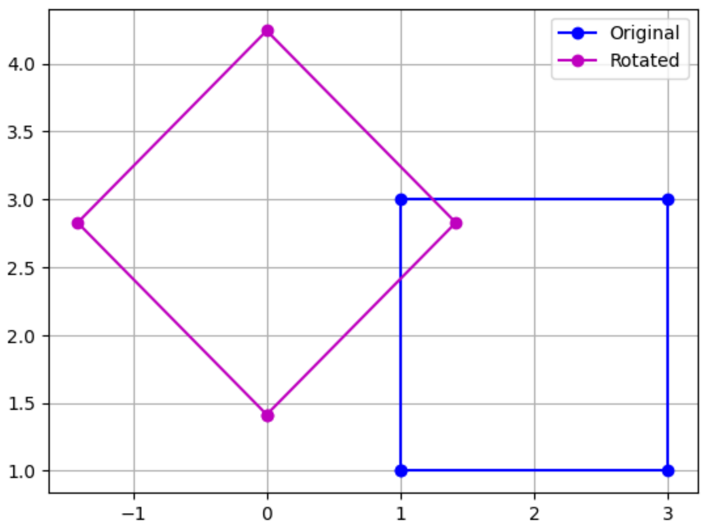
```
def scale(points, sx, sy):

  return [(x * sx, y * sy) for x, y in points]
```

```
scaled_square = scale(square, 2, 1.5)

x_scaled, y_scaled = zip(*scaled_square)

plt.plot(x_orig, y_orig, 'b-o', label='Original')

plt.plot(x_scaled, y_scaled, 'g-o', label='Scaled')

plt.legend()

plt.grid(True)

plt.show()
```



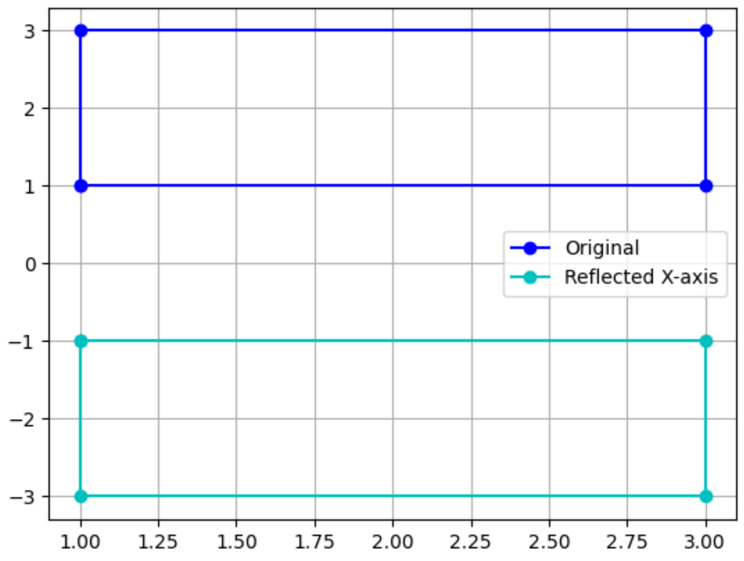
(iii) Rotation:

```
import math

def rotate(points, theta):

   rad = math.radians(theta)

   return [(x*math.cos(rad) - y*math.sin(rad), x*math.sin(rad) + y*math.cos(rad)) for x,y in points]

rotated_square = rotate(square, 45)

x_rot, y_rot = zip(*rotated_square)

plt.plot(x_orig, y_orig, 'b-o', label='Original')

plt.plot(x_rot, y_rot, 'm-o', label='Rotated')

plt.legend()

plt.grid(True)

plt.show()
```



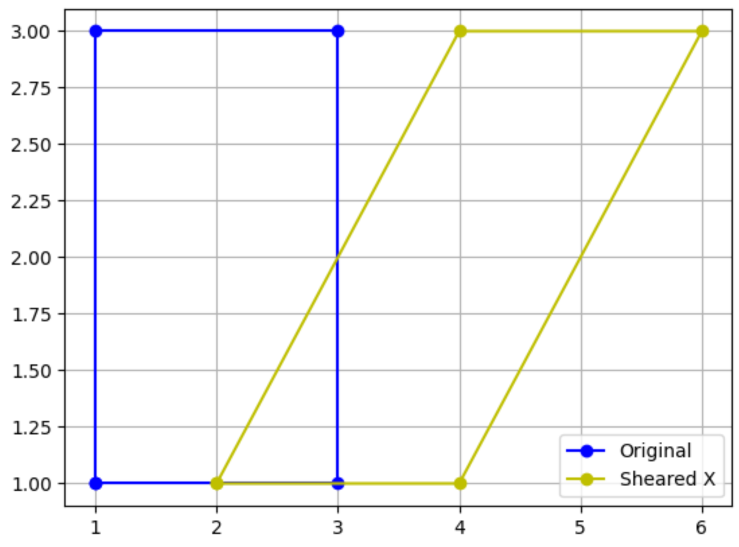
(iv) Reflection (about x-axis):

```
def reflect_x(points):
    return [(x, -y) for x, y in points]
reflected_square = reflect_x(square)
x_ref, y_ref = zip(*reflected_square)
plt.plot(x_orig, y_orig, 'b-o', label='Original')
plt.plot(x_ref, y_ref, 'c-o', label='Reflected X-axis')
plt.legend()
plt.grid(True)
plt.show()
```



(v) Shearing (x-direction):

```
def shear_x(points, shx):
    return [(x + shx*y, y) for x, y in points]
sheared_square = shear_x(square, 1)
x_shear, y_shear = zip(*sheared_square)
plt.plot(x_orig, y_orig, 'b-o', label='Original')
plt.plot(x_shear, y_shear, 'y-o', label='Sheared X')
plt.legend()
plt.grid(True)
plt.show()
```



Result / Observation / Conclusion:

- **Translation** moves the object without changing shape.
- **Scaling** changes object size; non-uniform scaling distorts shape.
- **Rotation** changes orientation while keeping shape and size intact.
- **Reflection** creates mirror images along specified axes.
- **Shearing** slants the object, altering angles but not parallelism.
- **Homogeneous matrices** allow combining multiple transformations efficiently.
- These transformations demonstrate how mathematical operations directly manipulate pixel positions on raster displays.