| | | | |
|---|---|---|---|
| **Course Name:** | **Advance Python Programming and its Application** | **Semester:** | **V** |
| **Date of Performance:** | **23 / 07 / 2024** | **Batch No:** | **APPA 2** |
| **Faculty Name:** | **Prof. Deepa Jain** | **Roll No:** | **16014022096** |
| **Faculty Sign & Date:** | | **Grade/Marks:** | |

# Experiment No: 1
# Title: Study GUI features in OpenCV

**Aim and Objective of the Experiment:**

To learn following GUI features in OpenCV

1) Load an image, display it and save it back.
2) Play videos, capture videos from Camera and write it as a video.
3) Draw First letter of your Name
4) Draw stuffs with your mouse.
5) Create trackbar to control certain parameters.

**COs to be achieved:**

**CO2**: Illustrate python libraries for image processing and its applications

**Tools required:**

Any python editor tool

**Theory:**

**1) Getting Started with Images:**

**Read an image**

Use the function **cv2.imread()** to read an image. The image should be in the working directory or a full path of image should be given. Second argument is a flag which specifies the way image should be read.

• cv2.IMREAD_COLOR : Loads a color image. Any transparency of image will be neglected. It is the default flag.

• cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode

• cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel

**Display an image**

Use the function **cv2.imshow()** to display an image in a window. The window automatically fits to the image size. First argument is a window name which is a string. second argument is our image. You can create as many windows as you wish, but with different window names.

**Write an image**

Use the function **cv2.imwrite()** to save an image.

First argument is the file name, second argument is the image you want to save.

**2) Getting Started with Videos:**

**Capture Video from Camera**

Often, we have to capture live stream with camera. OpenCV provides a very simple interface to this. Let's capture a video from the camera (I am using the in-built webcam of my laptop), convert it into grayscale video and display it. Just a simple task to get started. To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So I simply pass 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture.

**Playing Video from file**

It is same as capturing from Camera, just change camera index with video file name. Also while displaying the frame, use appropriate time for cv2.waitKey(). If it is too less, video will be very fast and if it is too high, video will be slow (Well, that is how you can display videos in slow motion). 25 milliseconds will be OK in normal cases.

**Saving a Video**

So we capture a video, process it frame-by-frame and we want to save that video. For images, it is very simple, just use cv2.imwrite(). Here a little more work is required.

This time we create a VideoWriter object. We should specify the output file name (eg: output.avi). Then we should specify the FourCC code (details in next paragraph). Then number of frames per second (fps) and frame size should be passed. And last one is isColor flag. If it is True, encoder expect color frame, otherwise it works with grayscale frame.

FourCC is a 4-byte code used to specify the video codec. The list of available codes can be found in fourcc.org. It is platform dependent. Following codecs works fine for me.

• In Fedora: DIVX, XVID, MJPG, X264, WMV1, WMV2. (XVID is more preferable. MJPG results in high size

video. X264 gives very small size video)

• In Windows: DIVX (More to be tested and added)

FourCC code is passed as cv2.VideoWriter_fourcc('M','J','P','G') or cv2.VideoWriter_fourcc(*'MJPG) for MJPG

**3) Draw different geometric shapes:**

Learn the following functions:

cv2.line(), cv2.circle() , cv2.rectangle(), cv2.ellipse(), cv2.putText(),cv2.polylines.

In all the above functions, you will see some common arguments as given below:

• img: The image where you want to draw the shapes

• color: Color of the shape. for BGR, pass it as a tuple, eg: (255,0,0) for blue. For grayscale, just pass the scalar value.

• thickness: Thickness of the line or circle etc. If -1 is passed for closed figures like circles, it will fill the shape.default thickness = 1

• lineType: Type of line, whether 8-connected, anti-aliased line etc. By default, it is 8-connected. cv2.LINE_AA gives anti-aliased line which looks great for curves.

**Adding Text to Images:**

To put texts in images, you need specify following things.

• Text data that you want to write

• Position coordinates of where you want put it (i.e. bottom-left corner where data starts).

• Font type (Check cv2.putText() docs for supported fonts)

• Font Scale (specifies the size of font)

• Regular things like color, thickness, lineType etc. For better look, lineType = cv2.LINE_AA is recommended

**4) Mouse as a Paint-Brush:**

Creating mouse callback function has a specific format which is same everywhere. It differs only in what the function does. So our mouse callback function does one thing, it draws a circle where we double-click.

Now we go for much more better application. In this, we draw either rectangles or circles (depending on the mode we select) by dragging the mouse like we do in Paint application. So our mouse callback function has two parts, one to draw rectangle and other to draw the circles. This specific example will be really helpful in creating and understanding some interactive applications like object tracking, image segmentation etc.

**5) Create trackbar to control certain parameters:**

Learn the following functions: cv2.getTrackbarPos(), cv2.createTrackbar()

Here we will create a simple application which shows the color you specify. You have a window which shows the color and three trackbars to specify each of B,G,R colors. You slide the trackbar and correspondingly window color changes. By default, initial color will be set to Black.

For cv2.getTrackbarPos() function, first argument is the trackbar name, second one is the window name to which it is attached, third argument is the default value, fourth one is the maximum value

and fifth one is the callback function which is executed every time trackbar value changes. The callback function always has a default argument which is the trackbar position. In our case, function does nothing, so we simply pass.

Another important application of trackbar is to use it as a button or switch. OpenCV, by default, doesn't have button functionality. So you can use trackbar to get such functionality. In our application, we have created one switch in which application works only if switch is ON, otherwise screen is always black.

**Code and Output :**

Q 1) Write a code to read color image, display image, convert into gray scale and stored new image.

CODE:

```
import cv2

img = cv2.imread('EXPERIMENT 01//bellingam.jpg', 0)
cv2.imshow('demo', img)

k = cv2.waitKey(0)

if (k == 27):
    cv2.destroyAllwindows()
else:
    k == ord("s")
    cv2.imwrite("newdemo.jpg", img)
    cv2.destroyAllWindows()
```

OUTPUT:


demo
EXPLORER
APPA
EXPERIMENT 01
bellingam.jpg
camtest.py
drawonimage.
listofevent.py
matplotlib_de
mouseclick1.p
mouseclick2.p
mouseclick3.p
mousepaintbr
myname.py
opencvlogo.p
paint_demo.tx
readimage.py
roi_event.py
test.py
trackbar.py
tempCodeRunn
OUTLINE
TIMELINE
PROJECT COMPONE
VS CODE PETS
Go Live

Q 2) Write a code to capture a video and play any video.
CODE:

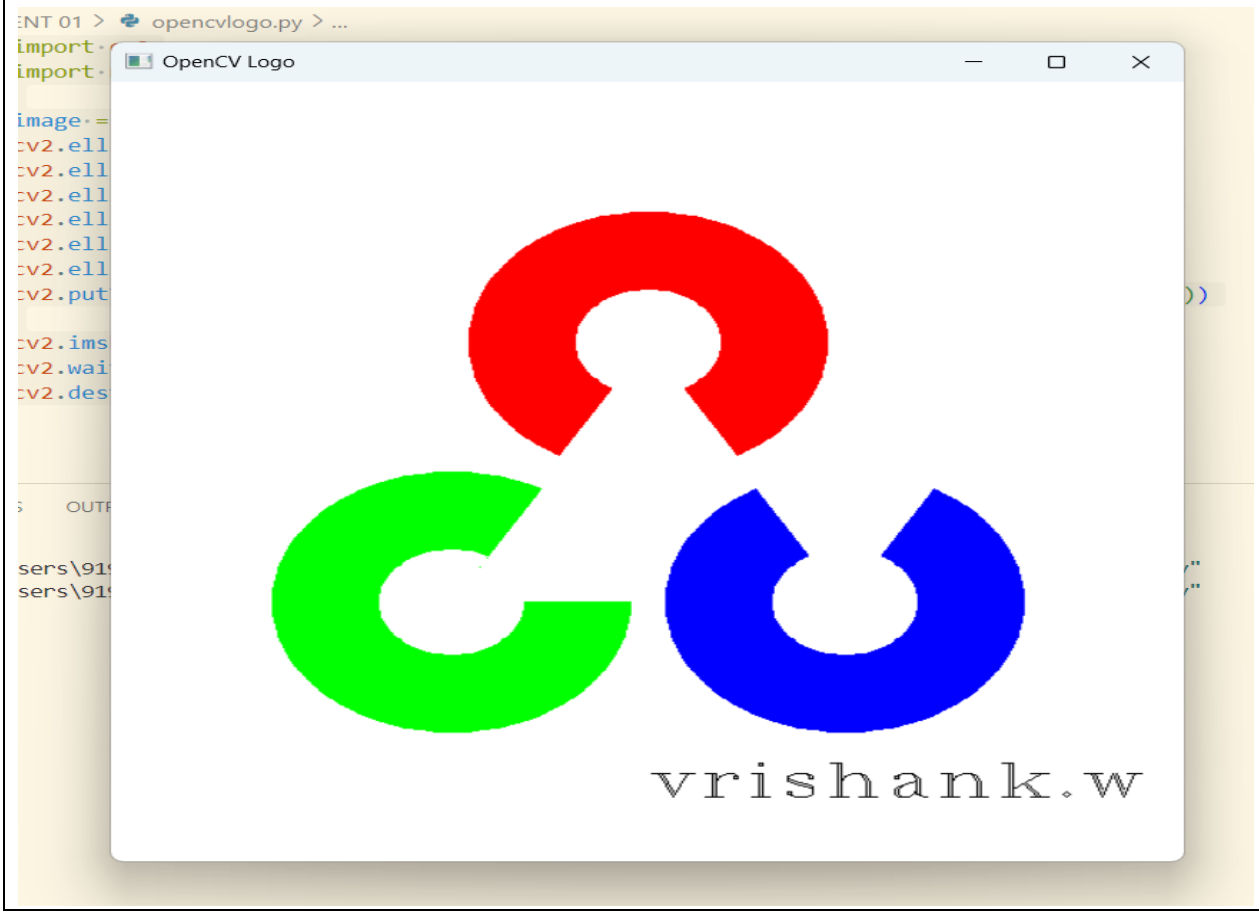
```
import cv2

cap = cv2.VideoCapture('EXPERIMENT 01//roberto.mp4')

while cap.isOpened():
    ret, frame = cap.read()

    grey = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('frame', grey)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT:

Q 3) Create the logo of OpenCV using drawing functions available in OpenCV.
CODE:

```
import cv2
import numpy

image = numpy.ones((600, 600, 3), numpy.uint8) * 255
cv2.ellipse(image, (190, 400), (100, 100), 0, 0, 300, (0, 255, 0), -1)
cv2.ellipse(image, (410, 400), (100, 100), 300, 0, 300, (255, 0, 0), -1)
cv2.ellipse(image, (300, 200), (100, 100), 120, 0, 300, (0, 0, 255), -1)
cv2.ellipse(image, (190, 400), (40, 40), 0, 0, 300, (255, 255, 255), -1)
cv2.ellipse(image, (410, 400), (40, 40), 300, 0, 300,(255, 255, 255), -1)
cv2.ellipse(image, (300, 200), (40, 40), 120, 0, 300, (255, 255, 255), -1)
cv2.putText(image, 'vrishank.w', (300,550), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2,
(0,0,0))

cv2.imshow("OpenCV Logo", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



Q 4) Create a Paint application with adjustable colors and brush radius using trackbars
CODE:

```
import cv2
import numpy as np

drawing = False
ix, iy = -1, -1

def nothing(x):
    pass

img = np.zeros((500, 700, 3), np.uint8)
cv2.namedWindow('image')

cv2.createTrackbar('R', 'image', 0, 255, nothing)
cv2.createTrackbar('G', 'image', 0, 255, nothing)
cv2.createTrackbar('B', 'image', 0, 255, nothing)
cv2.createTrackbar('Radius', 'image', 0, 10, nothing)
cv2.createTrackbar('Thickness', 'image', -1, 10, nothing)
cv2.createTrackbar('Shape', 'image', 0, 2, nothing)

def draw_circle(event, x, y, flags, param):
    global ix, iy, drawing, r, g, b, ra, crl, fb

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix, iy = x, y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            if crl == 0:
                cv2.circle(img, (x, y), ra, (b, g, r), fb)
            elif crl == 1:
                cv2.rectangle(img, (ix, iy), (x, y), (b, g, r), fb)
            else:
                cv2.line(img, (ix, iy), (x, y), (b, g, r), fb)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        if crl == 0:
            cv2.circle(img, (x, y), ra, (b, g, r), fb)
        elif crl == 1:
```

```
            cv2.rectangle(img, (ix, iy), (x, y), (b, g, r), fb)
        else:
            cv2.line(img, (ix, iy), (x, y), (b, g, r), fb)

cv2.setMouseCallback('image', draw_circle)

while True:
    cv2.imshow('image', img)
    if cv2.waitKey(1) & 0xFF == 27:
        break

    r = cv2.getTrackbarPos('R', 'image')
    g = cv2.getTrackbarPos('G', 'image')
    b = cv2.getTrackbarPos('B', 'image')
    crl = cv2.getTrackbarPos('Shape', 'image')
    ra = cv2.getTrackbarPos('Radius', 'image')
    fb = cv2.getTrackbarPos('Thickness', 'image')

cv2.destroyAllWindows()
```

OUTPUT:

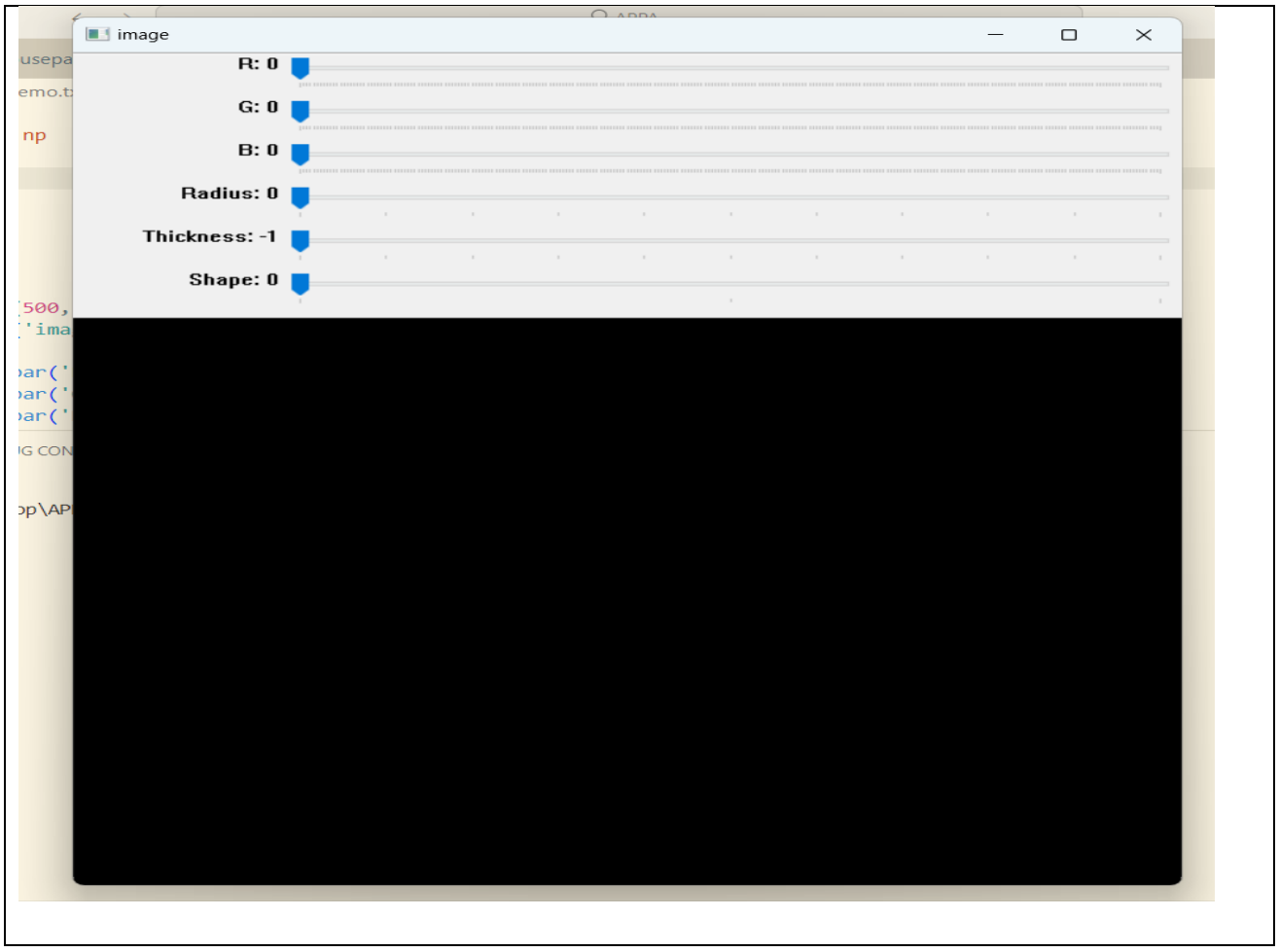


image
R: 0
G: 0
B: 0
Radius: 0
Thickness: -1
Shape: 0

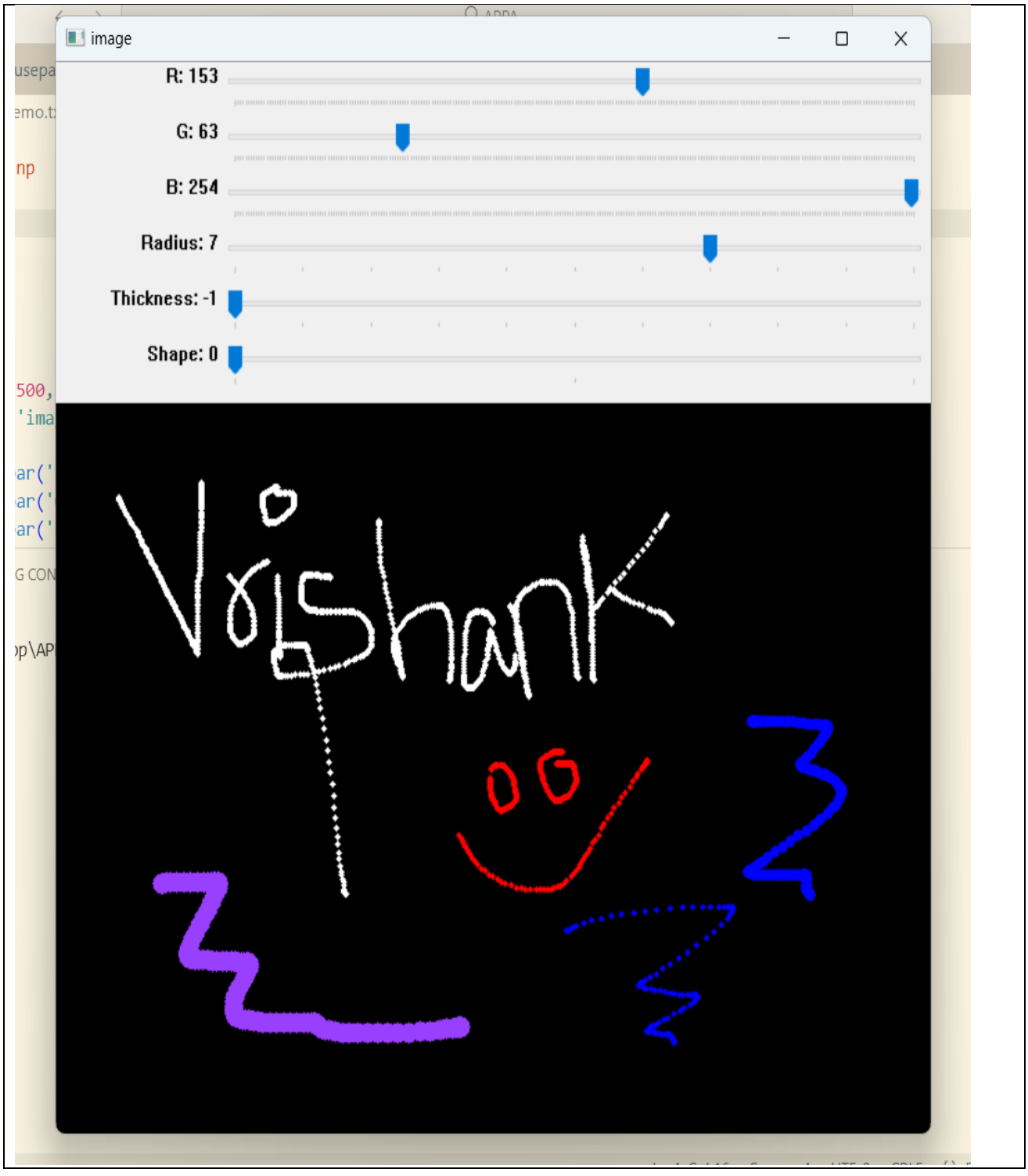


image
R: 153
G: 63
B: 254
Radius: 7
Thickness: -1
Shape: 0
Vaishnavi

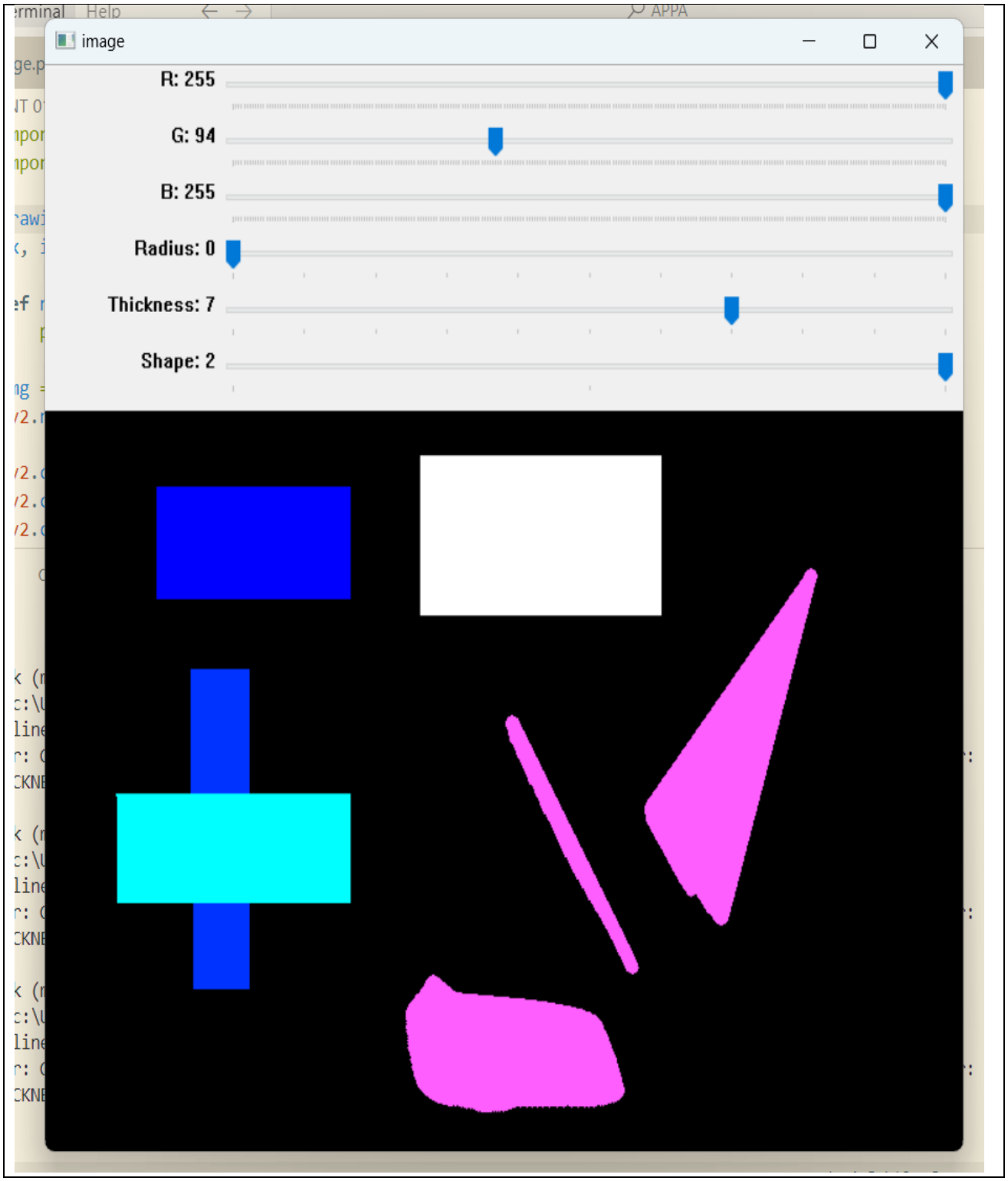


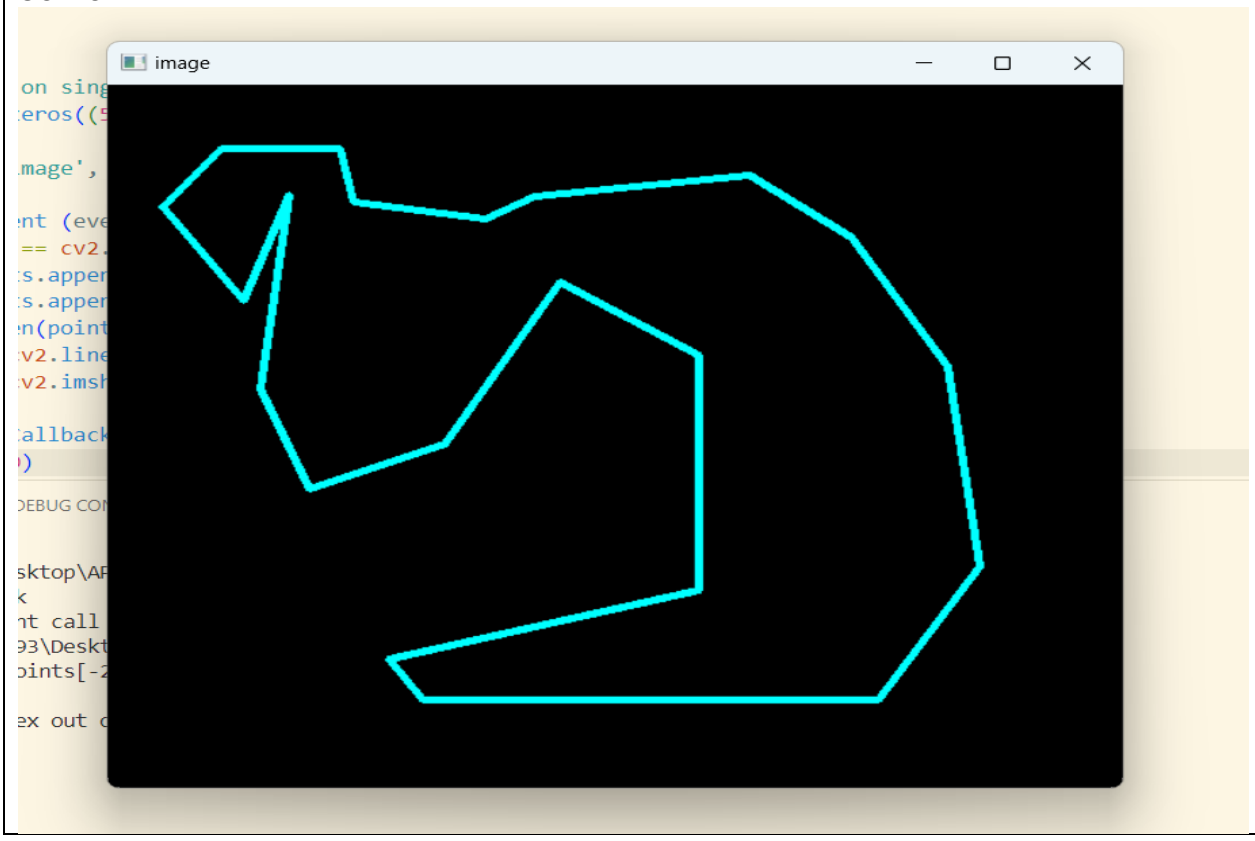
image
R: 255
G: 94
B: 255
Radius: 0
Thickness: 7
Shape: 2

## EXTRA CODES:

CODE for MULTILINES:

```
import cv2
import numpy

print("lines on single click")
img = numpy.zeros((500, 600, 3), numpy.uint8)
points = []
cv2.imshow('image', img)

def click_event (event, x, y, flag, param):
    if event == cv2.EVENT_LBUTTONDOWN :
        points.append(x)
        points.append(y)
        if len(points) >= 2:
            cv2.line(img, (points[-2], points[-1]),(points[-4], points[-3]),
(255,255,0), 4)
            cv2.imshow('image', img)

cv2.setMouseCallback('image', click_event)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:



CODE FOR PERTICULAR PIXAL IMAGE AND COORADINATE:

```
import cv2
import numpy

print("left click for coordinates and right click for rgb value")
img = cv2.imread('EXPERIMENT 01//bellingam.jpg',1)
cv2.imshow('image', img)

def click_event2(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        font = cv2.FONT_HERSHEY_SIMPLEX
        strXY = str(x)+','+str(y)
        cv2.putText(img, strXY, (x,y), font, 1, (255,255,0), 2)
        cv2.imshow('image', img)

    if event == cv2.EVENT_RBUTTONDOWN:
        blue = img[y, x, 0]
        green = img[y, x, 1]
        red = img[y, x, 2]
        font = cv2.FONT_HERSHEY_SIMPLEX
        strXY = str(red)+','+str(green)+','+str(blue)
        cv2.putText(img, strXY, (x,y), font, 1, (255,255,0), 2)
        cv2.imshow('image', img)

cv2.setMouseCallback('image', click_event2)
cv2.waitKey(0)
```

OUTPUT:

**Post Lab Subjective/Objective type Questions:**

1. Discuss the problem when you try to load color image in OpenCV and display it in Matplotlib.

ANS:

When displaying an image loaded with OpenCV in Matplotlib, the color channel order can cause issues. To fix this, you can convert the image from BGR to RGB using cv2.cvtColor(image, cv2.COLOR_BGR2RGB) before plotting it with Matplotlib. This ensures the correct color representation.

```
import cv2
import matplotlib.pyplot as plt

# Load image in BGR format
image = cv2.imread('path_to_image.jpg')

# Convert BGR to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image with Matplotlib
plt.imshow(image_rgb)
plt.title('Correct Colors')
plt.show()
```

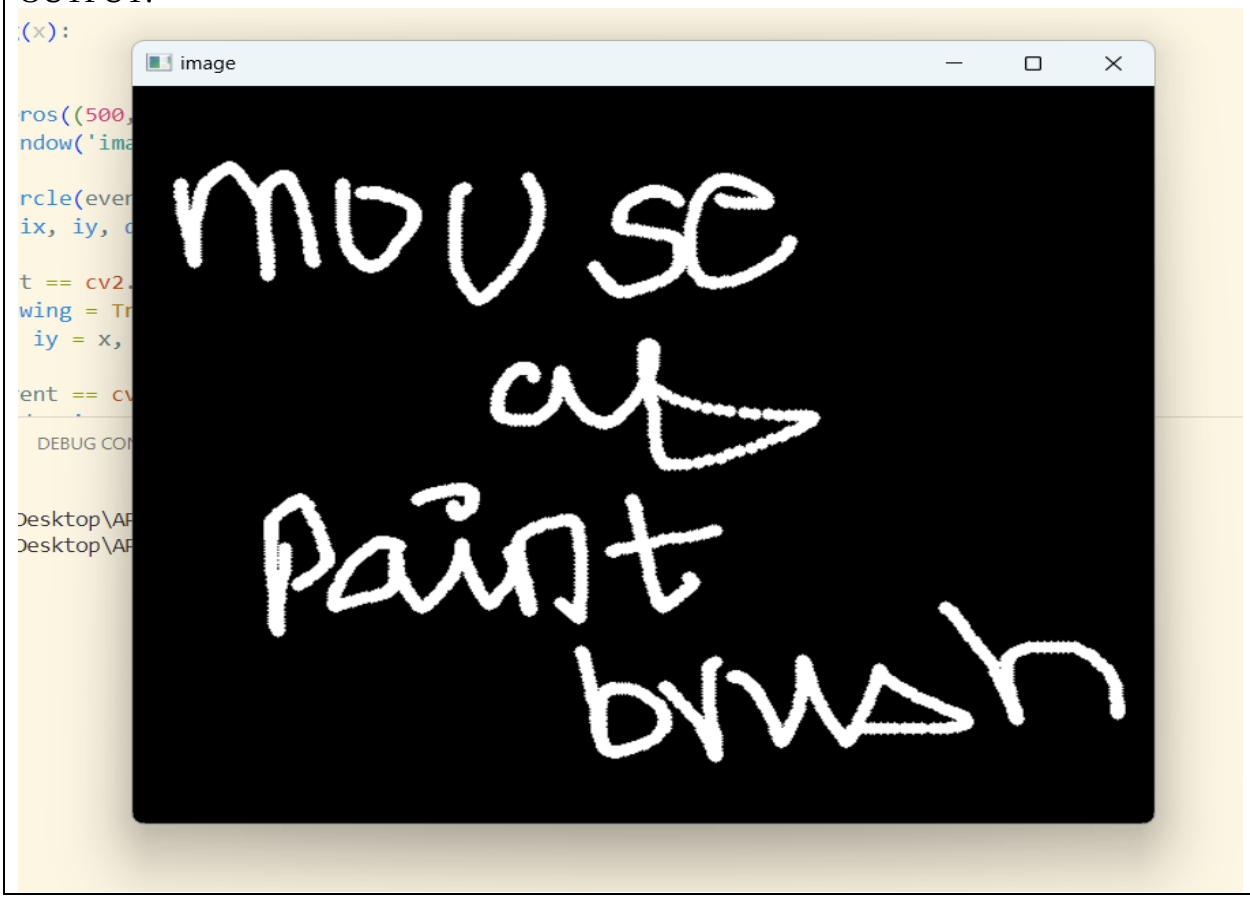
2. Write a code to make Mouse as a Paint-Brush

CODE:

```
import cv2
import numpy

drawing = False
ix, iy = -1, -1

def nothing(x):
    pass

img = numpy.zeros((500, 600, 3), numpy.uint8)
cv2.namedWindow('image')

def draw_circle(event, x, y, flags, param):
    global ix, iy, drawing, r, g, b, s

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix, iy = x, y
```

```
    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            cv2.circle(img, (x, y), 4, (255, 255, 255), -1)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        cv2.circle(img, (x, y), 4, (255, 255, 255), -1)

cv2.setMouseCallback('image', draw_circle)

while True:
    cv2.imshow('image', img)
    if cv2.waitKey(1) & 0xFF == 27:
        break


cv2.destroyAllWindows()
```

OUTPUT:



| **Conclusion:** |
| --- |
| We have successfully implemented Python, OpenCV, and NumPy libraries for image processing. We've used these libraries to read and display photos and videos, capture video from a laptop camera, create geometric shapes on images, generate black-and-white screen images, add text overlays, change pixel colors, and even use the mouse as a paintbrush. |

**Signature of faculty in-charge with Date:**