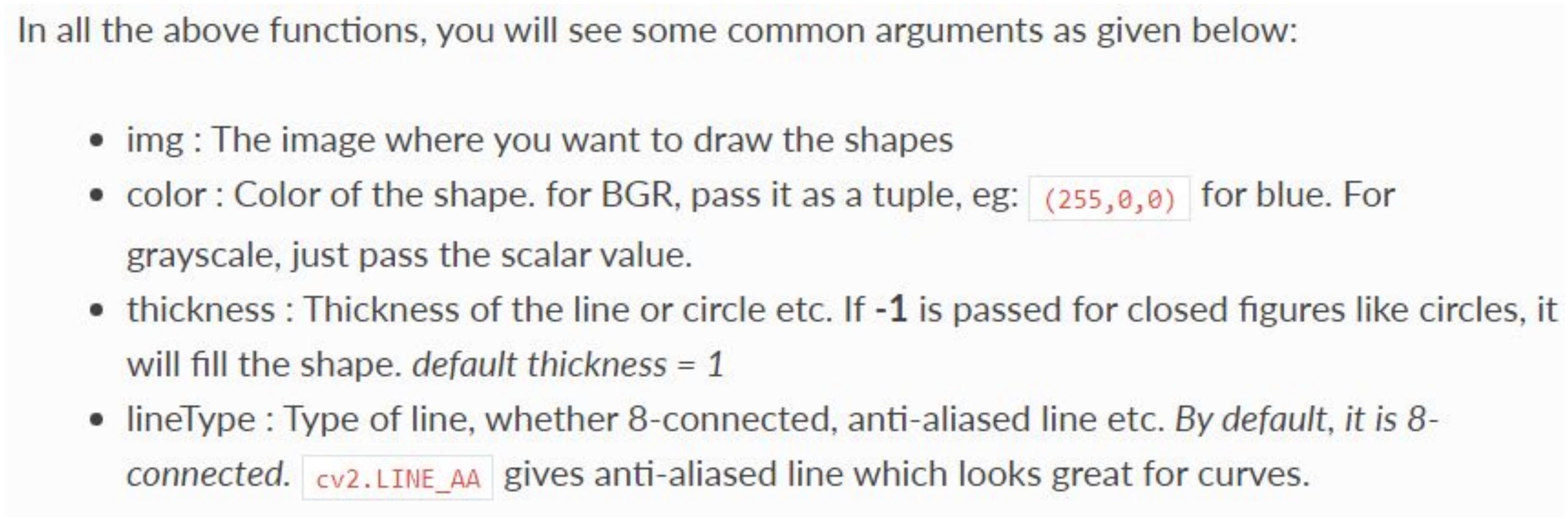
Gui Features in OpenCV

- Drawing Functions in OpenCV
- Mouse as a Paint-Brush
- Trackbar as the Color Palette

Drawing Functions in OpenCV



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.

Drawing Line

To draw a line, you need to pass starting and ending coordinates of line. We will create a black image and draw a blue line on it from top-left to bottom-right corners.

```
import numpy as np
import cv2

# Create a black image
img = np.zeros((512,512,3), np.uint8)

# Draw a diagonal blue line with thickness of 5 px
img = cv2.line(img,(0,0),(511,511),(255,0,0),5)
```

Drawing Rectangle

To draw a rectangle, you need top-left corner and bottom-right corner of rectangle. This time we will draw a green rectangle at the top-right corner of image.

```
img = cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)
```

Drawing Circle

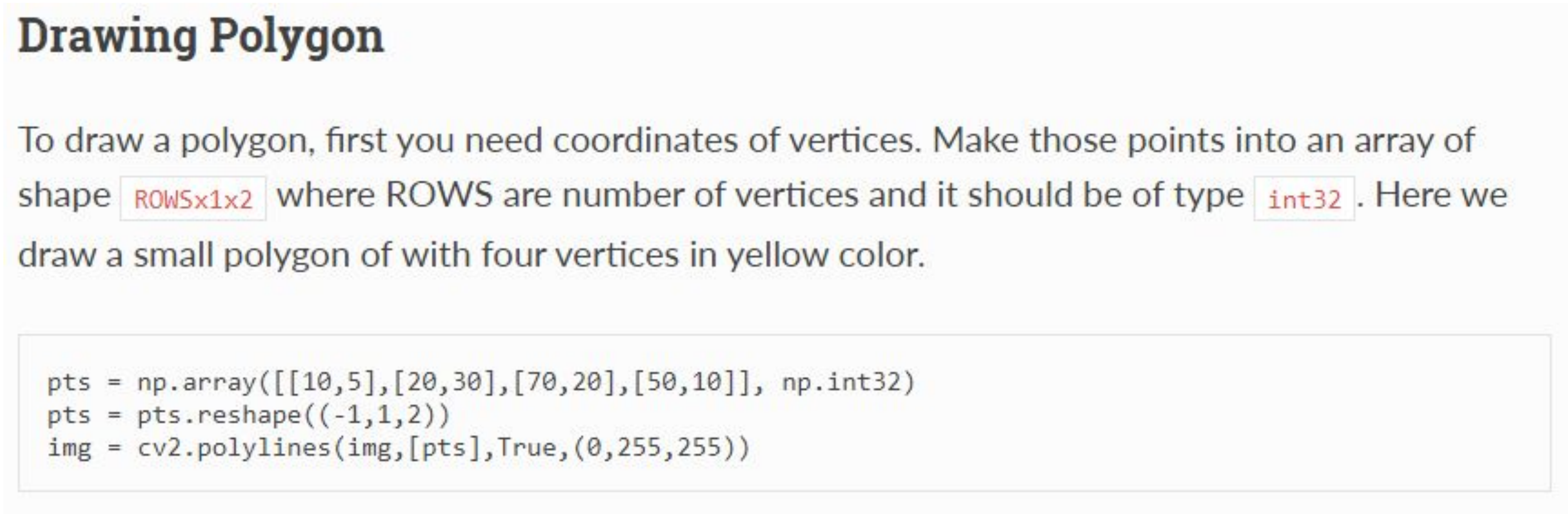
To draw a circle, you need its center coordinates and radius. We will draw a circle inside the rectangle drawn above.

```
img = cv2.circle(img,(447,63), 63, (0,0,255), -1)
```

Drawing Ellipse

To draw the ellipse, we need to pass several arguments. One argument is the center location (x,y). Next argument is axes lengths (major axis length, minor axis length). `angle` is the angle of rotation of ellipse in anti-clockwise direction. `startAngle` and `endAngle` denotes the starting and ending of ellipse arc measured in clockwise direction from major axis. i.e. giving values 0 and 360 gives the full ellipse. For more details, check the documentation of **cv2.ellipse()**. Below example draws a half ellipse at the center of the image.

```
img = cv2.ellipse(img,(256,256),(100,50),0,0,180,255,-1)
```



Drawing Polygon

To draw a polygon, first you need coordinates of vertices. Make those points into an array of shape `ROWSx1x2` where ROWS are number of vertices and it should be of type `int32`. Here we draw a small polygon of with four vertices in yellow color.

```
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
pts = pts.reshape((-1,1,2))
img = cv2.polylines(img,[pts],True,(0,255,255))
```

- If third argument is False, you will get a polylines joining all the points, not a closed shape.
- cv2.polylines() can be used to draw multiple lines. Just create a list of all the lines you want to draw and pass it to the function. All lines will be drawn individually. It is better and faster way to draw a group of lines than calling cv2.line() for each line.

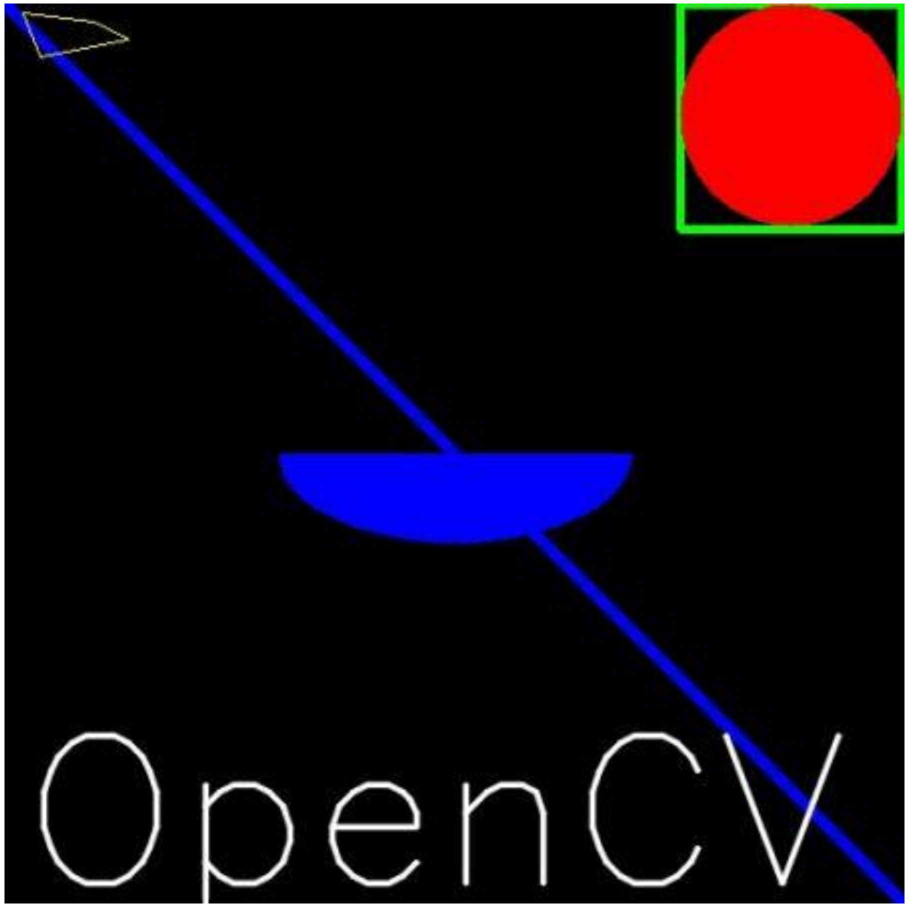
Adding Text to Images:

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

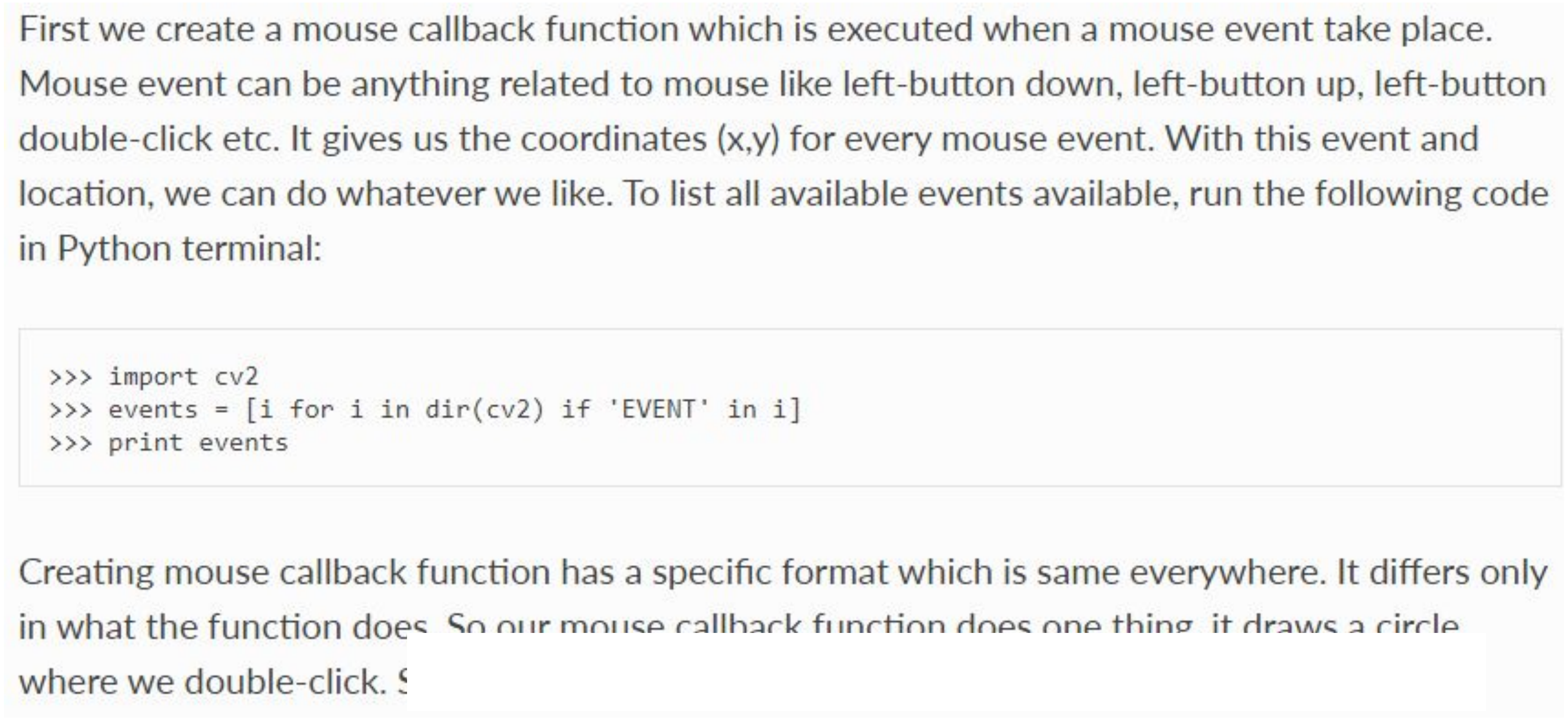
- Text data that you want to write
- Position coordinates of where you want put it (i.e. bottom-left corner where data starts).
- Font type (Check **cv2.putText()** docs for supported fonts)
- Font Scale (specifies the size of font)
- regular things like color, thickness, lineType etc. For better look, `lineType = cv2.LINE_AA` is recommended.

We will write **OpenCV** on our image in white color.

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV',(10,500), font, 4,(255,255,255),2,cv2.LINE_AA)
```


OpenCV

Mouse as a Paint-Brush



First we create a mouse callback function which is executed when a mouse event take place. Mouse event can be anything related to mouse like left-button down, left-button up, left-button double-click etc. It gives us the coordinates (x,y) for every mouse event. With this event and location, we can do whatever we like. To list all available events available, run the following code in Python terminal:

```
>>> import cv2
>>> events = [i for i in dir(cv2) if 'EVENT' in i]
>>> print events
```

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. S

```
import cv2
import numpy as np

# mouse callback function
def draw_circle(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        cv2.circle(img,(x,y),100,(255,0,0),-1)

# Create a black image, a window and bind the function to window
img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)

while(1):
    cv2.imshow('image',img)
    if cv2.waitKey(20) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```

- Next 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.

```python
import cv2
import numpy as np

drawing = False # true if mouse is pressed
mode = True # if True, draw rectangle. Press 'm' to toggle to curve
ix,iy = -1,-1

# mouse callback function
def draw_circle(event,x,y,flags,param):
    global ix,iy,drawing,mode

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

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing == True:
            if mode == True:
                cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
            else:
                cv2.circle(img,(x,y),5,(0,0,255),-1)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        if mode == True:
            cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
        else:
            cv2.circle(img,(x,y),5,(0,0,255),-1)
```

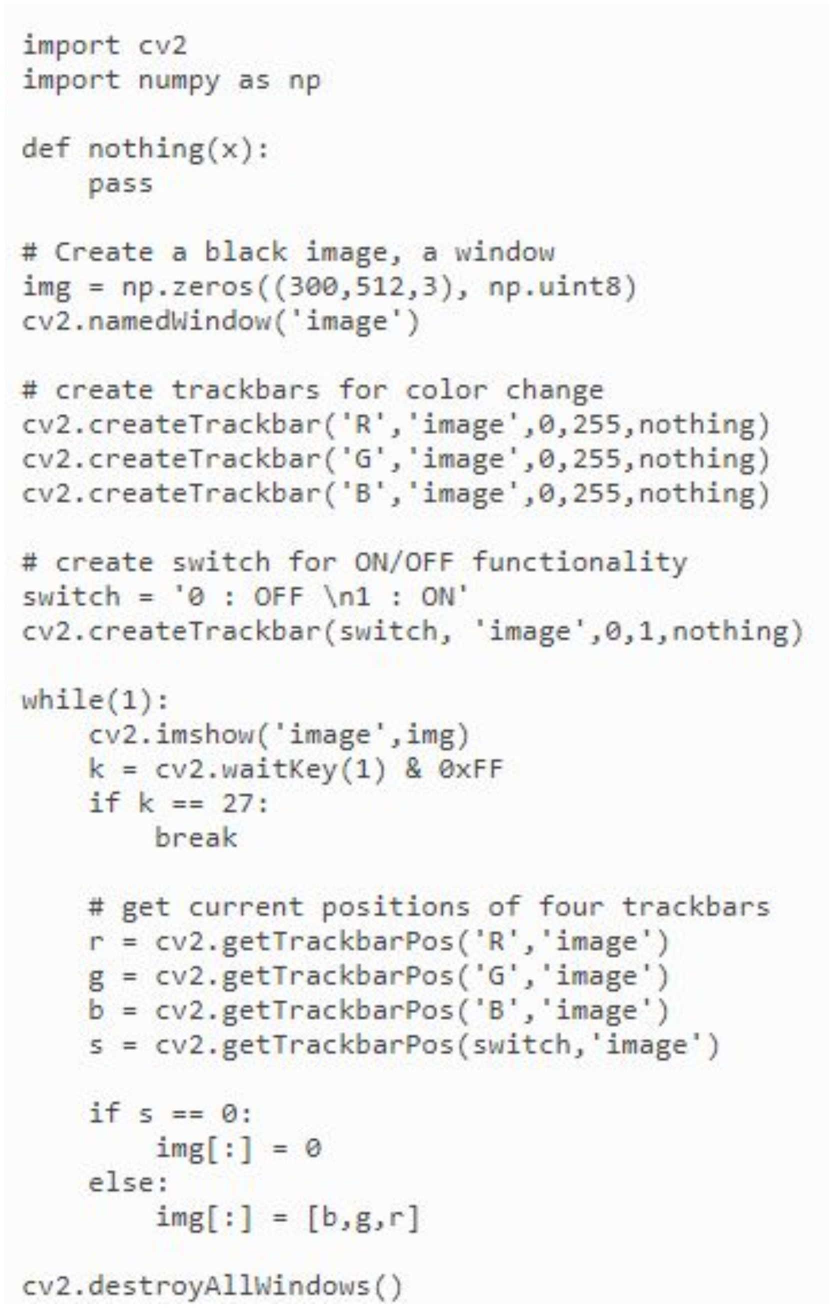
Next we have to bind this mouse callback function to OpenCV window. In the main loop, we should set a keyboard binding for key ‘m’ to toggle between rectangle and circle.

```
img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)

while(1):
    cv2.imshow('image',img)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('m'):
        mode = not mode
    elif k == 27:
        break

cv2.destroyAllWindows()
```

- 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 everytime 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.



```
import cv2
import numpy as np

def nothing(x):
    pass

# Create a black image, a window
img = np.zeros((300,512,3), np.uint8)
cv2.namedWindow('image')

# create trackbars for color change
cv2.createTrackbar('R','image',0,255,nothing)
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)

# create switch for ON/OFF functionality
switch = '0 : OFF \n1 : ON'
cv2.createTrackbar(switch, 'image',0,1,nothing)

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

    # get current positions of four trackbars
    r = cv2.getTrackbarPos('R','image')
    g = cv2.getTrackbarPos('G','image')
    b = cv2.getTrackbarPos('B','image')
    s = cv2.getTrackbarPos(switch,'image')

    if s == 0:
        img[:] = 0
    else:
        img[:] = [b,g,r]

cv2.destroyAllWindows()
```

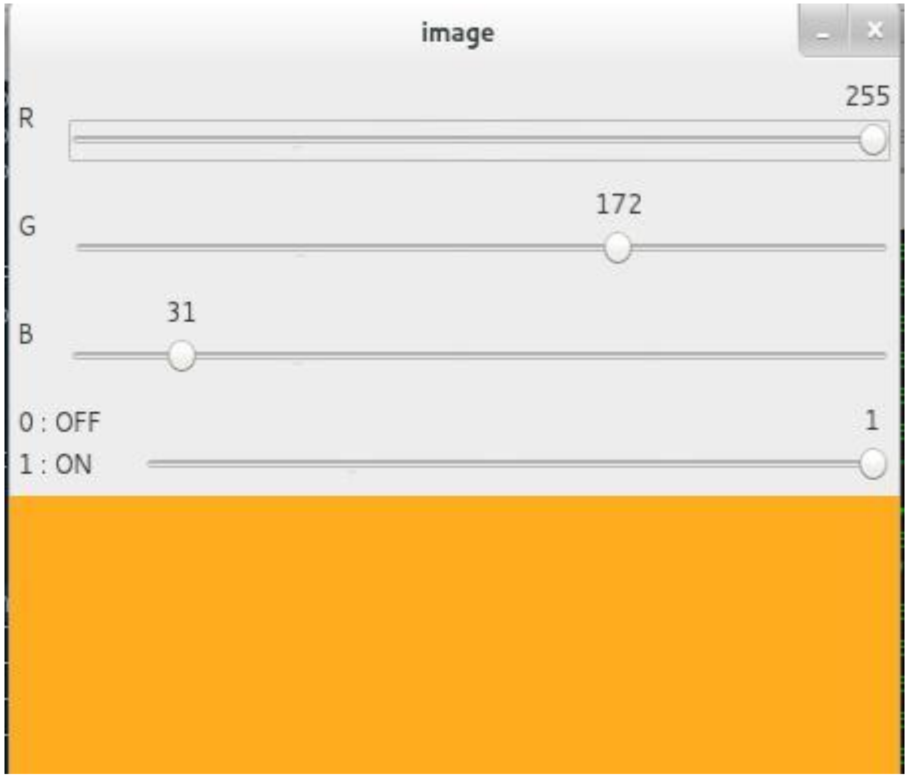


image
R
255
G
172
B
31
0 : OFF
1 : ON
1

Завдання

- Створіть логотип, використовуючи функціх малювання, що доступні в OpenCV
- Засобами OpenCV створіть «незаповнену» фігуру (коло, прямокутник)
- Створіть додаток з регульоваими кольорами та радіусом пензля (на основі mouse event)