Mouse callback function

We can access the mouse inside a window by using `cv2.setMouseCallback()`

Creating mouse callback function has a specific format. It differs only in what the function does.
The format of arguments in the callback function is as follows
`(event,x,y,flags,param)`

where

- x,y ---> are the current mouse position on the canvas
- Event ---> mouse event trigger captured by the opencv
- flags ---> Is the additional triggers passed by the opencv (eg keyboard click along with the main mouse event)
- params ---> Is the parameters that can be passed by the user while calling the function

Use the following code to get a list of all the EVENTS and FLAGS

In [3]:

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

```
['EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON', 'EVENT_F
LAG_MBUTTON', 'EVENT_FLAG_RBUTTON', 'EVENT_FLAG_SHIFTKEY', 'EVENT_LBUTTOND
BLCLK', 'EVENT_LBUTTONDOWN', 'EVENT_LBUTTONUP', 'EVENT_MBUTTONDBLCLK', 'EV
ENT_MBUTTONDOWN', 'EVENT_MBUTTONUP', 'EVENT_MOUSEHWHEEL', 'EVENT_MOUSEMOV
E', 'EVENT_MOUSEWHEEL', 'EVENT_RBUTTONDBLCLK', 'EVENT_RBUTTONDOWN', 'EVENT
_RBUTTONUP']
```

Practice Probelem : Drawing Circles on Double Click

In [1]:

```
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()
```

Practice Problem : Paint Brush Application

In [4]:

```
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)


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()
```

Create an Image cropping tool

In [3]:

```
import cv2
import numpy as np

drawing = 0 # true if mouse is pressed
crop = False
x1,y1 = -1,-1
cc = []
img_crop = []

img = cv2.imread("images\\lena.jpg",1)
# img_org = np.copy(img)
img_copy = np.copy(img)

# mouse callback function
def draw_circle(event,x,y,flags,param):
    global x1,y1,drawing,mode,cc,crop
    global img_copy

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = 1
        x1,y1 = x,y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing == 1 and drawing!=2:
            img_copy = np.copy(img)

            cv2.rectangle(img_copy,(x1,y1),(x,y),(0,255,0),2)


    elif event == cv2.EVENT_LBUTTONUP:
        if drawing == 1:
            drawing = 0
            cv2.rectangle(img_copy,(x1,y1),(x,y),(0,255,0),2)
            cc = [x1,y1,x,y]
            crop = True

cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)

while(1):
    cv2.imshow('image',img_copy)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('m'):
        mode = not mode
    if k == ord('c'):
        if crop == True:
            crop = False
            img_crop = img_copy[cc[1]:cc[3],cc[0]:cc[2],::]
            print(type(img_crop))
            cv2.imshow('preview',img_crop)
    elif k == 27:
        break

cv2.destroyAllWindows()
```

```
<class 'numpy.ndarray'>
```

In [ ]:

```
```