Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Computer Graphics Lab Report 03

on

‘Circle Generating & Polygon Transforming Algorithms - Lab 03 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

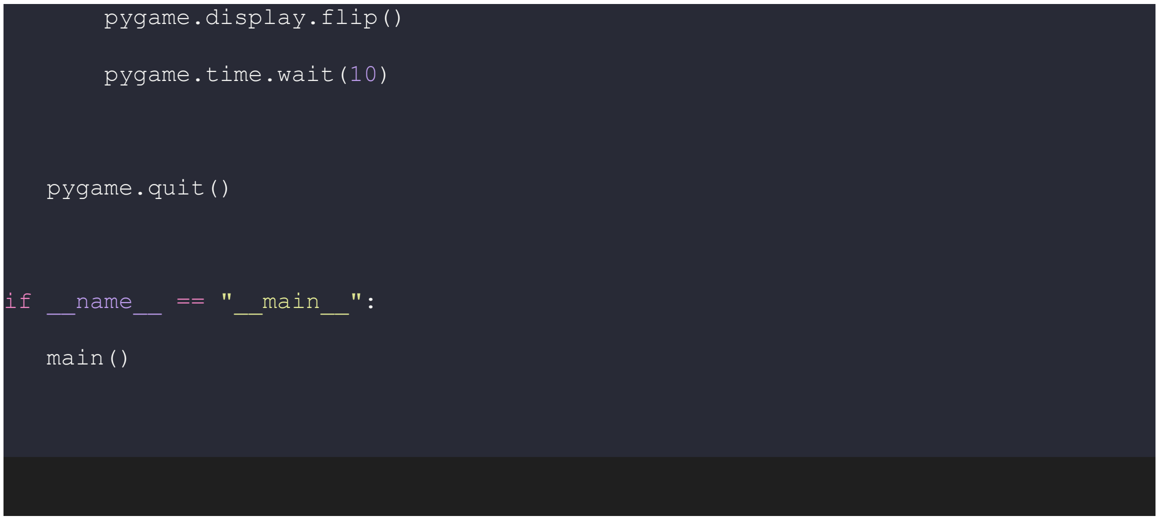
Submission Date: Friday 31 May 2024

Question No. 1 Write a Program to implement mid- point Circle Drawing Algorithm

Answer:

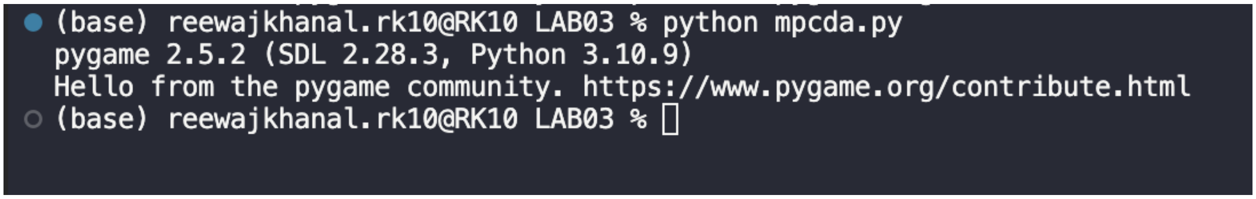
```
import pygame

from pygame.locals import *

from OpenGL.GL import *

from OpenGL.GLUT import *

from OpenGL.GLU import *


# Function to plot points in all octants

def plot_circle_points(x_center, y_center, x, y):

   glBegin(GL_POINTS)

   # Reflecting the points in all octants

   # Octant 1: Right Top Up

   glVertex2i(x_center + y, y_center + x)

   # Octant 2: Right Top

   glVertex2i(x_center + x, y_center + y)

   # Octant 3: Right Bottom Up

   glVertex2i(x_center + y, y_center - x)

   # Octant 4: Right Bottom

   glVertex2i(x_center + x, y_center - y)

   # Octant 5: Left Bottom

   glVertex2i(x_center - x, y_center - y)

   # Octant 6: Left Bottom Up

   glVertex2i(x_center - y, y_center - x)

   # Octant 7: Left Top Up

   glVertex2i(x_center - y, y_center + x)

   # Octant 8: Left Top

   glVertex2i(x_center - x, y_center + y)
```

```
    glEnd()

# Mid-Point Circle Drawing Algorithm
def midpoint_circle(x_center, y_center, radius):
    x = 0
    y = radius
    d = 1 - radius  # Decision parameter
    plot_circle_points(x_center, y_center, x, y)

    while x < y:
        if d < 0:
            # Move to the right
            d = d + 2 * x + 3
        else:
            # Move to the right and down
            d = d + 2 * (x - y) + 5
            y -= 1
        x += 1
        plot_circle_points(x_center, y_center, x, y)

def draw_circle(x_center,y_center,radius):

    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 1.0, 1.0)  # Set color to white
    # Draw the axes
    draw_axes()
     midpoint_circle(x_center, y_center, radius)  # Draw circle with radius 100 at origin
    glFlush()

def draw_axes():
    glBegin(GL_LINES)

    # Draw X axis

    glVertex2i(-400, 0)

    glVertex2i(400, 0)

    # Draw Y axis

    glVertex2i(0, -300)

    glVertex2i(0, 300)

    glEnd()



def get_input():

    x_center=int(input("Enter x coordinate of origin"))

    y_center=int(input("Enter x coordinate of origin"))

    radius=int(input("Enter radius of circle"))

    return x_center,y_center,radius



def main():

    pygame.init()

    display = (800, 600)

    pygame.display.set_mode(display, DOUBLEBUF | OPENGL)

    gluOrtho2D(-400, 400, -300, 300)  # Set up 2D coordinate system

    # x_center,y_center,radius=get_input()

    running = True

    while running:

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                running = False


        draw_circle(x_center=50,y_center=50,radius=100)
```



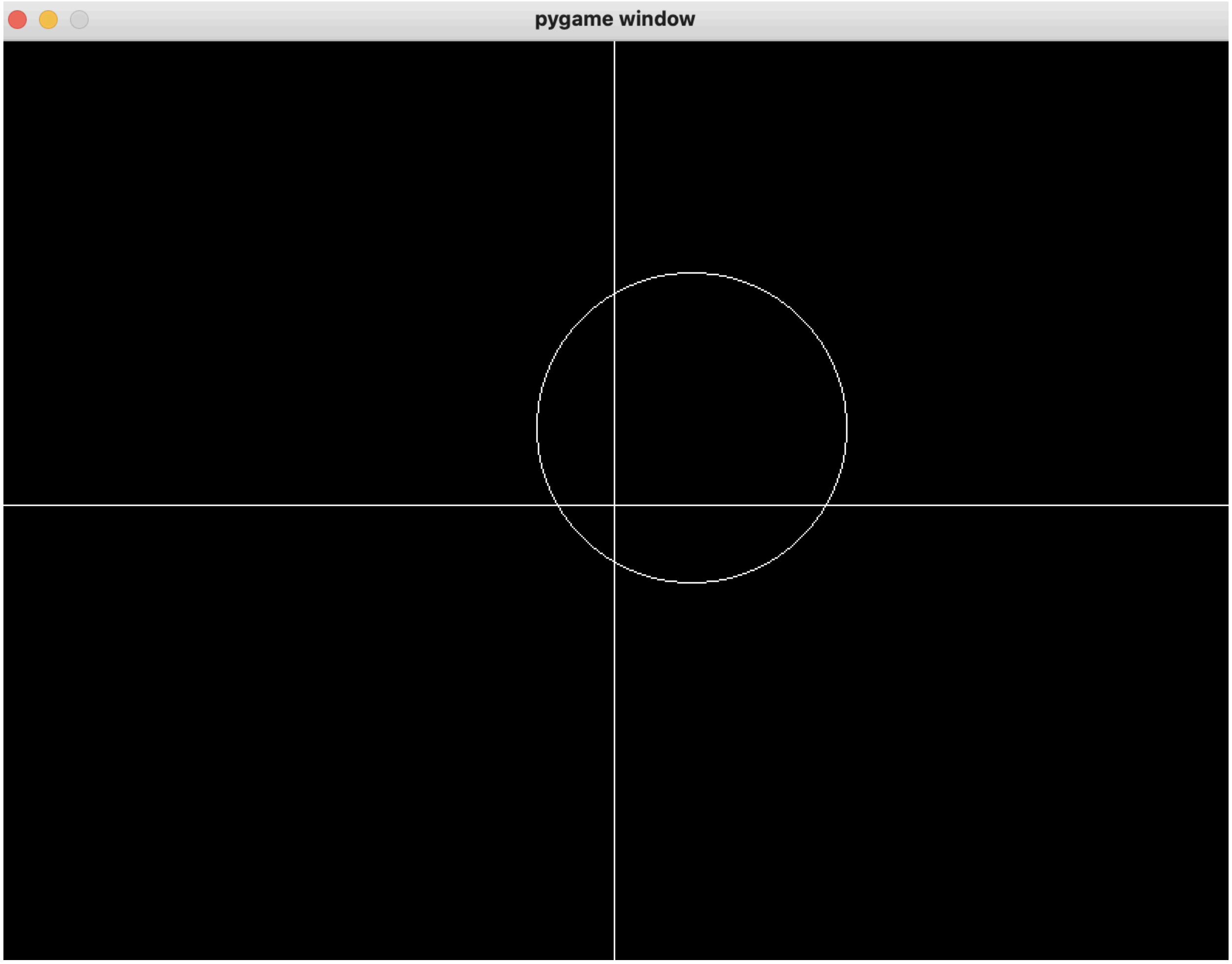
```
        pygame.display.flip()

        pygame.time.wait(10)


    pygame.quit()


if __name__ == "__main__":

    main()
```

Input:



```
● (base) reewajkhanal.rk10@RK10 LAB03 % python mpcda.py
  pygame 2.5.2 (SDL 2.28.3, Python 3.10.9)
  Hello from the pygame community. https://www.pygame.org/contribute.html
○ (base) reewajkhanal.rk10@RK10 LAB03 % 
```

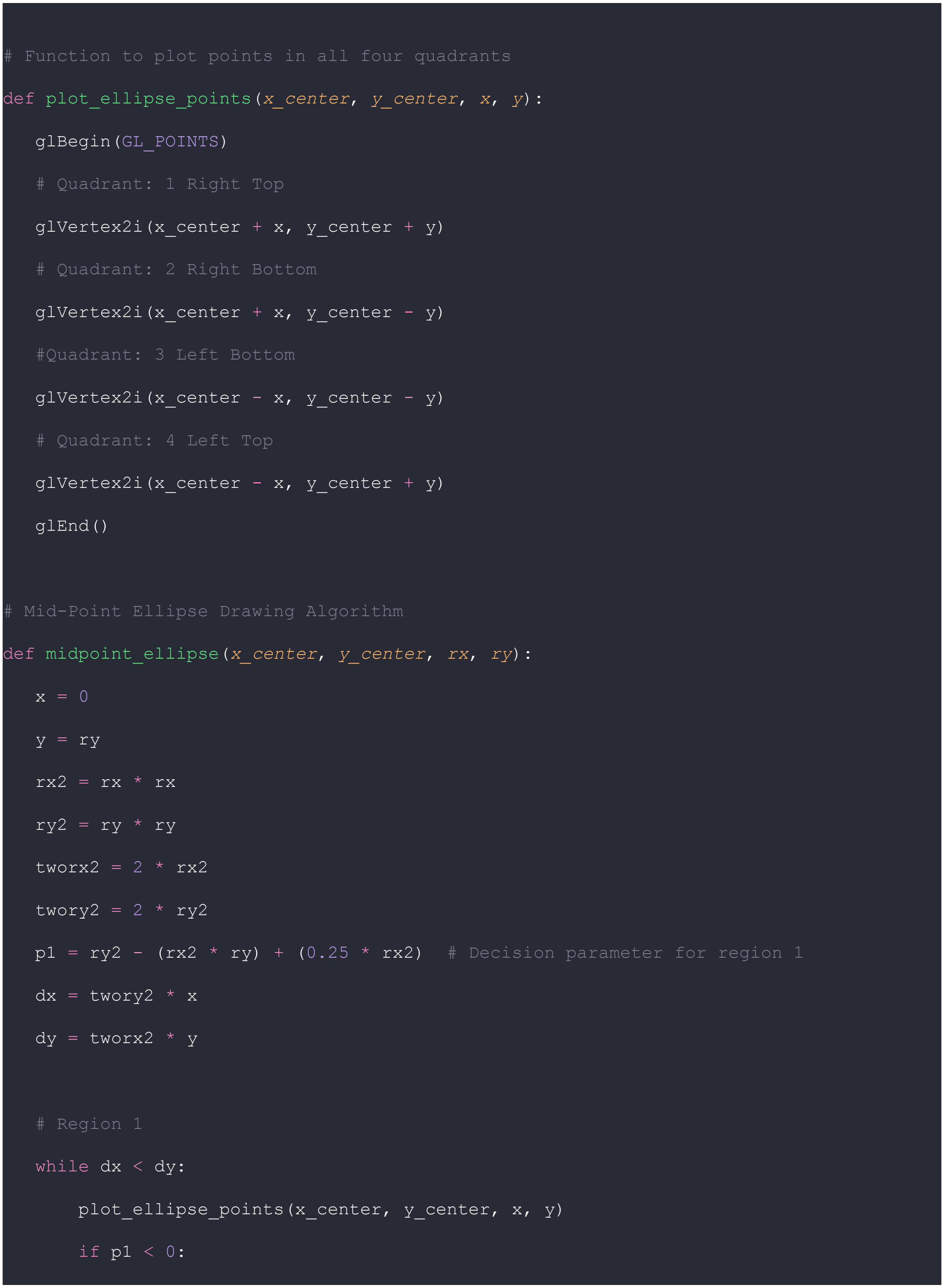
Output Generated:



Question No. 2 Write a Program to implement mid- point Ellipse Drawing Algorithm

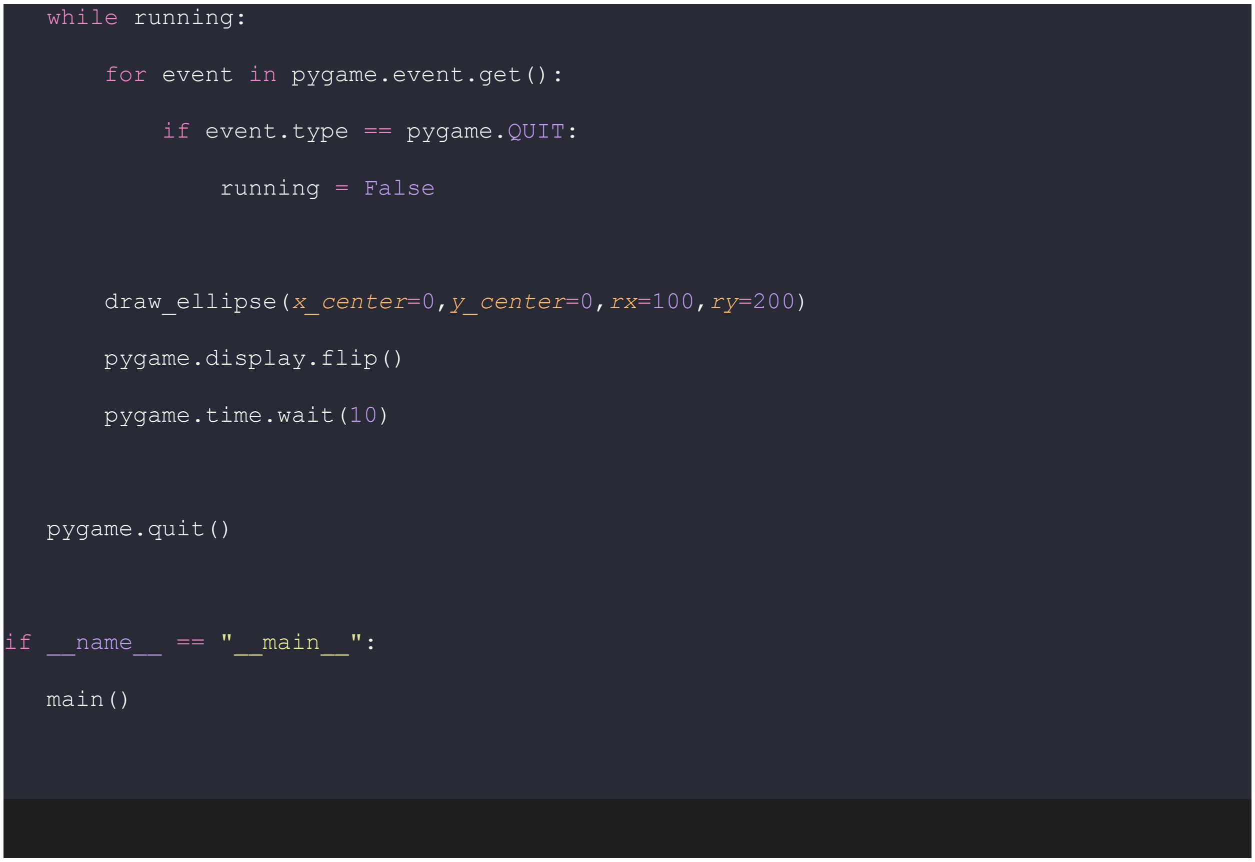
Answer:

```
import pygame

from pygame.locals import *

from OpenGL.GL import *

from OpenGL.GLUT import *

from OpenGL.GLU import *
```



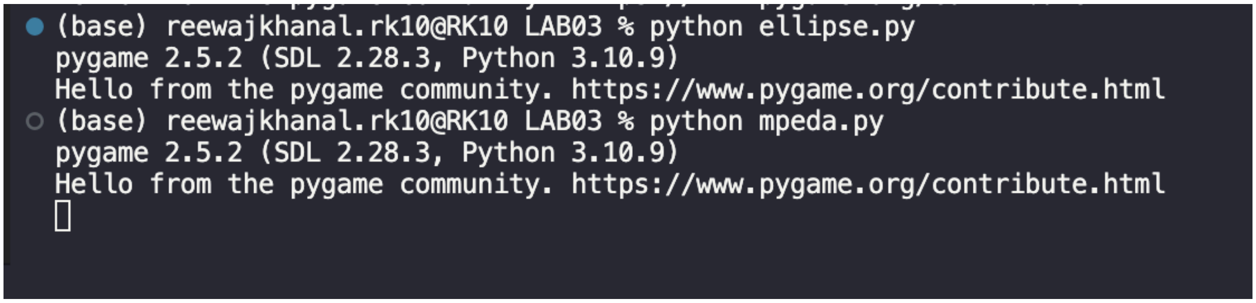
```
# Function to plot points in all four quadrants
def plot_ellipse_points(x_center, y_center, x, y):
   glBegin(GL_POINTS)
   # Quadrant: 1 Right Top
   glVertex2i(x_center + x, y_center + y)
   # Quadrant: 2 Right Bottom
   glVertex2i(x_center + x, y_center - y)
   #Quadrant: 3 Left Bottom
   glVertex2i(x_center - x, y_center - y)
   # Quadrant: 4 Left Top
   glVertex2i(x_center - x, y_center + y)
   glEnd()

# Mid-Point Ellipse Drawing Algorithm
def midpoint_ellipse(x_center, y_center, rx, ry):
   x = 0
   y = ry
   rx2 = rx * rx
   ry2 = ry * ry
   tworx2 = 2 * rx2
   twory2 = 2 * ry2
   p1 = ry2 - (rx2 * ry) + (0.25 * rx2)  # Decision parameter for region 1
   dx = twory2 * x
   dy = tworx2 * y

   # Region 1
   while dx < dy:
       plot_ellipse_points(x_center, y_center, x, y)
       if p1 < 0:
           x += 1
           dx += twory2
           p1 += dx + ry2
       else:
           x += 1
           y -= 1
           dx += twory2
           dy -= tworx2
           p1 += dx - dy + ry2

   # Region 2
   # Decision Parameter for region 2
   p2 = (ry2 * (x + 0.5) * (x + 0.5)) + (rx2 * (y - 1) * (y - 1)) - (rx2 * ry2)
   while y >= 0:
       plot_ellipse_points(x_center, y_center, x, y)
       if p2 > 0:
           y -= 1
           dy -= tworx2
           p2 += rx2 - dy
       else:
           x += 1
           y -= 1
           dx += twory2
           dy -= tworx2
           p2 += dx - dy + rx2


def draw_axes():
   glBegin(GL_LINES)
   # Draw X axis
   glVertex2i(-400, 0)
```

```
   glVertex2i(400, 0)
   # Draw Y axis
   glVertex2i(0, -300)
   glVertex2i(0, 300)
   glEnd()

def draw_ellipse(x_center=0,y_center=0,rx=100,ry=50):
   glClear(GL_COLOR_BUFFER_BIT)
   glColor3f(1.0, 1.0, 1.0)  # Set color to white
   # Draw the axes
   draw_axes()

   # Draw the ellipse
   midpoint_ellipse(x_center, y_center, rx, ry)  # Draw ellipse
   glFlush()

def get_input():
   x_center = int(input("Enter x coordinate of origin: "))
   y_center = int(input("Enter y coordinate of origin: "))
   rx = int(input("Enter x radius of ellipse: "))
   ry = int(input("Enter y radius of ellipse: "))
   return x_center, y_center, rx, ry

def main():
   pygame.init()
   display = (800, 600)
   pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
   gluOrtho2D(-400, 400, -300, 300)  # Set up 2D coordinate system
   # x_center,y_center,rx,ry=get_input()
   running = True
```



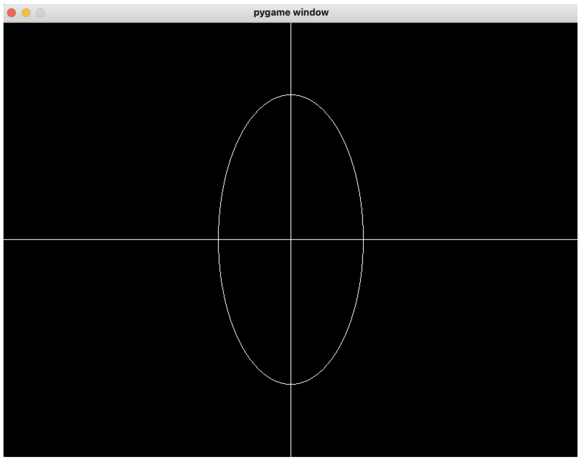
```
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

        draw_ellipse(x_center=0,y_center=0,rx=100,ry=200)
        pygame.display.flip()
        pygame.time.wait(10)

    pygame.quit()

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

Input:



```
● (base) reewajkhanal.rk10@RK10 LAB03 % python ellipse.py
  pygame 2.5.2 (SDL 2.28.3, Python 3.10.9)
  Hello from the pygame community. https://www.pygame.org/contribute.html
○ (base) reewajkhanal.rk10@RK10 LAB03 % python mpeda.py
  pygame 2.5.2 (SDL 2.28.3, Python 3.10.9)
  Hello from the pygame community. https://www.pygame.org/contribute.html
  ▯
```

Output:



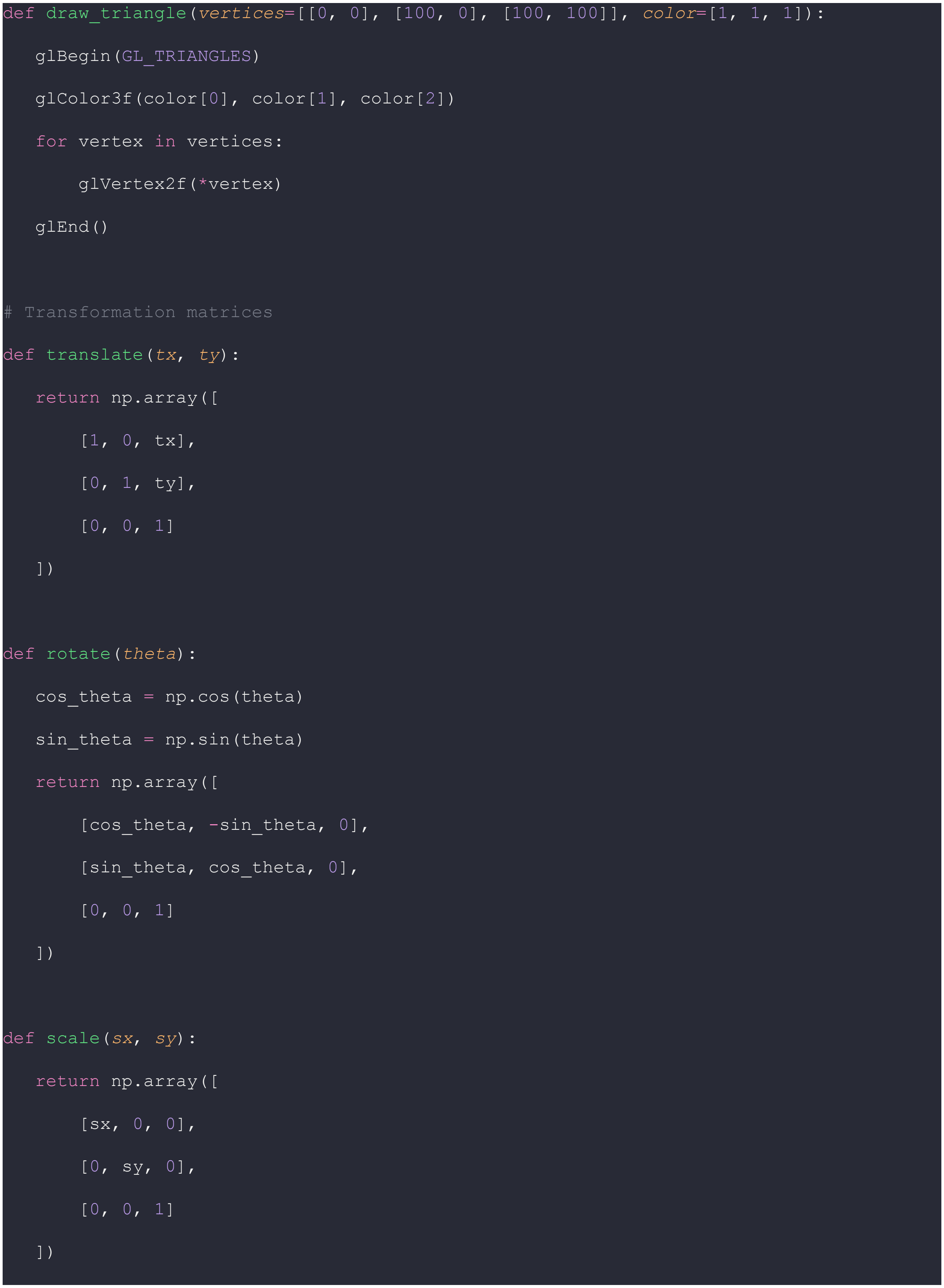
Question No. 3 Write a Program to implement:

- 2D Translation
- 2D Rotation
- 2D Scaling

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

Answer:

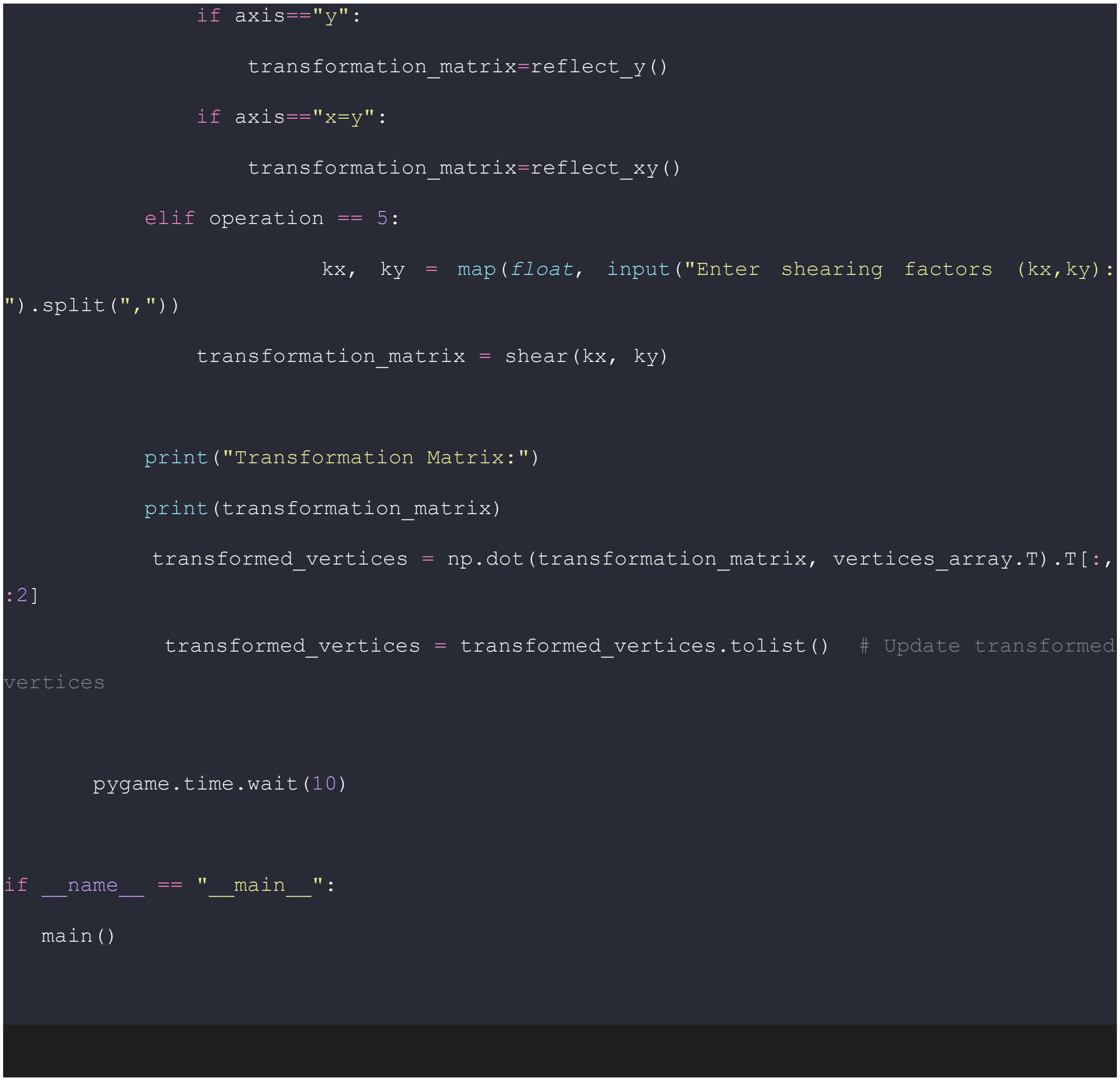
```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import numpy as np

# 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]
   ])
```

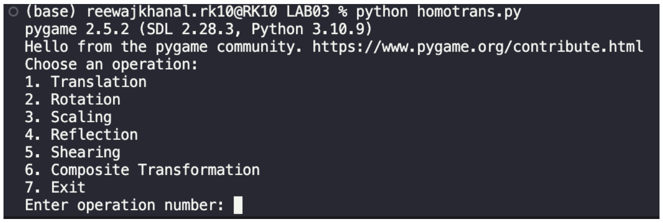
```
def reflect_x():

   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):
```

```python
    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()
```

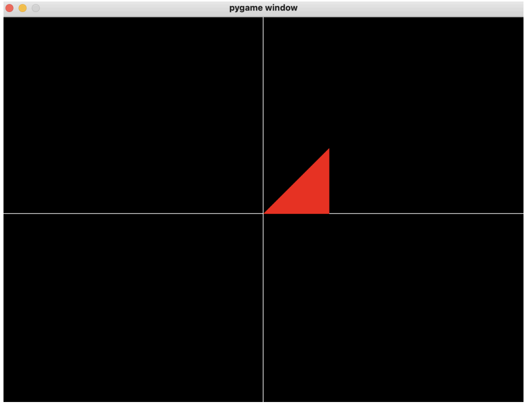


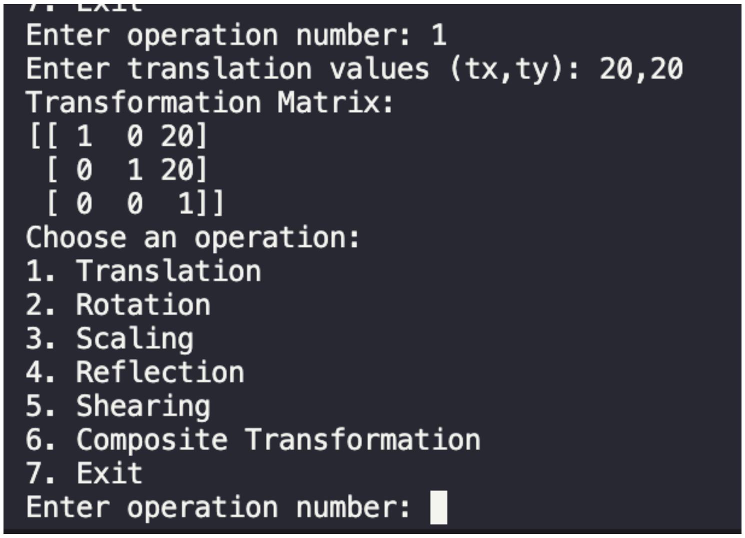
```
                if axis=="y":
                    transformation_matrix=reflect_y()
                if axis=="x=y":
                    transformation_matrix=reflect_xy()
            elif operation == 5:
                kx, ky = map(float, input("Enter shearing factors (kx,ky): ").split(","))
  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

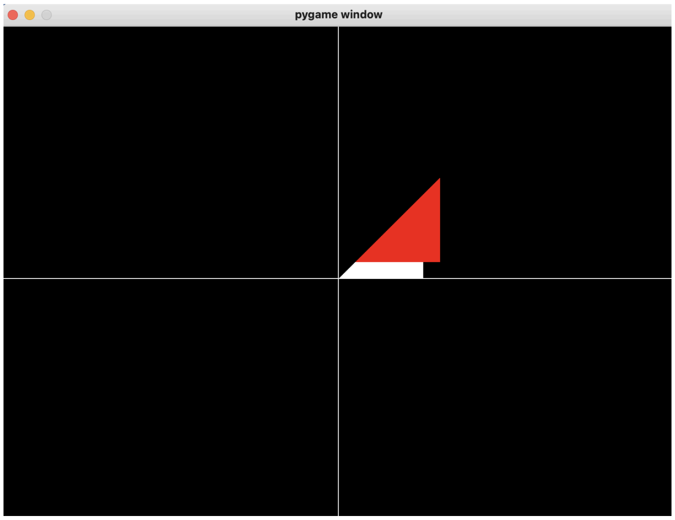


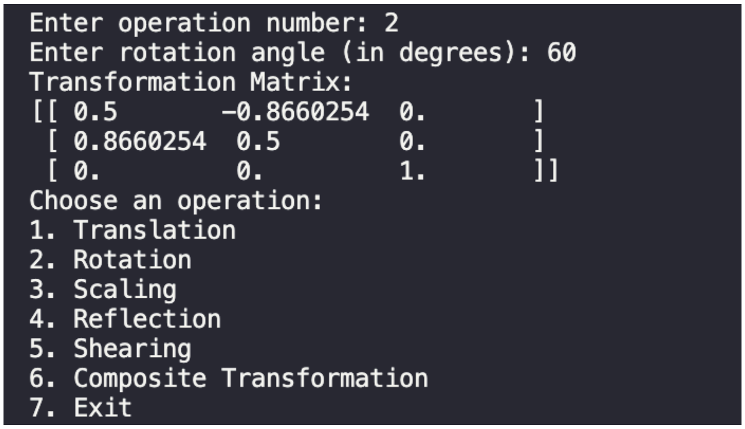
```
○ (base) reewajkhanal.rk10@RK10 LAB03 % python homotrans.py
  pygame 2.5.2 (SDL 2.28.3, Python 3.10.9)
  Hello from the pygame community. https://www.pygame.org/contribute.html
  Choose an operation:
  1. Translation
  2. Rotation
  3. Scaling
  4. Reflection
  5. Shearing
  6. Composite Transformation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



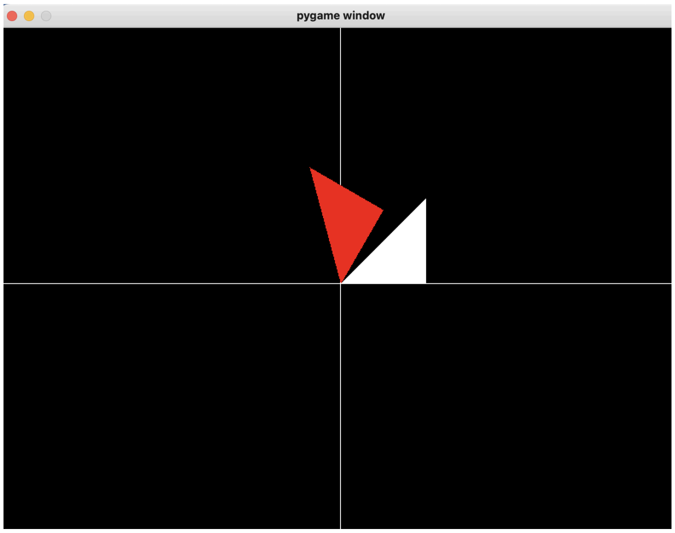


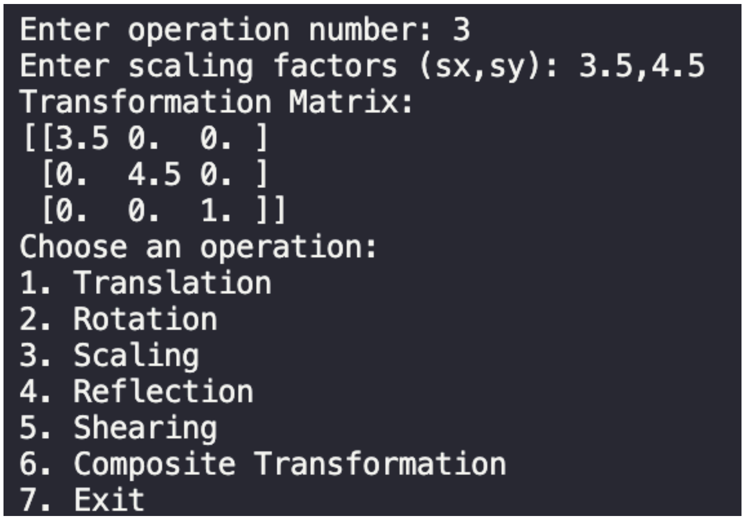
```
7. Exit
Enter operation number: 1
Enter translation values (tx,ty): 20,20
Transformation Matrix:
[[ 1  0 20]
 [ 0  1 20]
 [ 0  0  1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
Enter operation number: 
```



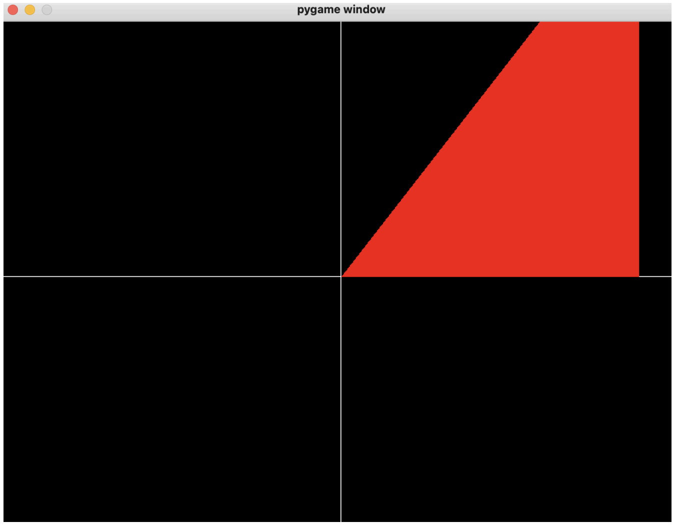


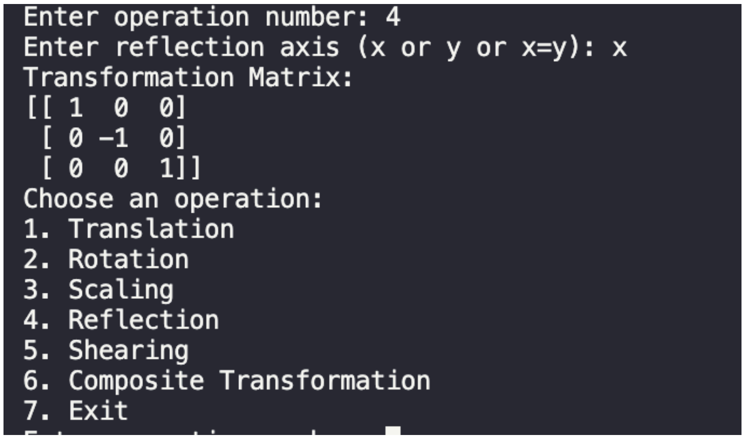
```
Enter operation number: 2
Enter rotation angle (in degrees): 60
Transformation Matrix:
[[ 0.5       -0.8660254  0.        ]
 [ 0.8660254  0.5        0.        ]
 [ 0.         0.         1.        ]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite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



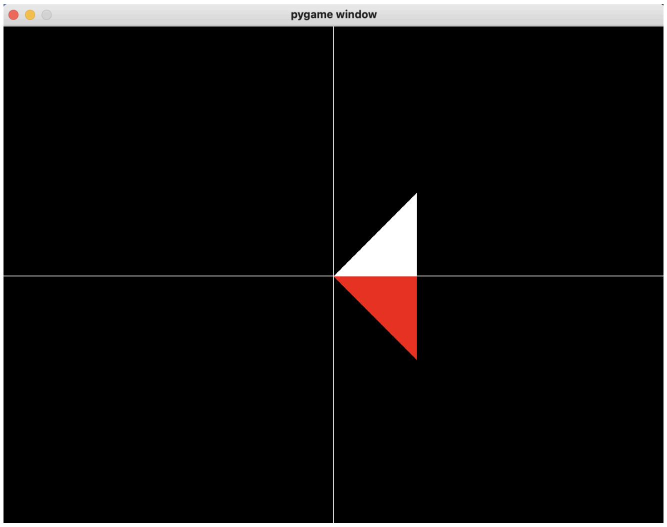


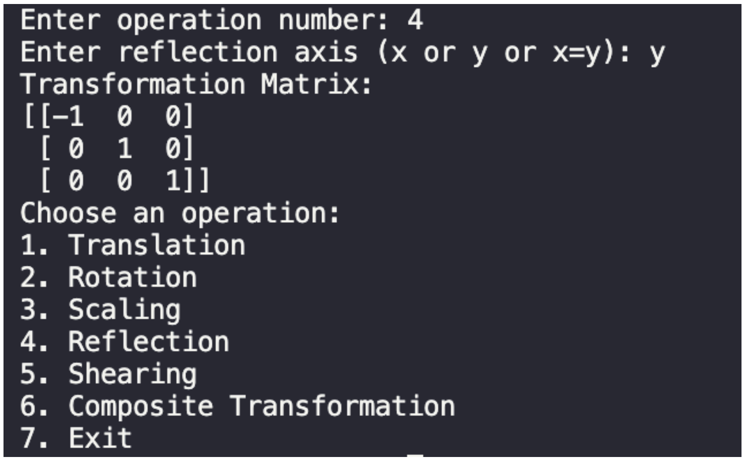
```
Enter operation number: 3
Enter scaling factors (sx,sy): 3.5,4.5
Transformation Matrix:
[[3.5 0.  0. ]
 [0.  4.5 0. ]
 [0.  0.  1. ]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



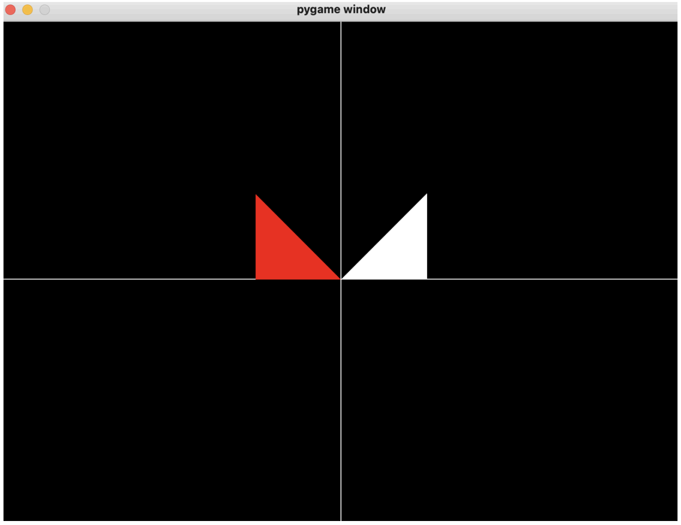


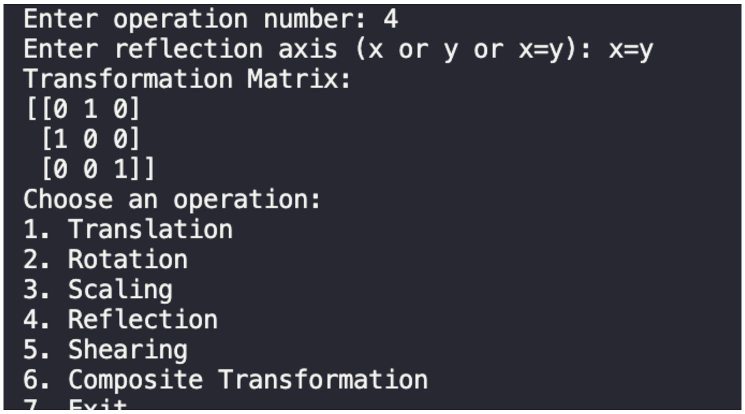
```
Enter operation number: 4
Enter reflection axis (x or y or x=y): x
Transformation Matrix:
[[ 1  0  0]
 [ 0 -1  0]
 [ 0  0  1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



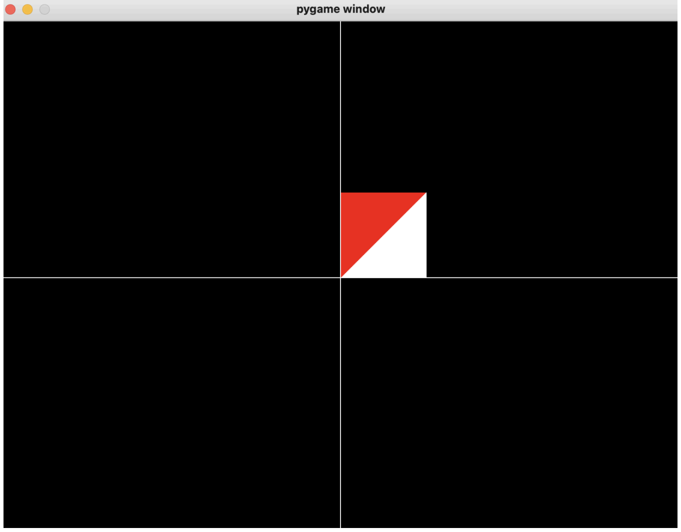


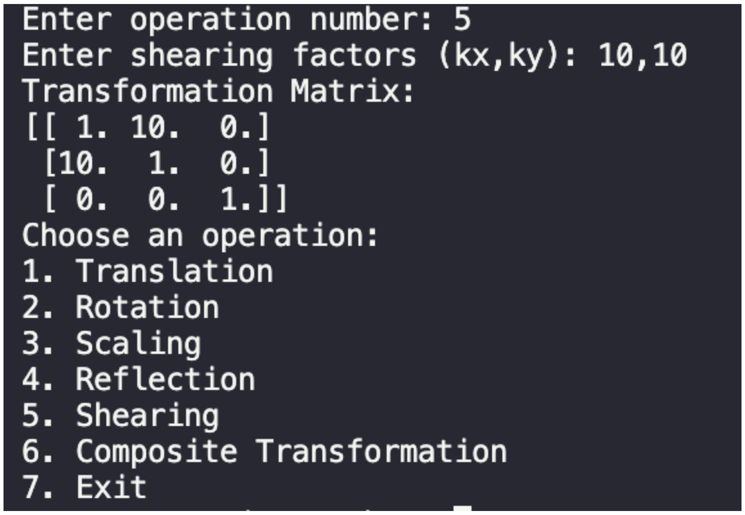
```
Enter operation number: 4
Enter reflection axis (x or y or x=y): y
Transformation Matrix:
[[-1  0  0]
 [ 0  1  0]
 [ 0  0  1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



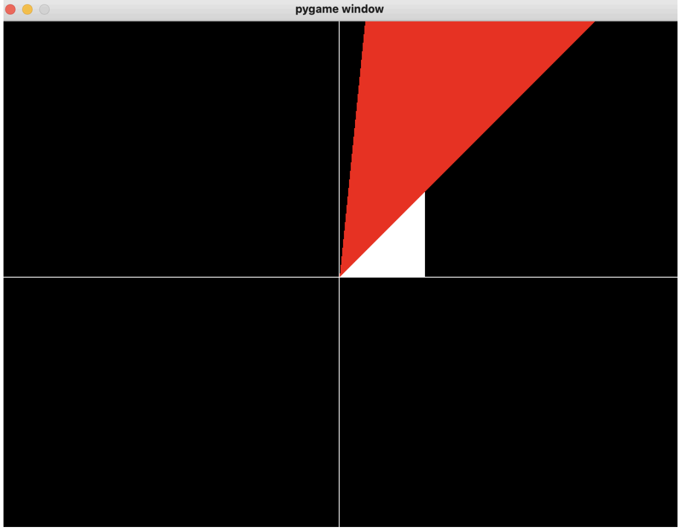


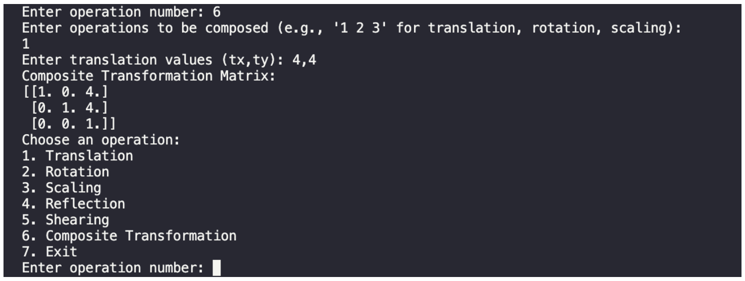
```
Enter operation number: 4
Enter reflection axis (x or y or x=y): x=y
Transformation Matrix:
[[0 1 0]
 [1 0 0]
 [0 0 1]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



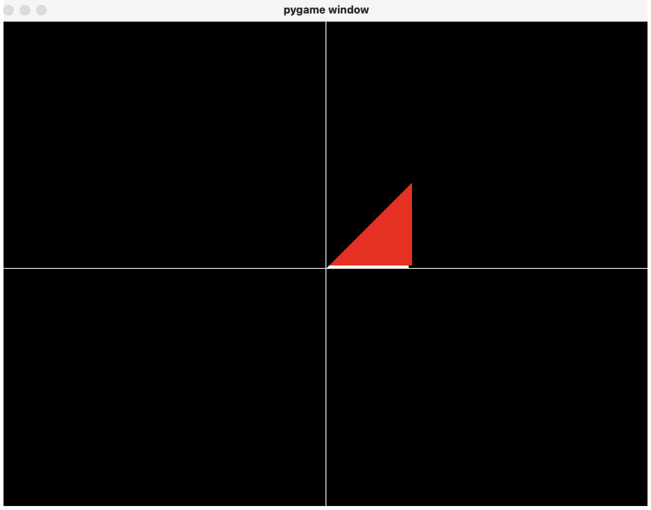


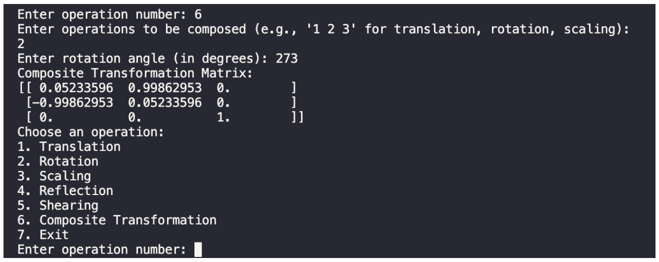
```
Enter operation number: 5
Enter shearing factors (kx,ky): 10,10
Transformation Matrix:
[[ 1. 10.  0.]
 [10.  1.  0.]
 [ 0.  0.  1.]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Composite Transformation
7. Exit
```



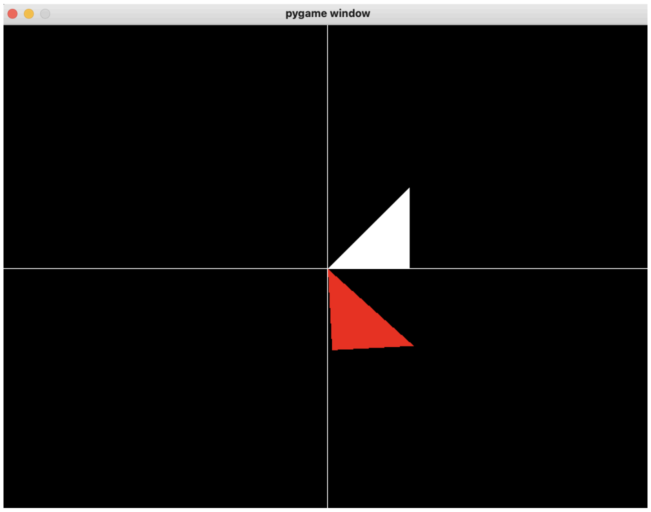


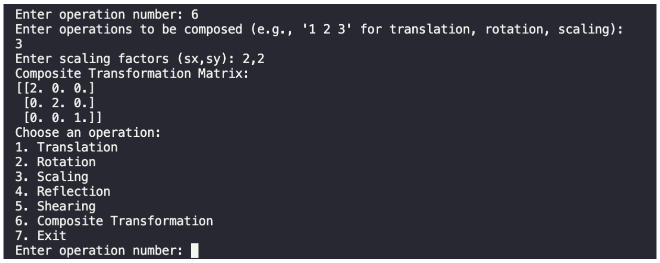
```
Enter operation number: 6
Enter operations to be composed (e.g., '1 2 3' for translation, rotation, scaling):
1
Enter translation values (tx,ty): 4,4
Composite Transformation Matrix:
[[1. 0. 4.]
 [0. 1. 4.]
 [0. 0. 1.]]
Choose an operation:
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing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



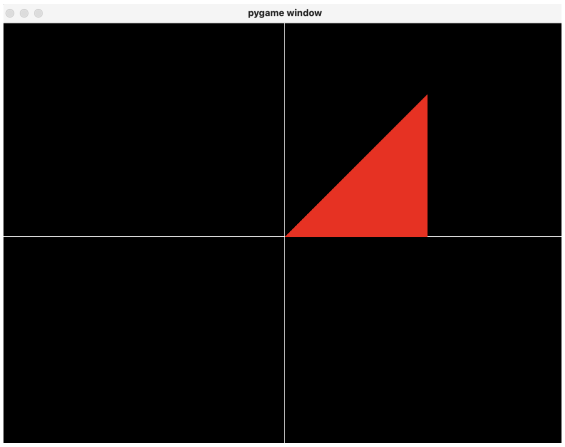


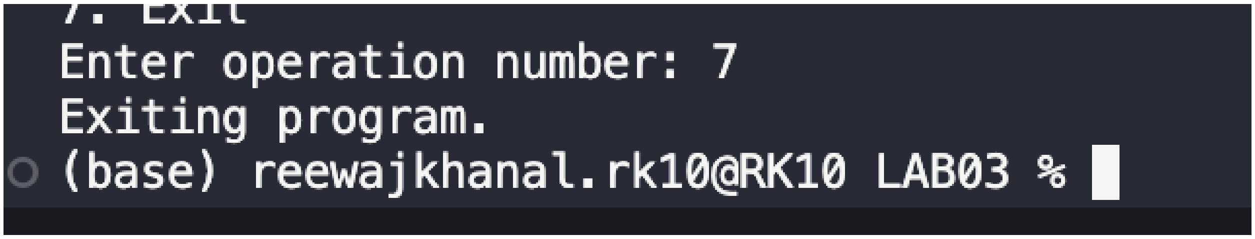
```
Enter operation number: 6
Enter operations to be composed (e.g., '1 2 3' for translation, rotation, scaling):
2
Enter rotation angle (in degrees): 273
Composite Transformation Matrix:
[[ 0.05233596  0.99862953  0.        ]
 [-0.99862953  0.05233596  0.        ]
 [ 0.          0.          1.        ]]
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 % 
```