

Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre



Lab report 4

[Code No: COMP 342]

Submitted by

Ashutosh Singh Khadka(22)

Submitted to:

Department of Computer Science and Engineering

Submission Date: 9th August, 2021

Table of Contents

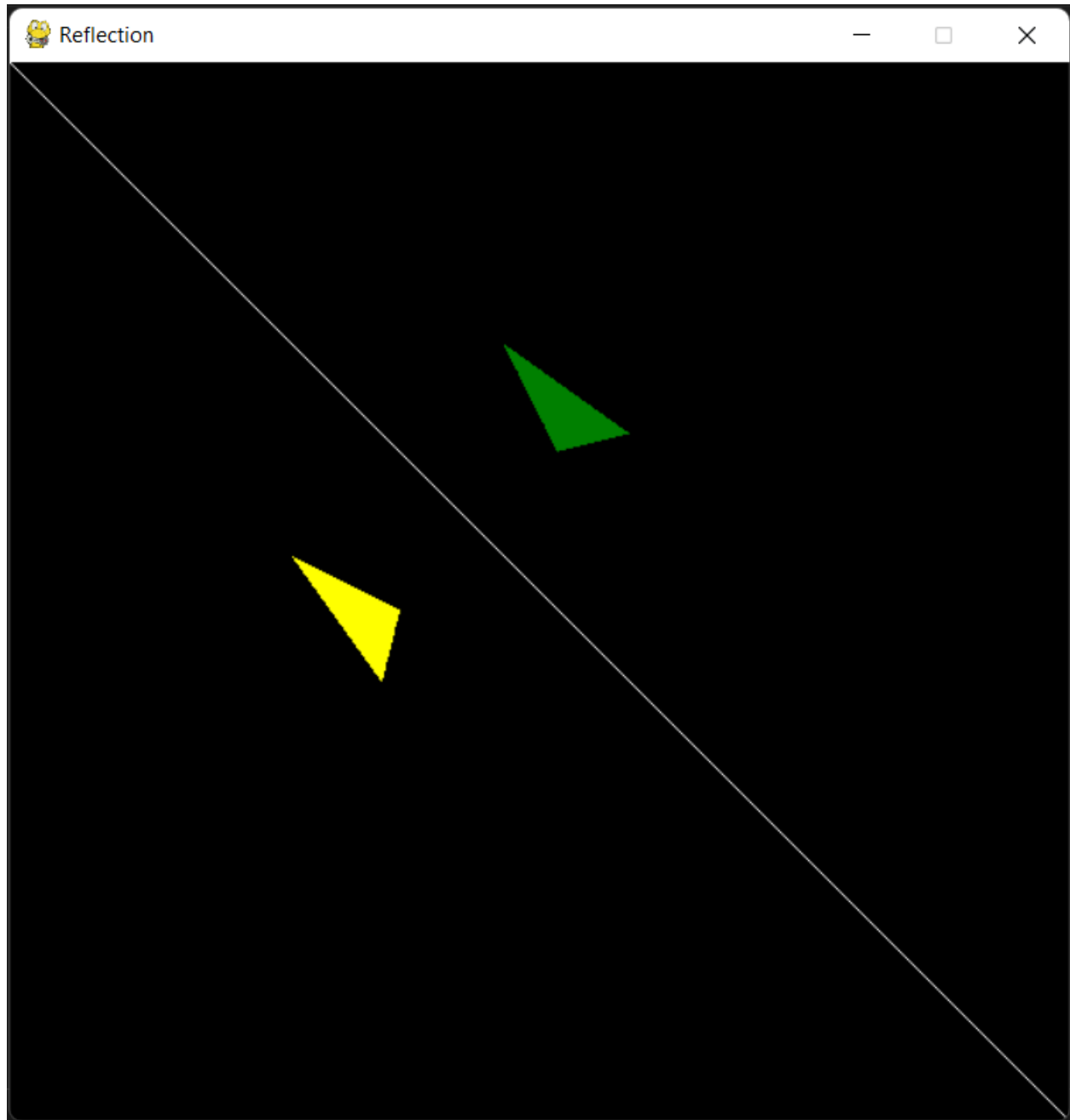
Reflection:	2
Implementation:	2
Output:	3
Algorithm:	4
Shearing:	5
Implementation:	5
Output:	6
Algorithm:	7
Translation:	8
Implementation:	8
Output:	9
Algorithm:	10
Rotation:	11
Implementation:	11
Output:	12
Algorithm:	13
Scaling:	14
Implementation:	14
Output:	15
Algorithm:	16

Reflection:

Implementation:

```
1  import pygame
2  from pygame import gfxdraw
3  import numpy as np
4  |
5
6  WHITE = (255, 255, 255)
7  YELLOW = (255, 255, 0)
8  GREEN = (0,128,0)
9
10 isp = False
11 x1 = y1 = x2 = y2 = 0
12 ps = (x1, y1)
13 pe = (x2, y2)
14
15 def prepare_screen():
16     """
17     Create the initial screen.
18     """
19     pygame.init()
20     screen = pygame.display.set_mode((600, 600))
21     screen.fill((0,0,0))
22     pygame.display.set_caption("Reflection")
23     return screen
24
25 def reflection(x,y):
26     mat =([x],
27           [y],
28           [1])
29
30     transformMat = ([0,1,0],[1,0,0],[0,0,1])
31
32     translatedPoints = np.dot(transformMat, mat)
33
34     return translatedPoints[0],translatedPoints[1]
35
36 screen = prepare_screen()
37 gfxdraw.line(screen,0,0,1000,1000, WHITE)
38 a = (160,280)
39 b = (210,350)
40 c = (220,310)
41 gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
42 a = reflection(a[0],a[1])
43 b = reflection(b[0],b[1])
44 c = reflection(c[0],c[1])
45 gfxdraw.filled_polygon(screen, [a,b,c], GREEN)
46
47 while True:
48     for event in pygame.event.get():
49         if event.type == pygame.QUIT:
50             pygame.quit()
51             quit()
52     pygame.display.update()
```

Output:



Algorithm:

2D Reflection:-

Algorithm:-

Step 1: Given points starting point (x_1, y_1) and end point (x_2, y_2)

Step 2: Reflection about x-axis (matrix form) is:-

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 3: Matrix form of starting and end points are:-

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Step 4: Matrix multiplication for reflection:-

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \\ 1 & 1 \end{bmatrix}$$

i.e

$$\begin{aligned} x_1' &= x_1 & \& & x_2' &= x_2 \\ y_1' &= -y_1 & & & y_2' &= -y_2 \\ 1 &= 1 & & & 1 &= 1 \end{aligned}$$

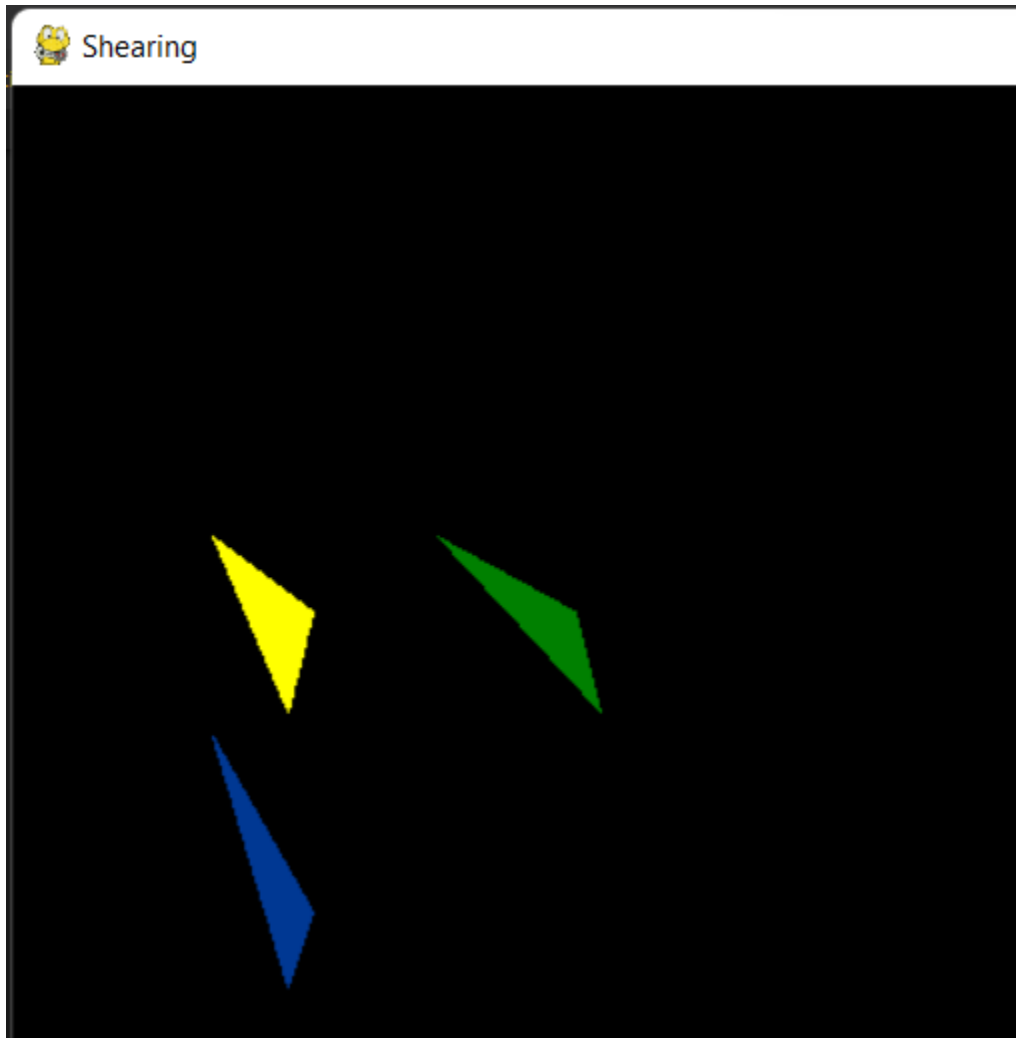
So, co-ordinates are $(x_1', y_1', 1)$ and $(x_2', y_2', 1)$.

Shearing:

Implementation:

```
1  import pygame
2  from pygame import gfxdraw
3  import numpy as np
4
5  (constant) YELLOW: tuple[Literal[255], Literal[255], Literal[0]]
6  YELLOW = (255, 255, 0)
7  GREEN = (0, 128, 0)
8  BLUE = (0, 56, 147)
9
10 isp = False
11 x1 = y1 = x2 = y2 = 0
12 ps = (x1, y1)
13 pe = (x2, y2)
14
15 def prepare_screen():
16     """
17     Create the initial screen.
18     """
19     pygame.init()
20     screen = pygame.display.set_mode((800, 800))
21     screen.fill((0, 0, 0))
22     pygame.display.set_caption("Shearing")
23     return screen
24
25 def shearX(x, y, shx):
26     mat = ([x],
27            [y],
28            [1])
29
30     transformMat = ([1, shx, 0], [0, 1, 0], [0, 0, 1])
31
32     translatedPoints = np.dot(transformMat, mat)
33
34     return translatedPoints[0], translatedPoints[1]
35
36 def shearY(x, y, shy):
37     mat = ([x],
38            [y],
39            [1])
40
41     transformMat = ([1, 0, 0], [shy, 1, 0], [0, 0, 1])
42
43     translatedPoints = np.dot(transformMat, mat)
44
45     return translatedPoints[0], translatedPoints[1]
46
47 screen = prepare_screen()
48 a = (80, 180)
49 b = (110, 250)
50 c = (120, 210)
51 gfxdraw.filled_polygon(screen, [a, b, c], YELLOW)
52 ax = shearX(a[0], a[1], 0.5)
53 bx = shearX(b[0], b[1], 0.5)
54 cx = shearX(c[0], c[1], 0.5)
55 gfxdraw.filled_polygon(screen, [ax, bx, cx], GREEN)
56
57 ay = shearY(a[0], a[1], 1)
58 by = shearY(b[0], b[1], 1)
59 cy = shearY(c[0], c[1], 1)
60 gfxdraw.filled_polygon(screen, [ay, by, cy], BLUE)
61
62 while True:
63     for event in pygame.event.get():
64         if event.type == pygame.QUIT:
65             pygame.quit()
66             quit()
67     pygame.display.update()
```

Output:



Algorithm:

Date _____
Page _____

2D shearing

Algorithm:-

Step 1: Given points starting point (x_1, y_1) and end points as (x_2, y_2)

Step 2: Given shearing along x -axis = sh_x
 shearing along y -axis = sh_y

Step 3: Mat'n form of shearing along sh_x and sh_y are:

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4: Mat'n form of starting point and end point.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Step 5: If shearing along x -axis

for starting point $\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$ for end point $\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$

i.e. $x_1' = x_1 + sh_x \cdot y_1$ i.e. $x_2' = x_2 + sh_x \cdot y_2$
 $y_1' = y_1$ $y_2' = y_2$
 $1 = 1$ $1 = 1$

new co-ordinates are $(x_1', y_1', 1)$ and $(x_2', y_2', 1)$

else

for starting point $\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$ for end point $\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$

i.e. $x_1' = x_1$ i.e. $x_2' = x_2$
 $y_1' = y_1 + sh_y \cdot x_1$ $y_2' = y_2 + sh_y \cdot x_2$
 $1 = 1$ $1 = 1$


So, new co-ordinates are $(x_1', y_1', 1)$ and $(x_2', y_2', 1)$

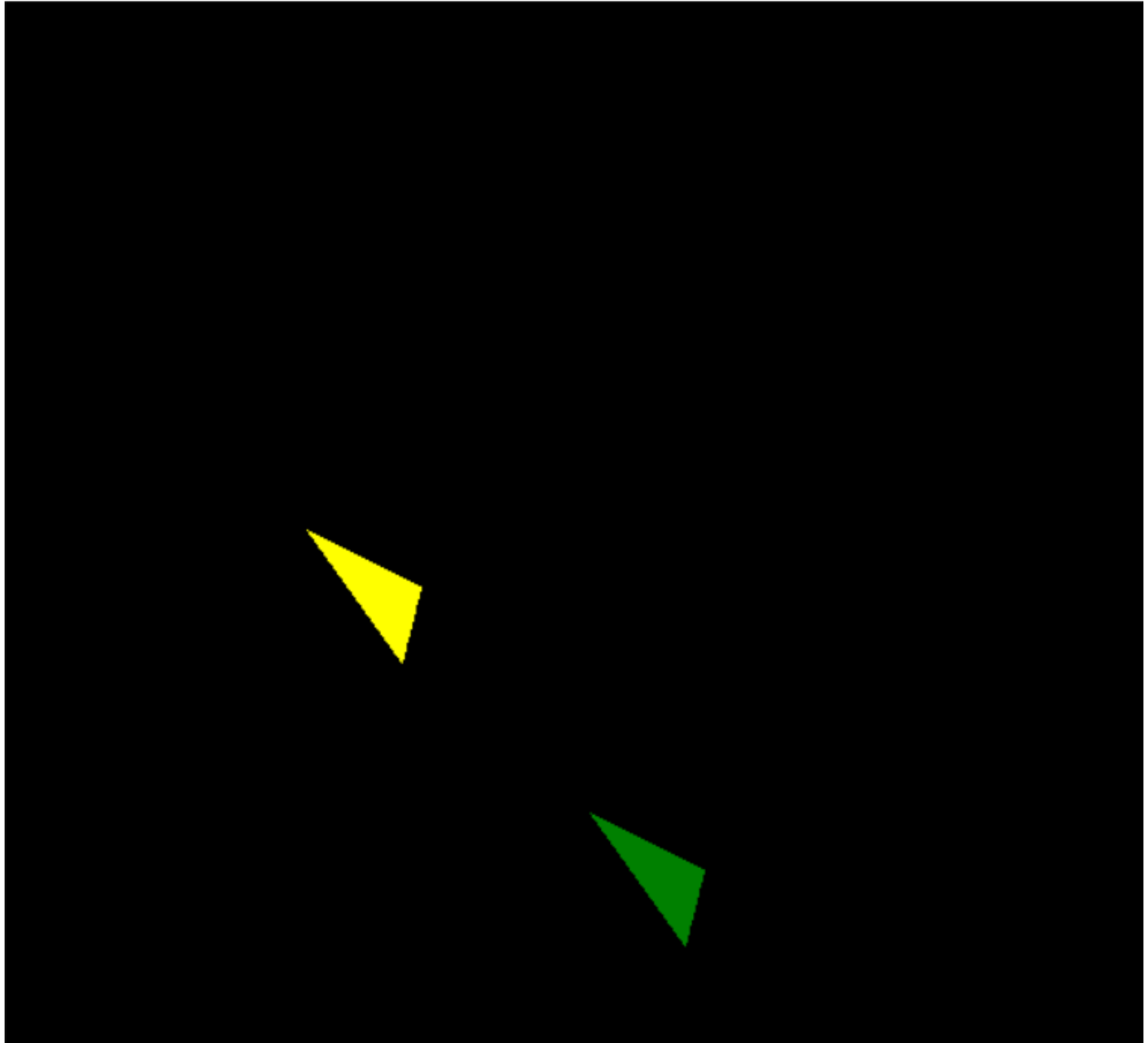
Translation:

Implementation:

```
1  import pygame
2  from pygame import gfxdraw
3  import numpy as np
4
5
6  WHITE = (255, 255, 255)
7  YELLOW = (255, 255, 0)
8  GREEN = (0,128,0)
9
10 isp = False
11 x1 = y1 = x2 = y2 = 0
12 ps = (x1, y1)
13 pe = (x2, y2)
14
15 def prepare_screen():
16     """
17     Create the initial screen.
18     """
19     pygame.init()
20     screen = pygame.display.set_mode((800, 800))
21     screen.fill((0,0,0))
22     pygame.display.set_caption("Translation")
23     return screen
24
25 def translate(x,y, tx, ty):
26     mat =([x],
27           [y],
28           [1])
29
30     transformMat = ([1,0,tx],[0,1,ty],[0,0,1])
31
32     translatedPoints = np.dot(transformMat, mat)
33
34     return translatedPoints[0],translatedPoints[1]
35
36 screen = prepare_screen()
37 a = (160,280)
38 b = (210,350)
39 c = (220,310)
40 gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
41 a = translate(a[0],a[1], 150, 150)
42 b = translate(b[0],b[1], 150, 150)
43 c = translate(c[0],c[1], 150, 150)
44 gfxdraw.filled_polygon(screen, [a,b,c], GREEN)
45
46 while True:
47     for event in pygame.event.get():
48         if event.type == pygame.QUIT:
49             pygame.quit()
50             quit()
51     pygame.display.update()
```

Output:

 Translation



Algorithm:

2D Translation:-

Algorithm:-

Step 1: Given points of line (x_1, y_1) as starting point and (x_2, y_2) as end points.

Step 2: Given translation point (t_x, t_y)

Step 3: Arranging above translation point into 3×3 matrix.

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4: Arranging about starting & end points into 3×1 matrix

$$\begin{array}{l} \text{for starting} \\ \text{point} \end{array} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad \begin{array}{l} \text{for end} \\ \text{points} \end{array} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Step 5: Carry out matrix multiplication:-

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\text{i.e. } x_1' = x_1 + t_x$$

$$y_1' = y_1 + t_y$$

$$1 = 1$$

new co-ordinates of starting point $(x_1', y_1', 1)$

Step 6: Carry out translation multiplication for end points.

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

$$x_2' = t_x + x_2$$

$$y_2' = y_2 + t_y$$

$$1 = 1$$

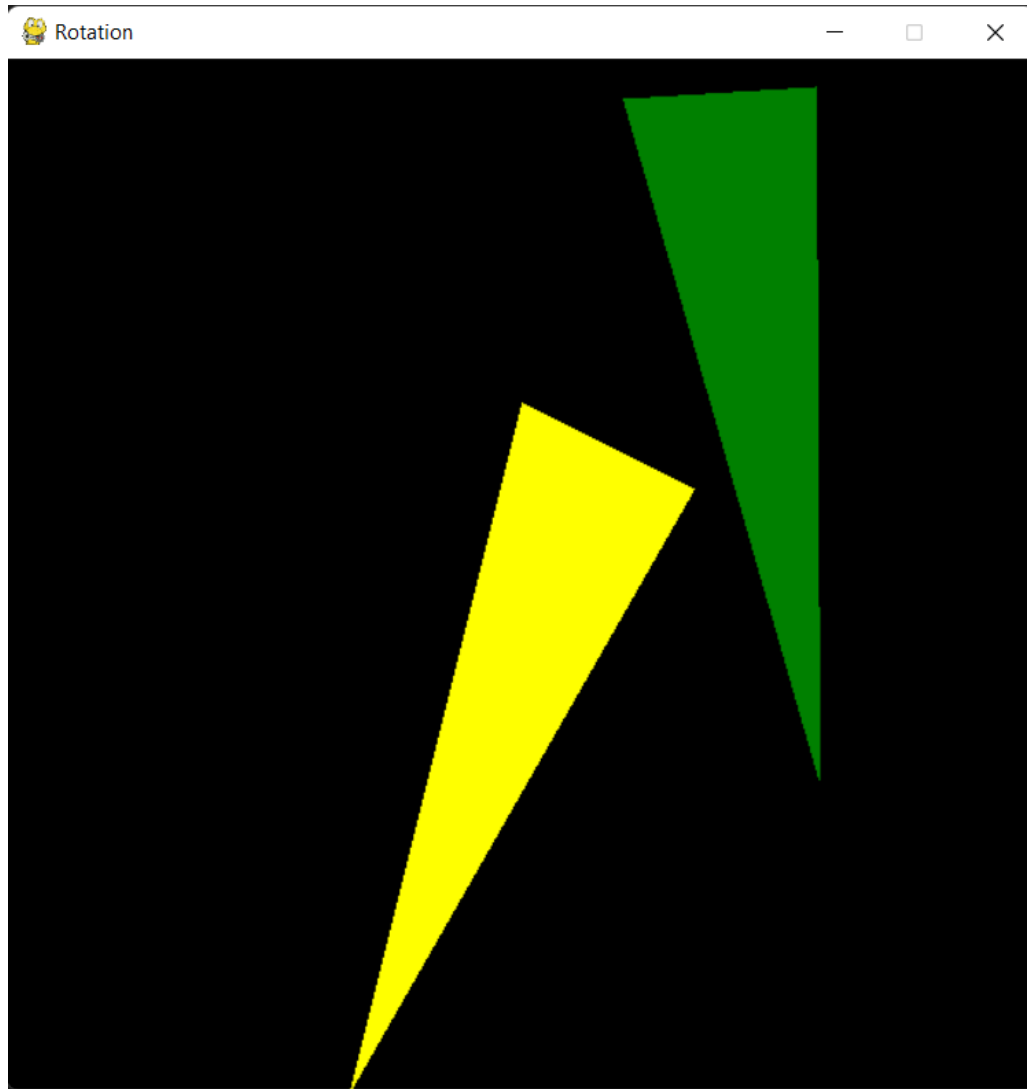
So, new co-ordinates for end points $(x_2', y_2', 1)$.

Rotation:

Implementation:

```
1  import pygame
2  from pygame import gfxdraw
3  import numpy as np
4  WHITE = (255, 255, 255)
5  YELLOW = (255, 255, 0)
6  GREEN = (0,128,0)
7
8  isp = False
9  x1 = y1 = x2 = y2 = 0
10 ps = (x1, y1)
11 pe = (x2, y2)
12 def prepare_screen():
13     """
14     Create the initial screen.
15     """
16     pygame.init()
17     screen = pygame.display.set_mode((600, 600))
18     screen.fill((0,0,0))
19     pygame.display.set_caption("Rotation")
20     return screen
21 screen = prepare_screen()
22 def rotat(x,y,a):
23     a = np.math.radians(a)
24     m =([x],
25         [y],
26         [1])
27     transformM = ([np.math.cos(a),-np.math.sin(a),0],[np.math.sin(a),np.math.cos(a),0],[0,0,1])
28     translatedPoints = np.dot(transformM, m)
29     return translatedPoints[0],translatedPoints[1]
30 m = (200,600)
31 n = (300,200)
32 o = (400,250)
33 gfxdraw.filled_polygon(screen, [m,n,o], YELLOW)
34 m = rotat(m[0],m[1], 330)
35 n = rotat(n[0],n[1], 330)
36 o = rotat(o[0],o[1], 330)
37 gfxdraw.filled_polygon(screen, [m,n,o], GREEN)
38 while True:
39     for event in pygame.event.get():
40         if event.type == pygame.QUIT:
41             pygame.quit()
42             quit()
43     pygame.display.update()
```

Output:



Algorithm:

2D Rotation:-

Algorithm:-

- Step 1: starting point (x_1, y_1) and end point (x_2, y_2)
Step 2: Rotation by angle θ
Step 3: Arranging above rotation angle in 3×3 matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Step 4: starting point & end point in matrix.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

- Step 5: Carry out matrix multiplication for rotation

For starting point $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$

i.e. $x_1' = x_1 \cos\theta - y_1 \sin\theta$

$y_1' = x_1 \sin\theta + y_1 \cos\theta$

$1 = 1$

and

For end points $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$

i.e. $x_2' = x_2 \cos\theta - y_2 \sin\theta$

$y_2' = x_2 \sin\theta + y_2 \cos\theta$

$1 = 1$

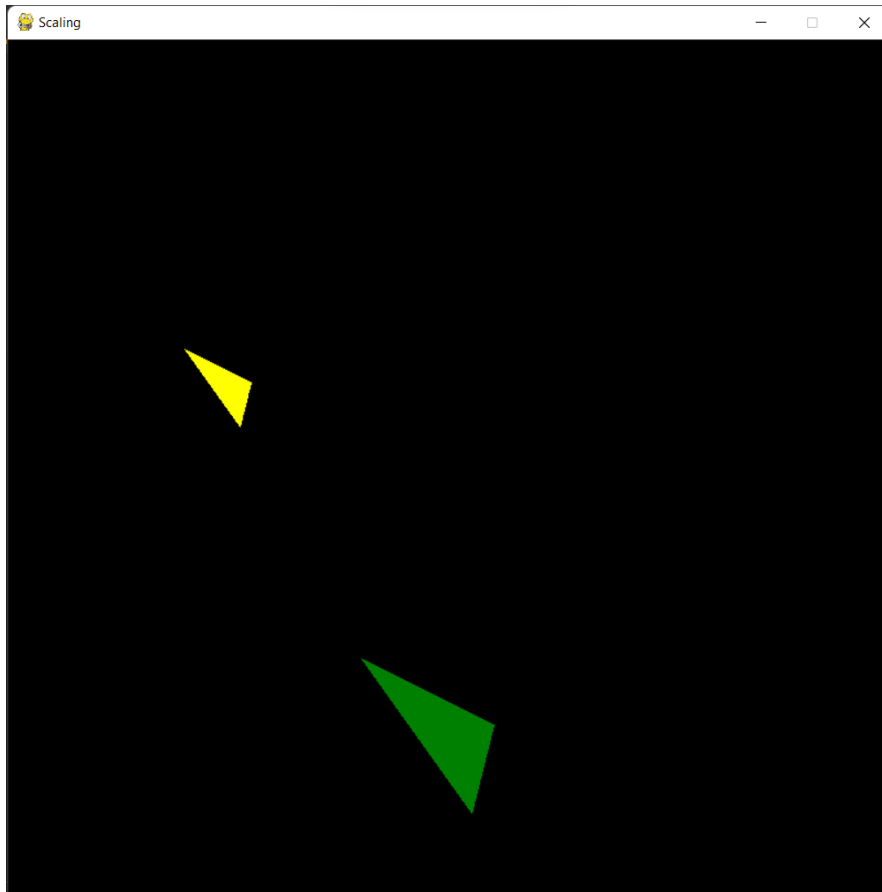
So, co-ordinates are $(x_1', y_1', 1)$ & $(x_2', y_2', 1)$

Scaling:

Implementation:

```
1  import pygame
2  from pygame import gfxdraw
3  import numpy as np
4
5  WHITE = (255, 255, 255)
6  YELLOW = (255, 255, 0)
7  GREEN = (0,128,0)
8
9  isp = False
10 x1 = y1 = x2 = y2 = 0
11 ps = (x1, y1)
12 pe = (x2, y2)
13
14 def prepare_screen():
15     """
16     Create the initial screen.
17     """
18     pygame.init()
19     screen = pygame.display.set_mode((800, 800))
20     screen.fill((0,0,0))
21     pygame.display.set_caption("Scaling")
22     return screen
23
24 def scal(x,y,sx,sy):
25     mat =([x],
26           [y],
27           [1])
28
29     transformMat = ([sx,0,0],[0,sy,0],[0,0,1])
30
31     translatedPoints = np.dot(transformMat, mat)
32
33     return translatedPoints[0],translatedPoints[1]
34
35 screen = prepare_screen()
36 a = (160,280)
37 b = (210,350)
38 c = (220,310)
39 gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
40 a = scal(a[0],a[1], 2, 2)
41 b = scal(b[0],b[1], 2, 2)
42 c = scal(c[0],c[1], 2, 2)
43
44 gfxdraw.filled_polygon(screen, [a,b,c], GREEN)
45
46 while True:
47     for event in pygame.event.get():
48         if event.type == pygame.QUIT:
49             pygame.quit()
50             quit()
51     pygame.display.update()
```

Output:



Algorithm:

2D scaling:

Algorithm:-

Step 1: Given points: starting point (x_1, y_1) and end point (x_2, y_2)

Step 2: Given scaling factor (s_x, s_y)

Step 3: 3x3 matrix form for scaling factor is:-

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4: Matrix form of starting and end points are:-

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Step 5: Matrix multiplication for scaling:-

for starting point: $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$ i.e. $x_1' = x_1 \cos \theta - y_1 \sin \theta$
 $y_1' = x_1 \sin \theta + y_1 \cos \theta$
 $1 = 1$

for end points $\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$ i.e. $x_2' = x_2 \cos \theta - y_2 \sin \theta$
 $y_2' = x_2 \sin \theta + y_2 \cos \theta$
 $1 = 1$

So, co-ordinates are $(x_1', y_1', 1)$ & $(x_2', y_2', 1)$.

Conclusion:

In this way, we used python programming language and pygame environment to reflect, shear, translate, rotate and scale a triangle in this lab segment.