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

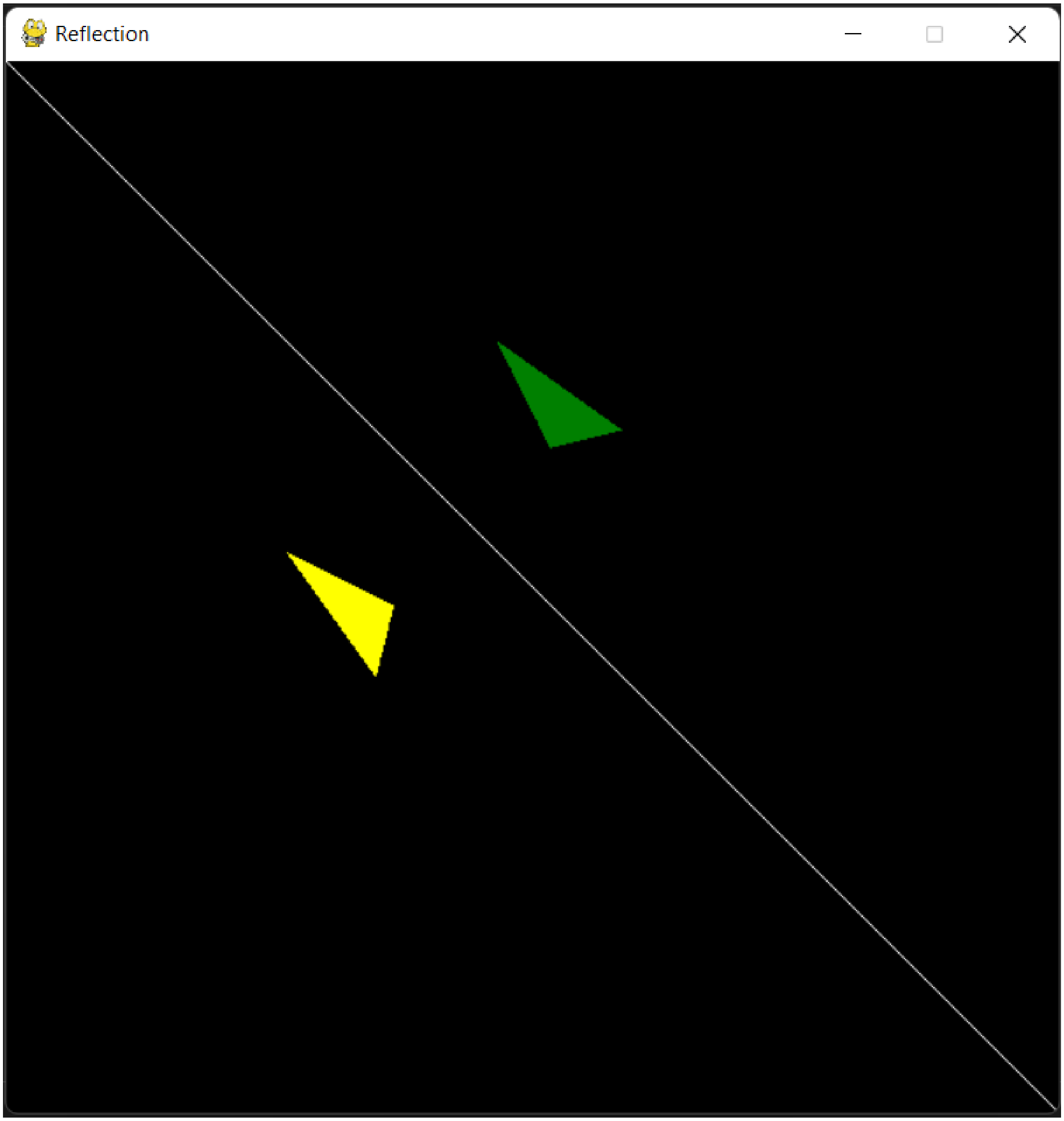
Table of Contents

Reflection:

Implementation:

```python
import pygame
from pygame import gfxdraw
import numpy as np


WHITE = (255, 255, 255)
YELLOW = (255, 255, 0)
GREEN =  (0,128,0)

isp = False
x1 = y1 = x2 = y2 = 0
ps = (x1, y1)
pe = (x2, y2)

def prepare_screen():
    """
    Create the initial screen.
    """
    pygame.init()
    screen = pygame.display.set_mode((600, 600))
    screen.fill((0,0,0))
    pygame.display.set_caption("Reflection")
    return screen

def reflection(x,y):
    mat =([x],
        [y],
        [1])

    transformMat = ([0,1,0],[1,0,0],[0,0,1])

    translatedPoints = np.dot(transformMat, mat)

    return translatedPoints[0],translatedPoints[1]

screen = prepare_screen()
gfxdraw.line(screen,0,0,1000,1000, WHITE)
a = (160,280)
b = (210,350)
c = (220,310)
gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
a = reflection(a[0],a[1])
b = reflection(b[0],b[1])
c = reflection(c[0],c[1])
gfxdraw.filled_polygon(screen, [a,b,c], GREEN)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    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

$x_1' = x_1$ & $x_2' = x_2$

$y_1' = -y_1$ $\quad$ $y_2' = -y_2$

$1 = 1$ $\quad$ $1 = 1$

so,

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

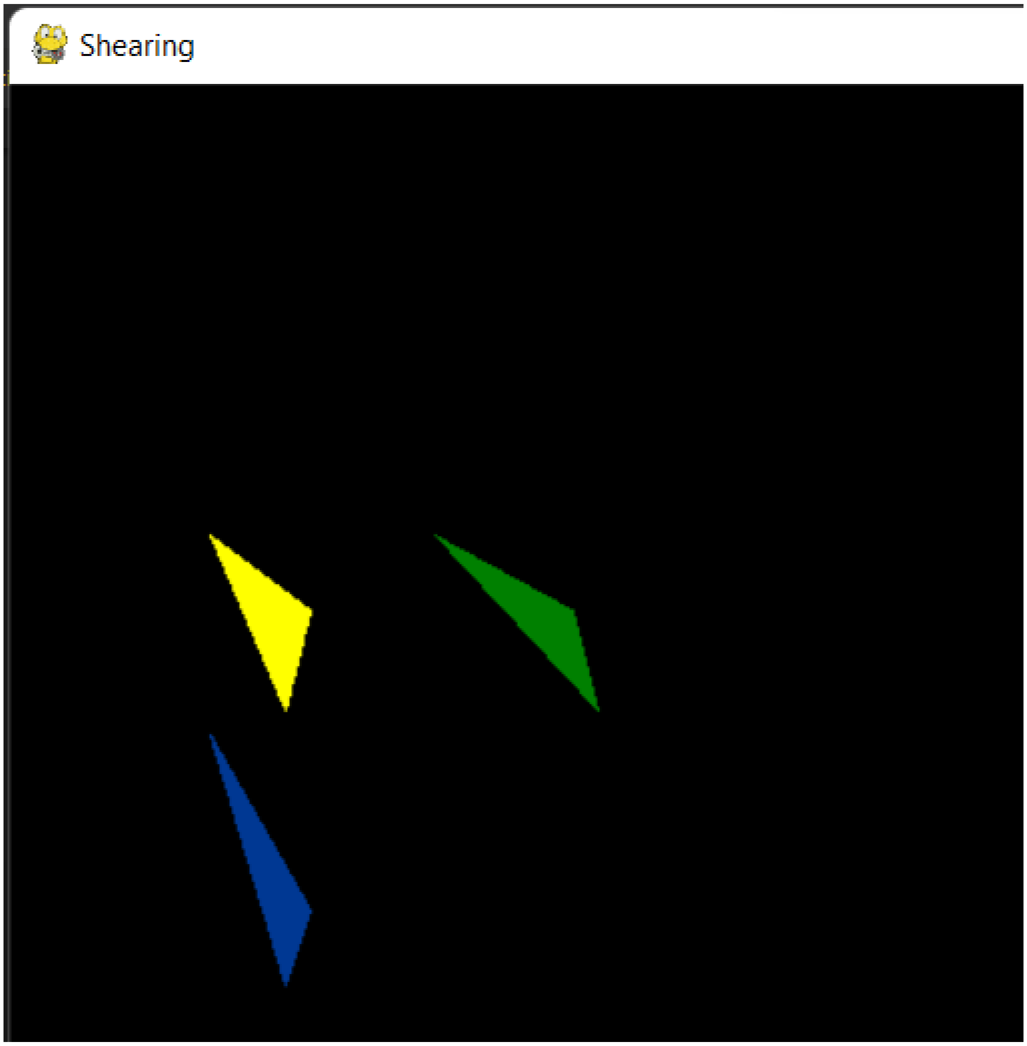
Shearing:

Implementation:

```python
import pygame
from pygame import gfxdraw
import numpy as np

(constant) YELLOW: tuple[Literal[255], Literal[255], Literal[0]]
YELLOW = (255, 255, 0)
GREEN =  (0,128,0)
BLUE = (0, 56, 147)

isp = False
x1 = y1 = x2 = y2 = 0
ps = (x1, y1)
pe = (x2, y2)

def prepare_screen():
    """
    Create the initial screen.
    """
    pygame.init()
    screen = pygame.display.set_mode((800, 800))
    screen.fill((0,0,0))
    pygame.display.set_caption("Shearing")
    return screen

def shearX(x,y,shx):
    mat =([x],
        [y],
        [1])

    transformMat = ([1,shx,0],[0,1,0],[0,0,1])

    translatedPoints = np.dot(transformMat, mat)

    return translatedPoints[0],translatedPoints[1]

def shearY(x,y,shy):
    mat =([x],
        [y],
        [1])

    transformMat = ([1,0,0],[shy,1,0],[0,0,1])

    translatedPoints = np.dot(transformMat, mat)

    return translatedPoints[0],translatedPoints[1]
```

```python
screen = prepare_screen()
a = (80,180)
b = (110,250)
c = (120,210)
gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
ax = shearX(a[0],a[1], 0.5)
bx = shearX(b[0],b[1], 0.5)
cx = shearX(c[0],c[1], 0.5)
gfxdraw.filled_polygon(screen, [ax,bx,cx], GREEN)

ay = shearY(a[0],a[1], 1)
by = shearY(b[0],b[1], 1)
cy = shearY(c[0],c[1], 1)
gfxdraw.filled_polygon(screen, [ay,by,cy], BLUE)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    pygame.display.update()
```

Output:



Algorithm:

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: Matrix 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: Matrix from 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 points $\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$.

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

$y_1' = y_1$ $y_2' = y_2$

$1 = 1$ $1 = 1$

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 $x_2' = x_2$

$y_1' = y_1 + sh_y . x_1$ $y_2' = y_2 + sh_y . y_2$

$1 = 1$ $1 = 1$

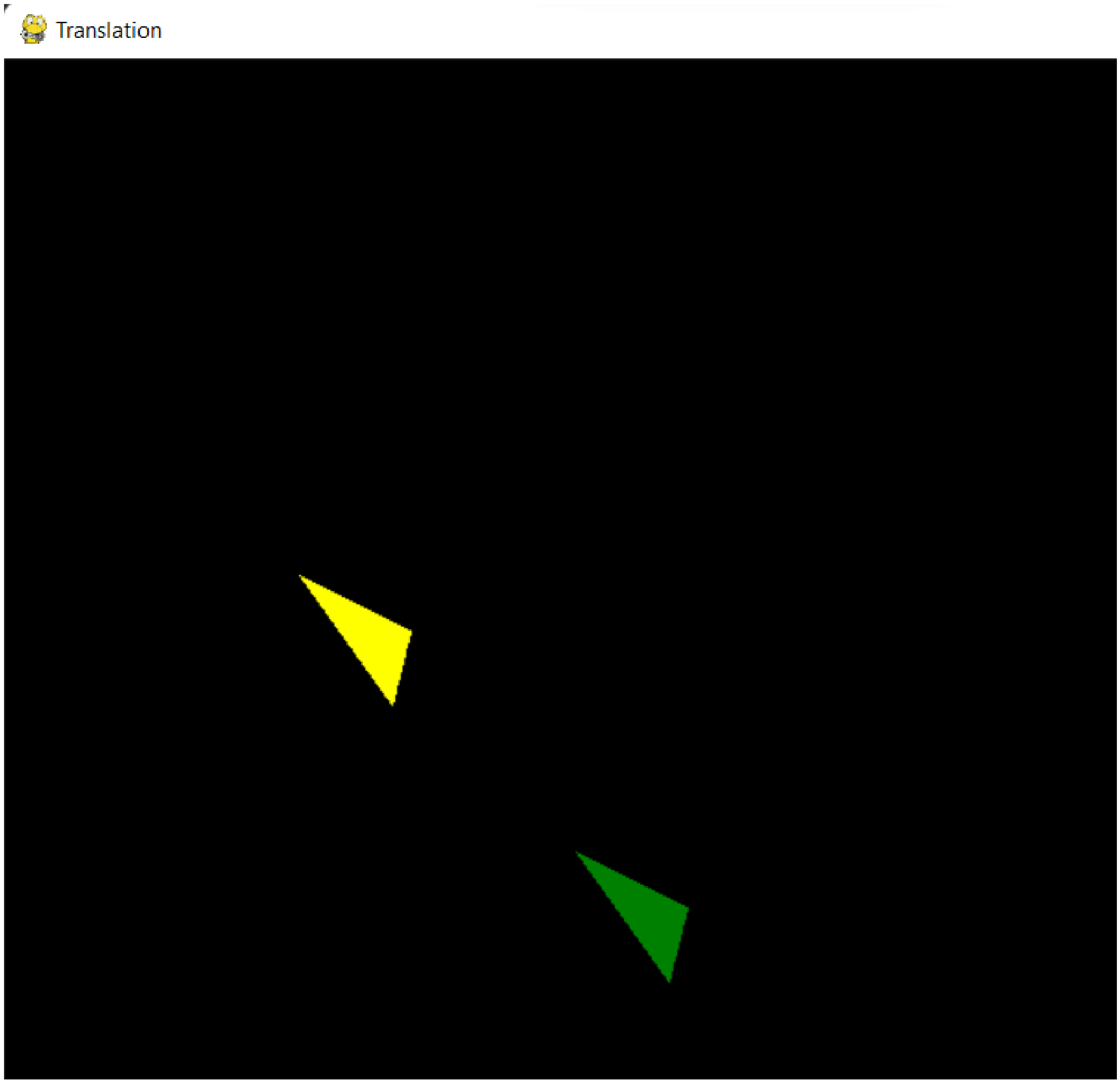
So, new co-ordinates are $(x_1', y_1', 1)$ and $(x_2', y_2', 1)$

Translation:

Implementation:

```python
import pygame
from pygame import gfxdraw
import numpy as np


WHITE = (255, 255, 255)
YELLOW = (255, 255, 0)
GREEN =  (0,128,0)

isp = False
x1 = y1 = x2 = y2 = 0
ps = (x1, y1)
pe = (x2, y2)

def prepare_screen():
    """
    Create the initial screen.
    """
    pygame.init()
    screen = pygame.display.set_mode((800, 800))
    screen.fill((0,0,0))
    pygame.display.set_caption("Translation")
    return screen

def translate(x,y, tx, ty):
    mat =([x],
        [y],
        [1])

    transformMat = ([1,0,tx],[0,1,ty],[0,0,1])

    translatedPoints = np.dot(transformMat, mat)

    return translatedPoints[0],translatedPoints[1]

screen = prepare_screen()
a = (160,280)
b = (210,350)
c = (220,310)
gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
a = translate(a[0],a[1], 150, 150)
b = translate(b[0],b[1], 150, 150)
c = translate(c[0],c[1], 150, 150)
gfxdraw.filled_polygon(screen, [a,b,c], GREEN)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    pygame.display.update()
```

Output:



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 3x3 matrix.

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

Step 4: Arranging about starting & end points into 3x1 matrix

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

Step 5: Carry out matrix multiplication:-

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

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:

```
import pygame
from pygame import gfxdraw
import numpy as np
WHITE = (255, 255, 255)
YELLOW = (255, 255, 0)
GREEN =  (0,128,0)

isp = False
x1 = y1 = x2 = y2 = 0
ps = (x1, y1)
pe = (x2, y2)
def prepare_screen():
    """
    Create the initial screen.
    """
    pygame.init()
    screen = pygame.display.set_mode((600, 600))
    screen.fill((0,0,0))
    pygame.display.set_caption("Rotation")
    return screen
screen = prepare_screen()
def rotat(x,y,a):
    a = np.math.radians(a)
    m =([x],
        [y],
        [1])
    transformM = ([np.math.cos(a),-np.math.sin(a),0],[np.math.sin(a),np.math.cos(a),0],[0,0,1])
    translatedPoints = np.dot(transformM, m)
    return translatedPoints[0],translatedPoints[1]
m = (200,600)
n = (300,200)
o = (400,250)
gfxdraw.filled_polygon(screen, [m,n,o], YELLOW)
m = rotat(m[0],m[1], 330)
n = rotat(n[0],n[1], 330)
o = rotat(o[0],o[1], 330)
gfxdraw.filled_polygon(screen, [m,n,o], GREEN)
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    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 3x3 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 $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$

ie : $x_1' = x_1\cos\theta - y_1\sin\theta$

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

$1 = 1.$

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

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

$1 = 1$

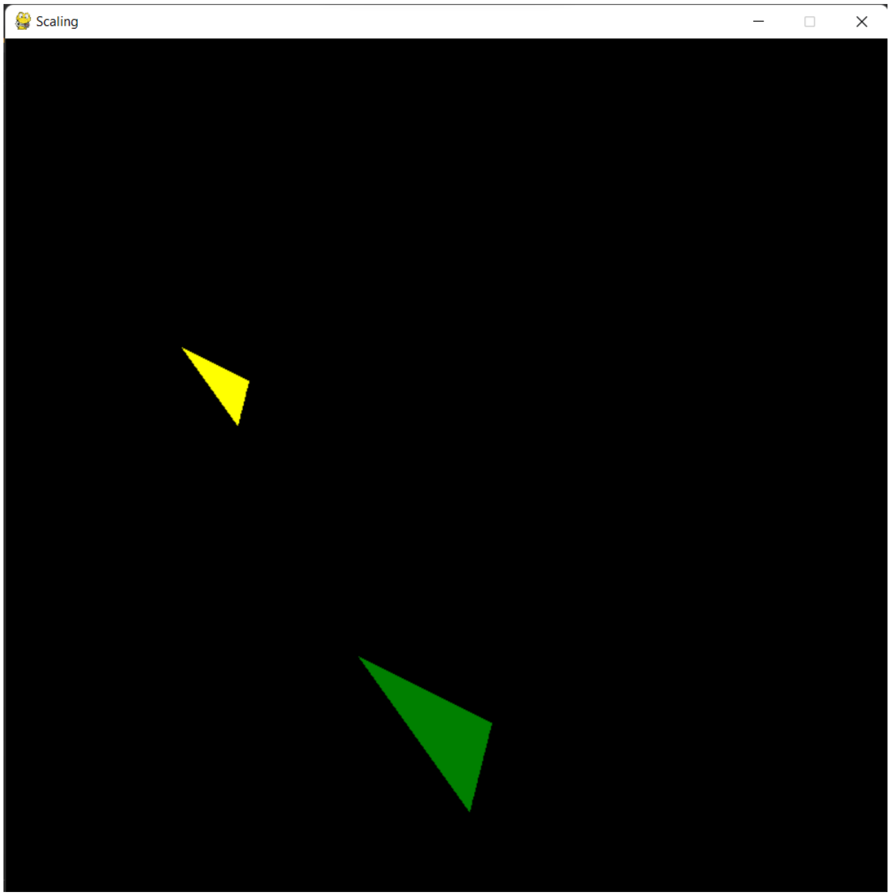
So, co-ordinates are $(x_1', y_1', 1)$ & $(x_2', y_2', 1)$.

Scaling:

Implementation:

```python
import pygame
from pygame import gfxdraw
import numpy as np

WHITE = (255, 255, 255)
YELLOW = (255, 255, 0)
GREEN =  (0,128,0)

isp = False
x1 = y1 = x2 = y2 = 0
ps = (x1, y1)
pe = (x2, y2)

def prepare_screen():
    """
    Create the initial screen.
    """
    pygame.init()
    screen = pygame.display.set_mode((800, 800))
    screen.fill((0,0,0))
    pygame.display.set_caption("Scaling")
    return screen

def scal(x,y,sx,sy):
    mat =([x],
        [y],
        [1])

    transformMat = ([sx,0,0],[0,sy,0],[0,0,1])

    translatedPoints = np.dot(transformMat, mat)

    return translatedPoints[0],translatedPoints[1]

screen = prepare_screen()
a = (160,280)
b = (210,350)
c = (220,310)
gfxdraw.filled_polygon(screen, [a,b,c], YELLOW)
a = scal(a[0],a[1], 2, 2)
b = scal(b[0],b[1], 2, 2)
c = scal(c[0],c[1], 2, 2)

gfxdraw.filled_polygon(screen, [a,b,c], GREEN)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    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 $x_2' = x_2\cos\theta - y_2\sin\theta$
$y_2' = x_2\sin\theta + y_2\sin\theta$
$1 = 1$

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.