import numpy as np  
import cv2 as cv  
# función de devolución de llamada del mouse  
def draw\_circle(evento,x,y,banderas,param):  
 if evento == cv.EVENT\_LBUTTONDBLCLK:  
 cv.circle (img,(x,y),100,(255,0,0),-1)

# Crea una imagen en negro, una ventana y vincula la función a la ventana  
img = np.zeros((512,512,3), np.uint8)  
cv.namedWindow ( 'imagen' )  
cv.setMouseCallback ( 'imagen' , draw\_circle)  
while(1):  
 cv.imshow ( 'imagen' ,img)  
 if cv.waitKey (20) and 0xFF == 27:  
 break  
cv.destroyAllWindows ()

import numpy as np  
import cv2 as cv  
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 == cv.EVENT\_LBUTTONDOWN:  
 drawing = True  
 ix,iy = x,y  
 elif event == cv.EVENT\_MOUSEMOVE:  
 if drawing == True:  
 if mode == True:  
 cv.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)  
 else:  
 cv.circle(img,(x,y),5,(0,0,255),-1)  
 elif event == cv.EVENT\_LBUTTONUP:  
 drawing = False  
 if mode == True:  
 cv.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)  
 else:  
 cv.circle(img,(x,y),5,(0,0,255),-1)

img = np.zeros((512,512,3), np.uint8)  
cv.namedWindow('image')  
cv.setMouseCallback('image',draw\_circle)  
while(1):  
 cv.imshow('image',img)  
 k = cv.waitKey(1) & 0xFF  
 if k == ord('m'):  
 mode = not mode  
 elif k == 27:  
 break  
cv.destroyAllWindows()

import numpy as np  
import cv2 as cv  
  
def nothing(x):  
 pass  
  
# Función de devolución de llamada del ratón  
def draw\_circle(event, x, y, flags, param):  
 global drawing, ix, iy, b, g ,r   
 if event == cv.EVENT\_LBUTTONDOWN:  
 drawing = True  
 ix, iy = x, y  
 elif event == cv.EVENT\_LBUTTONUP:  
 drawing = False  
 cv.circle(img, (x, y), 10, (b, g, r), -1)  
  
# Inicialización de variables globales  
drawing = False  
ix, iy = -1, -1  
  
# Crear una imagen negra y una ventana  
img = np.zeros((300, 512, 3), np.uint8)  
cv.namedWindow('image')  
  
# Crear trackbars para el cambio de color  
cv.createTrackbar('R', 'image', 0, 255, nothing)  
cv.createTrackbar('G', 'image', 0, 255, nothing)  
cv.createTrackbar('B', 'image', 0, 255, nothing)  
  
# Crear interruptor para la funcionalidad ON/OFF  
switch = '0 : OFF \n1 : ON'  
cv.createTrackbar(switch, 'image', 0, 1, nothing)  
  
# Asociar la función de devolución de llamada del ratón a la ventana  
cv.setMouseCallback('image', draw\_circle)  
  
while True:  
 cv.imshow('image', img)  
 k = cv.waitKey(1) & 0xFF  
 if k == 27:  
 break  
  
 # Obtener posiciones actuales de los trackbars e interruptor  
 r = cv.getTrackbarPos('R', 'image')  
 g = cv.getTrackbarPos('G', 'image')  
 b = cv.getTrackbarPos('B', 'image')  
 s = cv.getTrackbarPos(switch, 'image')  
  
 if s == 0:  
 img[:] = 0  
 else:  
 img[:] = [b, g, r]  
  
cv.destroyAllWindows()

qt.qpa.plugin: Could not find the Qt platform plugin "wayland" in "/home/dread/miniconda3/envs/tf/lib/python3.9/site-packages/cv2/qt/plugins"

Ejercicio

import numpy as np  
import cv2 as cv  
  
dibujo = False # verdadero si se presiona el mouse  
modo = True # si es Verdadero, dibuja el rectángulo. Presione 'm' para cambiar a curva  
ix, iy = -1, -1  
fx, fy = -1, -1 # finales x, y para seguir la posición del mouse  
  
# función de devolución de llamada del mouse  
def draw\_rectangle(evento, x, y, banderas, param):  
 global ix, iy, fx, fy, dibujo, modo  
 if evento == cv.EVENT\_LBUTTONDOWN:  
 dibujo = True  
 ix, iy = x, y  
 elif evento == cv.EVENT\_MOUSEMOVE:  
 if dibujo == True:  
 fx, fy = x, y  
 if modo == True:  
 img\_temp = img.copy()  
 cv.rectangle(img\_temp, (ix, iy), (fx, fy), (0, 255, 0), 1)  
 cv.imshow('imagen', img\_temp)  
 elif evento == cv.EVENT\_LBUTTONUP:  
 dibujo = False  
 fx, fy = x, y  
 if modo == True:  
 cv.rectangle(img, (ix, iy), (fx, fy), (0, 255, 0), 1)  
 cv.imshow('imagen', img)  
  
img = np.zeros((512, 512, 3), np.uint8)  
cv.namedWindow('imagen')  
cv.setMouseCallback('imagen', draw\_rectangle)  
  
while(1):  
 cv.imshow('imagen', img)  
 k = cv.waitKey(1) & 0xFF  
 if k == ord('m'):  
 modo = not modo  
 elif k == 27:  
 break  
  
cv.destroyAllWindows()