Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Computer Graphics Lab Report 04

on

'**Polygon Transforming Algorithms - Lab 04 Task**'

Submitted By:

Reewaj Khanal (61)

Submitted to:

Mr. Dhiraj Shrestha

Assistant Professor

Department of Computer Science and Engineering

School of Engineering

Kathmandu University

Dhulikhel, Kavre

Submission Date: Thursday 06 June 2024

Question No. 1 Write a Program to implement:

- 2D Translation
- 2D Rotation
- 2D Scaling
- 2D Reflection
- 2D Shearing

Composite Transformation (Should be able to perform at least 3 transformations)

(For doing these Transformations consider any 2D shapes (Line, Triangle, Rectangle etc), and use Homogeneous coordinate Systems)

Answer:

```
import pygame

from pygame.locals import *

from OpenGL.GL import *

from OpenGL.GLUT import *

from OpenGL.GLU import *

import numpy as np


# Function to draw axes

def draw_axes():

   glBegin(GL_LINES)

   glColor3f(1.0, 1.0, 1.0)  # Set color to white

   glVertex2i(-400, 0)

   glVertex2i(400, 0)

   glVertex2i(0, -300)
```

```
    glVertex2i(0, 300)
    glEnd()

# Function to draw a triangle
def draw_triangle(vertices=[[0, 0], [100, 0], [100, 100]], color=[1, 1, 1]):
    glBegin(GL_TRIANGLES)
    glColor3f(color[0], color[1], color[2])
    for vertex in vertices:
        glVertex2f(*vertex)
    glEnd()

# Transformation matrices
def translate(tx, ty):
    return np.array([
        [1, 0, tx],
        [0, 1, ty],
        [0, 0, 1]
    ])

def rotate(theta):
    cos_theta = np.cos(theta)
    sin_theta = np.sin(theta)
    return np.array([
        [cos_theta, -sin_theta, 0],
        [sin_theta, cos_theta, 0],
        [0, 0, 1]
    ])

def scale(sx, sy):
    return np.array([
```

```
        [sx, 0, 0],
        [0, sy, 0],
        [0, 0, 1]
    ])

def reflect_x():
    return np.array([
        [1, 0, 0],
        [0, -1, 0],
        [0, 0, 1]
    ])

def reflect_y():
    return np.array([
        [-1, 0, 0],
        [0, 1, 0],
        [0, 0, 1]
    ])

def reflect_xy():
    return np.array([
        [0, 1, 0],
        [1, 0, 0],
        [0, 0, 1]
    ])

def shear(kx, ky):
    return np.array([
        [1, kx, 0],
        [ky, 1, 0],
```

```
        [0, 0, 1]
    ])

def composite(*transformations):
    result = np.eye(3)
    for transformation in transformations:
        result = np.dot(transformation, result)
    return result

def display_menu():
    print("Choose an operation:")
    print("1. Translation")
    print("2. Rotation")
    print("3. Scaling")
    print("4. Reflection")
    print("5. Shearing")
    print("6. Composite Transformation")
    print("7. Exit")

def get_triangle_vertices():
    vertices = []
    for i in range(3):
        while True:
            try:
                x, y = map(float, input(f"Enter coordinate {i+1} (x,y): ").split(","))
                vertices.append([x, y])
                break
            except ValueError:
                print("Invalid input! Please enter numbers separated by comma.")
    return vertices
```

```
def get_input():
    operation = int(input("Enter operation number: "))
    if operation == 7:
        print("Exiting program.")
        return operation, None
    elif operation == 6:
            print("Enter operations to be composed (e.g., '1 2 3' for translation,
rotation, scaling):")
        operations = list(map(int, input().split()))
        return operation, operations
    return operation, None


def main():
    pygame.init()
    display = (800, 600)
    pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
    gluOrtho2D(-400, 400, -300, 300)  # Set up 2D coordinate system

    # The default coordinates to use
    vertices = [[0, 0], [100, 0], [100, 100]]
    # Get user input for coordinates
    # vertices=get_triangle_vertices()
    vertices_homogeneous = [[x, y, 1] for x, y in vertices]
    vertices_array = np.array(vertices_homogeneous)

    transformed_vertices = vertices  # Initialize with original vertices

    while True:
        for event in pygame.event.get():
```

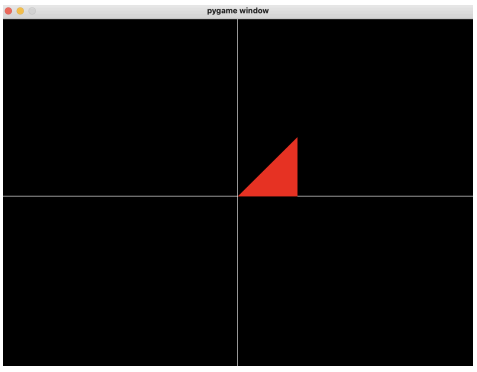
```
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        draw_axes()
        draw_triangle(vertices, [1, 1, 1])  # Draw original triangle in white
         draw_triangle(transformed_vertices, [1, 0, 0])  # Draw transformed triangle in
red
        pygame.display.flip()

        display_menu()
        operation, operations = get_input()

        if operation == 7:
            break

        if operation == 6:
            transformations = []
            for op in operations:
                if op == 1:
                            tx, ty = map(int, input("Enter translation values (tx,ty):
").split(","))
                    transformations.append(translate(tx, ty))
                elif op == 2:
                    theta = float(input("Enter rotation angle (in degrees): "))
                    transformations.append(rotate(np.radians(theta)))
                elif op == 3:
                            sx, sy = map(float, input("Enter scaling factors (sx,sy):
").split(","))
                    transformations.append(scale(sx, sy))
```

```
                elif op == 4:
                    axis = input("Enter reflection axis (x or y or x=y): ")
                    if axis == 'x':
                        transformations.append(reflect_x())
                    if axis=="y":
                        transformations.append(reflect_y())
                    if axis=="x=y":
                        transformations.append(reflect_xy())
                elif op == 5:
                            kx, ky = map(float, input("Enter shearing factors (kx,ky):
").split(","))
                    transformations.append(shear(kx, ky))


            composite_transform = composite(*transformations)
            print("Composite Transformation Matrix:")
            print(composite_transform)
              transformed_vertices = np.dot(composite_transform, vertices_array.T).T[:,
:2]
             transformed_vertices = transformed_vertices.tolist()  # Update transformed
vertices
        else:
            if operation == 1:
                        tx, ty = map(int, input("Enter translation values (tx,ty):
").split(","))
                transformation_matrix = translate(tx, ty)
            elif operation == 2:
                theta = float(input("Enter rotation angle (in degrees): "))
                transformation_matrix = rotate(np.radians(theta))
            elif operation == 3:
                        sx, sy = map(float, input("Enter scaling factors (sx,sy):
").split(","))
                transformation_matrix = scale(sx, sy)
```

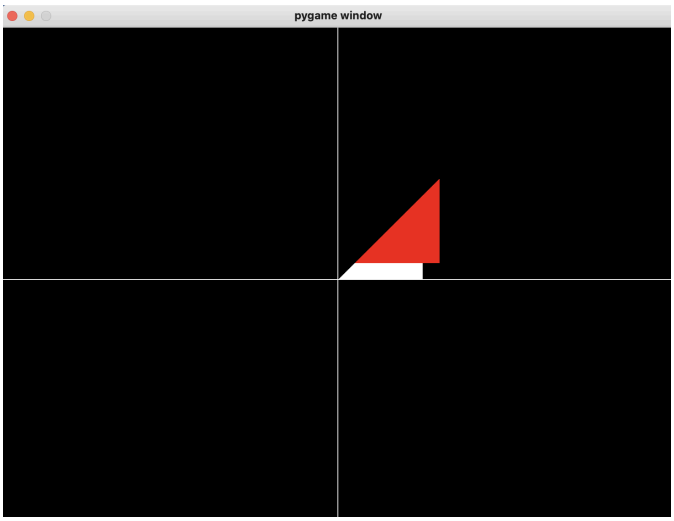
```
            elif operation == 4:
                axis = input("Enter reflection axis (x or y or x=y): ")
                if axis == 'x':
                    transformation_matrix = reflect_x()
                if axis=="y":
                    transformation_matrix=reflect_y()
                if axis=="x=y":
                    transformation_matrix=reflect_xy()
            elif operation == 5:
                        kx, ky = map(float, input("Enter shearing factors (kx,ky):
").split(","))
                transformation_matrix = shear(kx, ky)

            print("Transformation Matrix:")
            print(transformation_matrix)
             transformed_vertices = np.dot(transformation_matrix, vertices_array.T).T[:,
:2]
              transformed_vertices = transformed_vertices.tolist()  # Update transformed
vertices

        pygame.time.wait(10)

if __name__ == "__main__":
    main()
```

Inputs and Outputs:

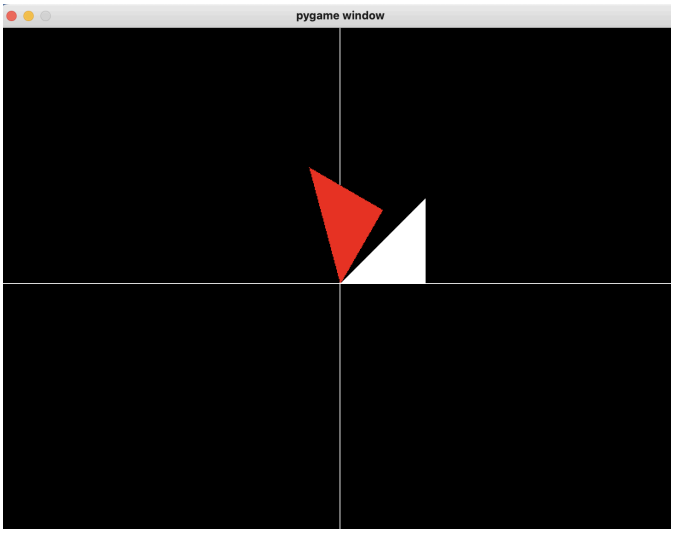
```
(base) reewajkhanal.rk10@RK10 LAB03 % python homotrans.py
pygame 2.5.2 (SDL 2.28.3, Python 3.10.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
Enter operation number:
```



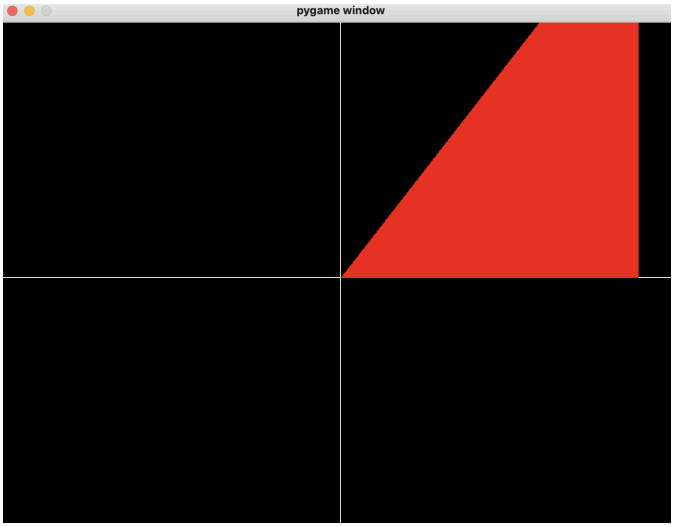
```
Enter operation number: 1
Enter translation values (tx,ty): 20,20
Transformation Matrix:
[[ 1  0 20]
 [ 0  1 20]
 [ 0  0  1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
Enter operation number:
```



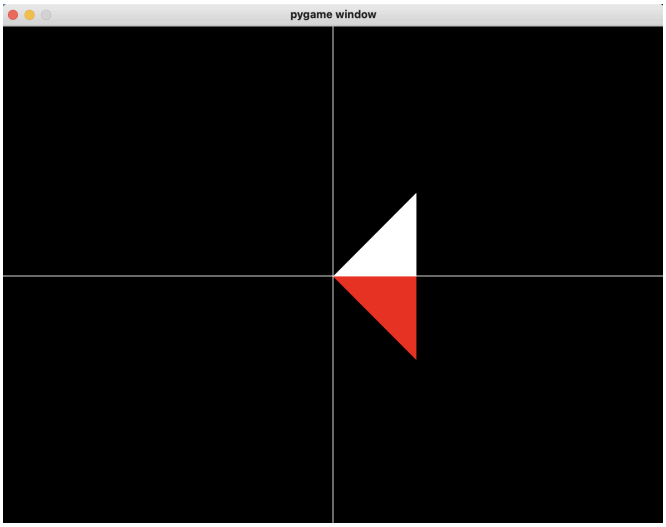

```
Enter operation number: 2
Enter rotation angle (in degrees): 60
Transformation Matrix:
[[ 0.5       -0.8660254  0.       ]
 [ 0.8660254  0.5        0.       ]
 [ 0.         0.         1.       ]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



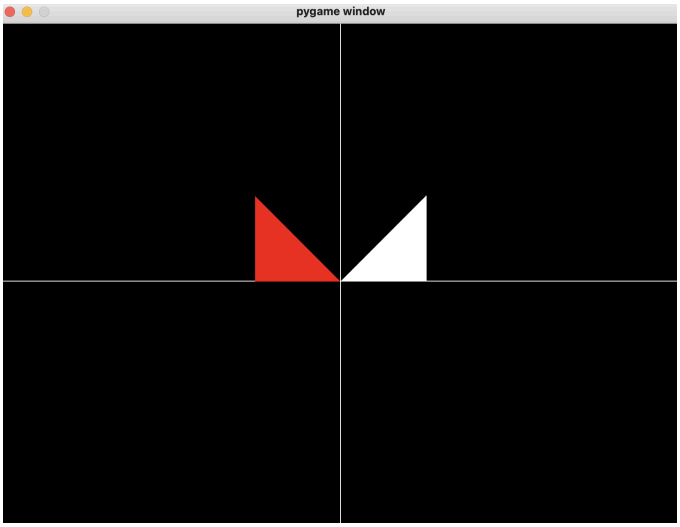
```
Enter operation number: 3
Enter scaling factors (sx,sy): 3.5,4.5
Transformation Matrix:
[[3.5 0.  0. ]
 [0.  4.5 0. ]
 [0.  0.  1. ]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



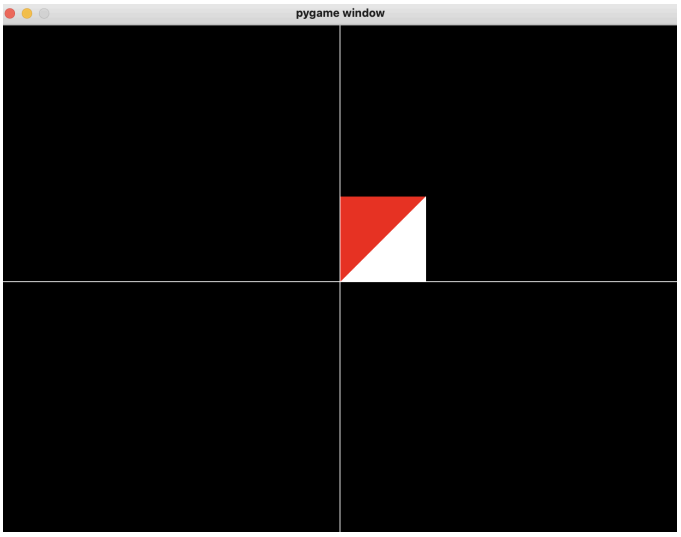

```
Enter operation number: 4
Enter reflection axis (x or y or x=y): x
Transformation Matrix:
[[ 1  0  0]
 [ 0 -1  0]
 [ 0  0  1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



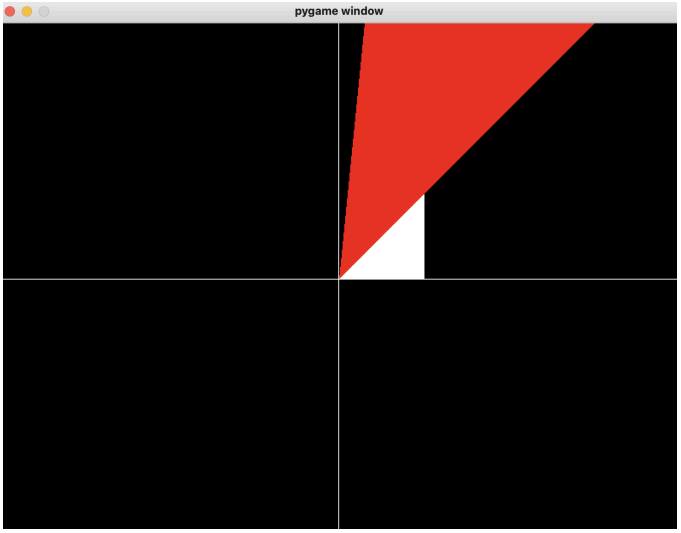

```
Enter operation number: 4
Enter reflection axis (x or y or x=y): y
Transformation Matrix:
[[-1  0  0]
 [ 0  1  0]
 [ 0  0  1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



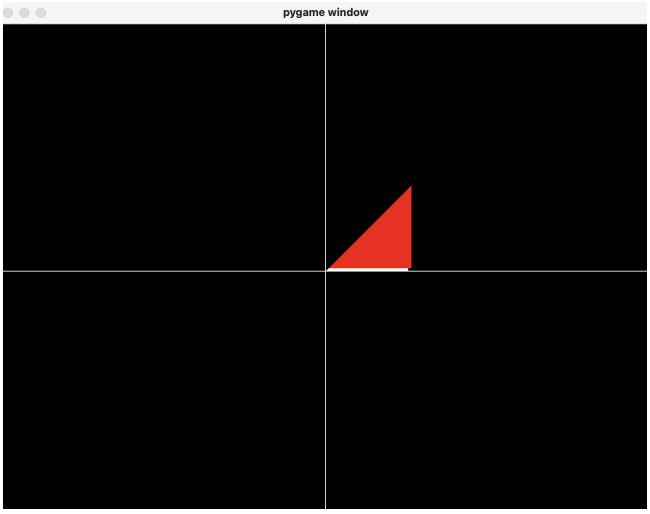
```
Enter operation number: 4
Enter reflection axis (x or y or x=y): x=y
Transformation Matrix:
[[0 1 0]
 [1 0 0]
 [0 0 1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



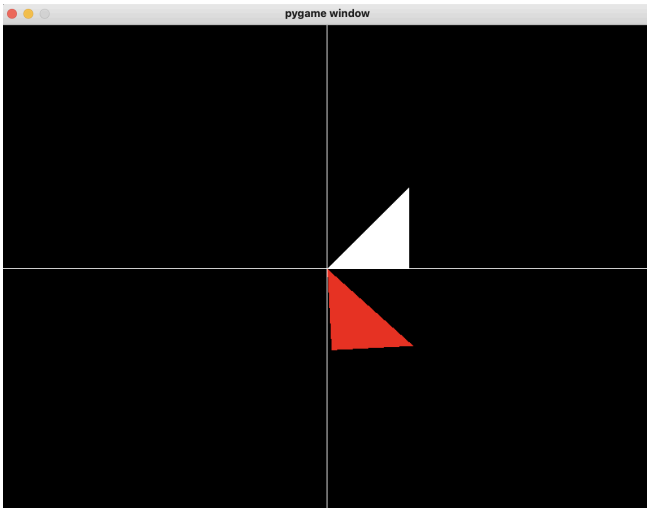

```
Enter operation number: 5
Enter shearing factors (kx,ky): 10,10
Transformation Matrix:
[[ 1. 10.  0.]
 [10.  1.  0.]
 [ 0.  0.  1.]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



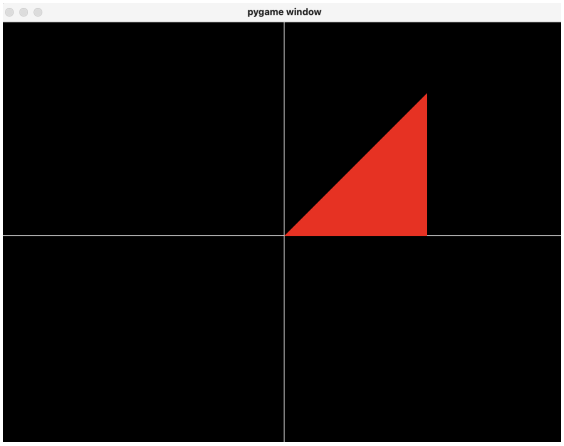

```
Enter operation number: 6
Enter operations to be composed (e.g., '1 2 3' for translation, rotation, scaling):
1
Enter translation values (tx,ty): 4,4
Composite Transformation Matrix:
[[1. 0. 4.]
 [0. 1. 4.]
 [0. 0. 1.]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
Enter operation number:
```



```
Enter operation number: 6
Enter operations to be composed (e.g., '1 2 3' for translation, rotation, scaling):
2
Enter rotation angle (in degrees): 273
Composite Transformation Matrix:
[[ 0.05233596  0.99862953  0.        ]
 [-0.99862953  0.05233596  0.        ]
 [ 0.          0.          1.        ]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
Enter operation number:
```




```
Enter operation number: 6
Enter operations to be composed (e.g., '1 2 3' for translation, rotation, scaling):
3
Enter scaling factors (sx,sy): 2,2
Composite Transformation Matrix:
[[2. 0. 0.]
 [0. 2. 0.]
 [0. 0. 1.]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
Enter operation number:
```




```
7. Exit
Enter operation number: 7
Exiting program.
(base) reewajkhanal.rk10@RK10 LAB03 %
```