import numpy as np  
import cv2 as cv  
def nothing(x):  
 pass  
# Create a black image, a window  
img = np.zeros((300,512,3), np.uint8)  
cv.namedWindow('image')  
# create trackbars for color change  
cv.createTrackbar('R','image',0,255,nothing)  
cv.createTrackbar('G','image',0,255,nothing)  
cv.createTrackbar('B','image',0,255,nothing)  
# create switch for ON/OFF functionality  
switch = '0 : OFF \n1 : ON'  
cv.createTrackbar(switch, 'image',0,1,nothing)  
while(1):  
 cv.imshow('image',img)  
 k = cv.waitKey(1) & 0xFF  
 if k == 27:  
 break  
 # get current positions of four trackbars  
 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"

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, img, b, g, r  
 if event == cv.EVENT\_LBUTTONDOWN:  
 drawing = True  
 ix, iy = x, y  
 elif event == cv.EVENT\_LBUTTONUP and drawing:  
 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()

Ejercicio

import numpy as np  
import cv2 as cv  
  
dibujo = False # Verdadero si se presiona el mouse  
color\_pincel = (0, 0, 255) # Color inicial del pincel en rojo  
radio\_pincel = 5 # Radio inicial del pincel  
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\_paint(evento, x, y, banderas, param):  
 global ix, iy, fx, fy, dibujo, color\_pincel, radio\_pincel  
 if evento == cv.EVENT\_LBUTTONDOWN:  
 dibujo = True  
 ix, iy = x, y  
 elif evento == cv.EVENT\_MOUSEMOVE:  
 if dibujo == True:  
 fx, fy = x, y  
 cv.circle(img, (fx, fy), radio\_pincel, color\_pincel, 1)  
 elif evento == cv.EVENT\_LBUTTONUP:  
 dibujo = False  
 fx, fy = x, y  
 cv.circle(img, (fx, fy), radio\_pincel, color\_pincel, 1)  
 cv.imshow('paint', img)  
  
# Funciones de devolución de llamada para las barras de seguimiento  
def cambiar\_color\_pincel(valor):  
 global color\_pincel  
 color\_pincel = (cv.getTrackbarPos('Azul', 'paint'),  
 cv.getTrackbarPos('Verde', 'paint'),  
 cv.getTrackbarPos('Rojo', 'paint'))  
  
def cambiar\_radio\_pincel(valor):  
 global radio\_pincel  
 radio\_pincel = cv.getTrackbarPos('Radio', 'paint')  
  
# Inicializar la imagen  
img = np.zeros((512, 512, 3), np.uint8)  
cv.namedWindow('paint')  
  
# Configurar funciones de devolución de llamada del mouse  
cv.setMouseCallback('paint', draw\_paint)  
  
# Crear barras de seguimiento para cambiar el color del pincel y el radio del pincel  
cv.createTrackbar('Azul', 'paint', 255, 255, cambiar\_color\_pincel)  
cv.createTrackbar('Verde', 'paint', 0, 255, cambiar\_color\_pincel)  
cv.createTrackbar('Rojo', 'paint', 0, 255, cambiar\_color\_pincel)  
cv.createTrackbar('Radio', 'paint', 5, 20, cambiar\_radio\_pincel)  
  
while True:  
 cv.imshow('paint', img)  
   
 k = cv.waitKey(1) & 0xFF  
 if k == 27:  
 break  
  
cv.destroyAllWindows()