用调色板左键画圆右键画矩形

import cv2

import numpy as np

def nothing(x):

pass

#定义一个createTrackbar的常量，用于返回回调函数的数据

drawing =False

#这个是鼠标的状态，当为False时候表明鼠标此时的状态为关闭

ix,iy=-1,-1#ix，iy，的作用是用于记录鼠标点击的起始位置

#以下创建一个回调函数，用于获取当前轨迹的位置

def draw\_circle(event,x,y,flags,param):

r=cv2.getTrackbarPos("R","image")

g=cv2.getTrackbarPos("G","image")

b=cv2.getTrackbarPos("B","image")

color=(r,g,b)

global ix,iy,drawing,mode

#将ix,iy,drawing,mode变为全局变量

if event==cv2.EVENT\_RBUTTONDOWN:#如果右键按下的时候

drawing=True

ix,iy=x,y

elif event==cv2.EVENT\_MOUSEMOVE and flags==cv2.EVENT\_FLAG\_RBUTTON:#如果右键按下，并且移动的时候

if drawing==True:

cv2.rectangle(img, (ix, iy), (x, y), color, -1)#画下矩形

# cv2.rectangle(img, (ix, iy), (x, y), (0,0,0), -1)

elif event==cv2.EVENT\_RBUTTONUP:#如果鼠标右键松开的话，mode变为关闭状态

drawing=False

if event == cv2.EVENT\_LBUTTONDOWN:#如果鼠标左键点击的话，记录点击的的位置

drawing = True

ix, iy = x, y

elif event == cv2.EVENT\_MOUSEMOVE and flags == cv2.EVENT\_FLAG\_LBUTTON:#如果鼠标点击并且移动的话

if drawing == True:

r=int(np.sqrt((x-ix)\*\*2+(y-iy)\*\*2))#此处用的是求圆半径的公式

cv2.circle(img, (x, y),r,color,-1)#画填充的圆

# cv2.circle(img, (x, y), r,(0,0,0),-1)

elif event == cv2.EVENT\_LBUTTONUP:#同理，改变鼠标的状态

drawing = False

img=np.zeros((512,512,3),np.uint8)#创建一个图片

cv2.namedWindow("image")#命名一个窗口

cv2.createTrackbar("R","image",0,255,nothing)#创建一个可以滑动的条，滑动范围是（0~255）

cv2.createTrackbar("G","image",0,255,nothing)

cv2.createTrackbar("B","image",0,255,nothing)

cv2.setMouseCallback("image",draw\_circle)#将窗口与回调函数进行绑定

while(1):

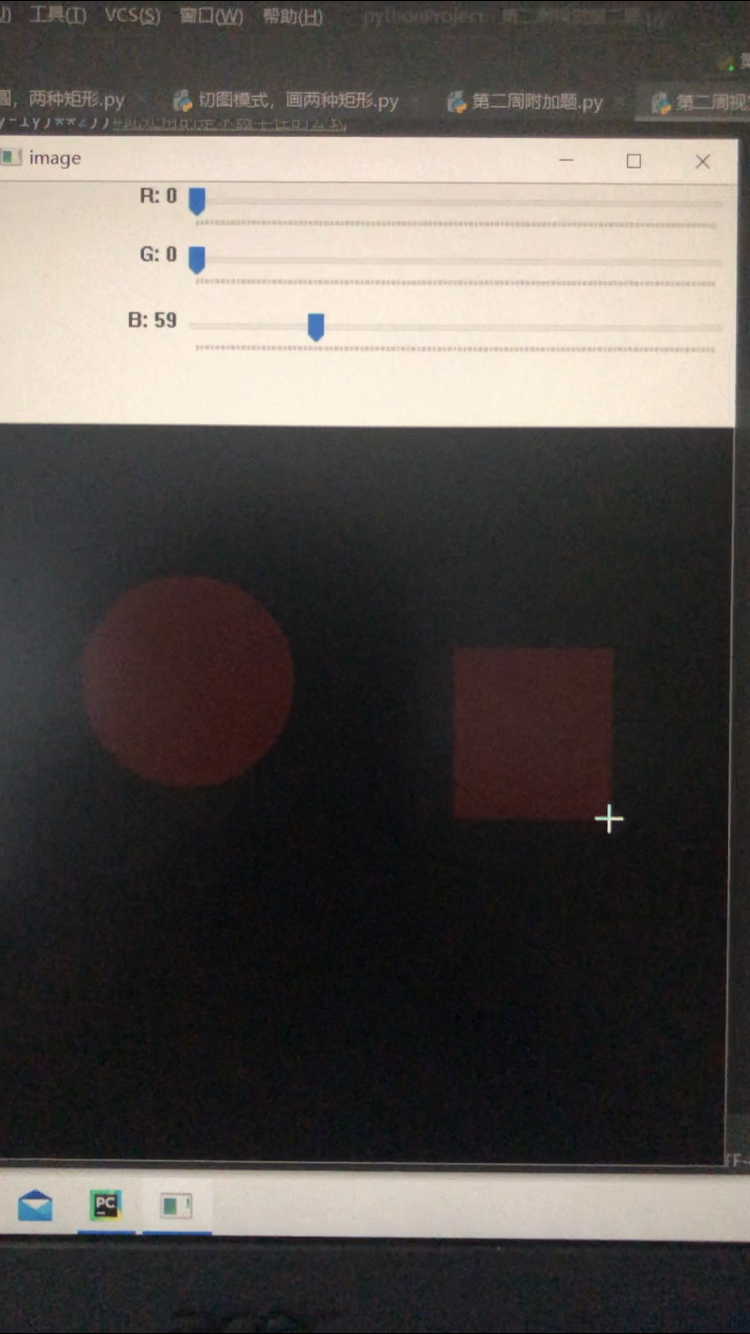
cv2.imshow("image",img)#持续显示图像

k=cv2.waitKey(1)&0xFF

if k==27:

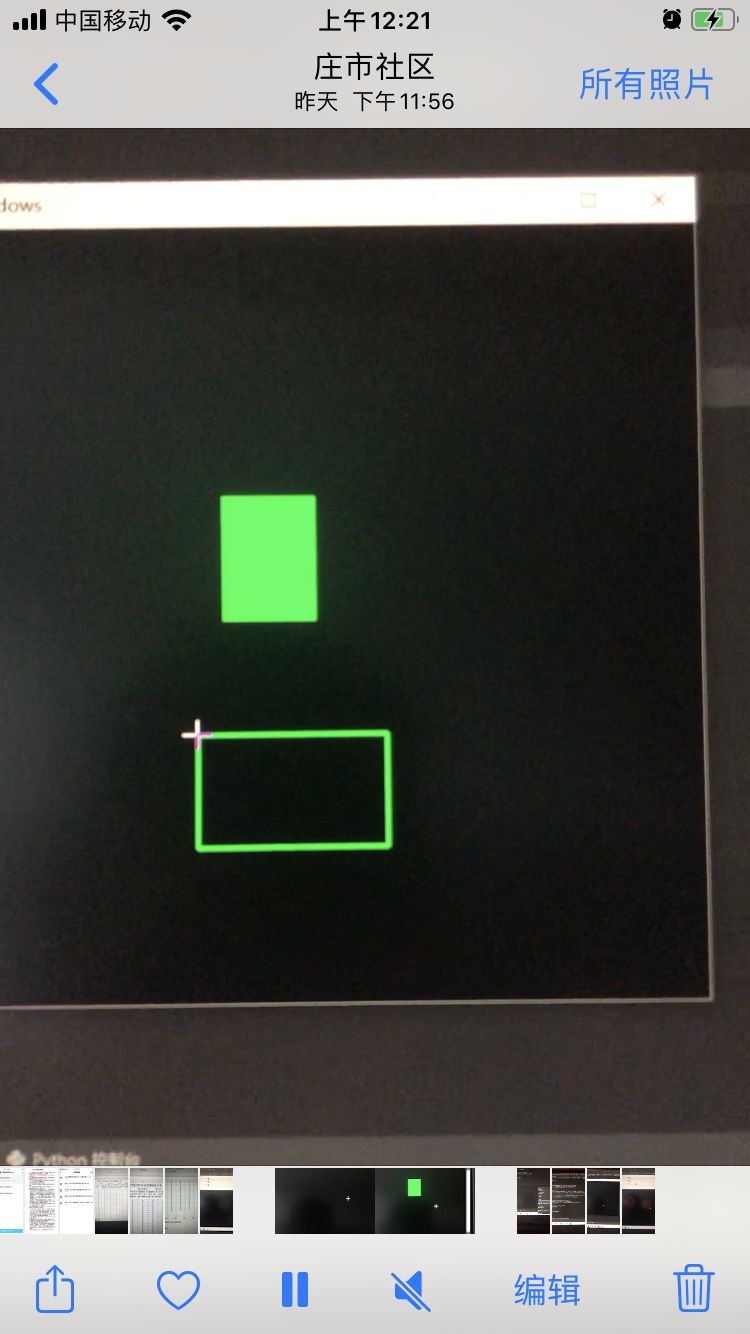
break

cv2.destroyAllWindows()

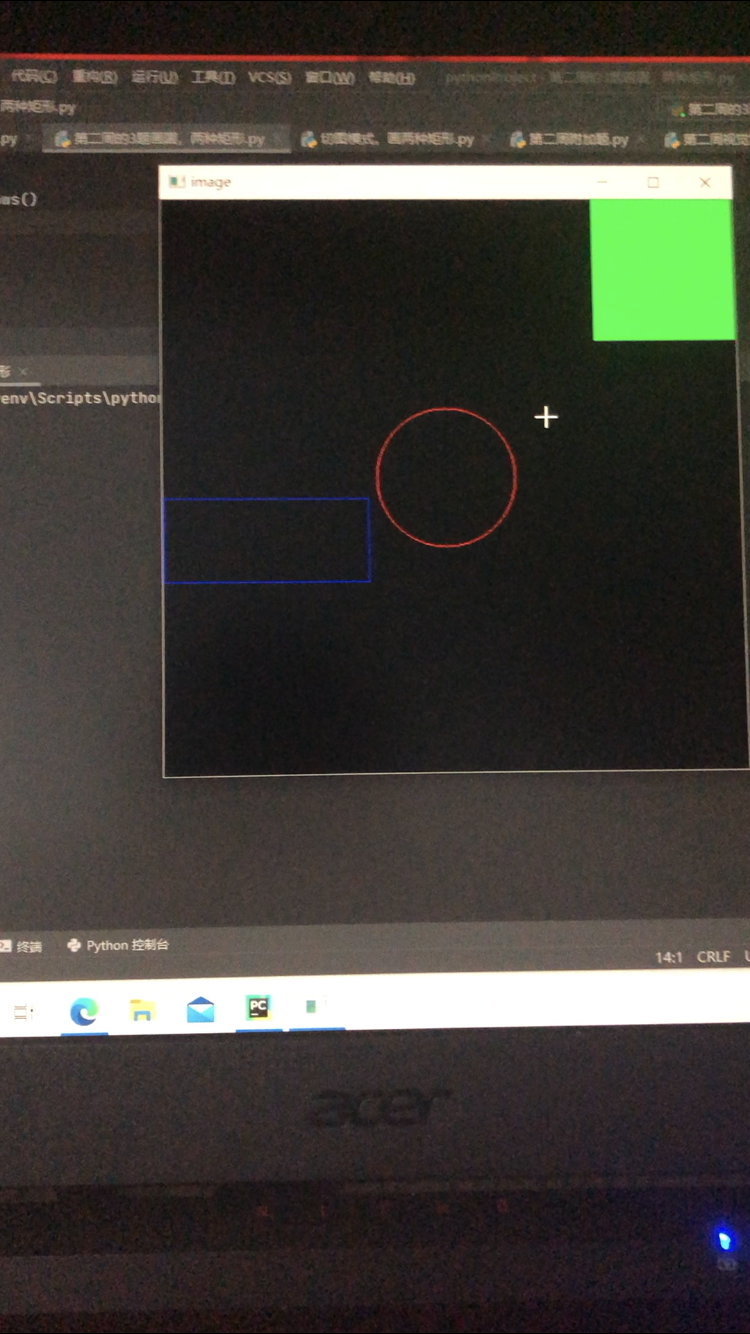


切图，按一时画填充矩形，按二的时候画不填充的矩形

import cv2import numpy as npdrawing=False#鼠标按下时为Truemode=True#模式为Ture时开始画矩形ix,iy=-1,-1def draw\_cicle(event,x,y,flags,param): global ix,iy,drawing,mode if event==cv2.EVENT\_LBUTTONDOWN:#左键按下时候 drawing=True ix,iy=x,y elif event==cv2.EVENT\_MOUSEMOVE and flags==cv2.EVENT\_FLAG\_LBUTTON:#左键移动并且左键按下时 if drawing==True: if mode==True: cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1) else:cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),2) cv2.rectangle(img, (ix, iy), (x, y), (0,0,0),-1) # r=int(np.sqrt((x-ix)\*\*2+(y-iy)\*\*2)) # cv2.circle(img,(x,y),r,(0,0,255),-1) elif event==cv2.EVENT\_LBUTTONUP:#elidf是else if 的简写,当鼠标松开的时候 drawing==False if mode==True: cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1) else:cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),2) # cv2.rectangle(img,(ix,iy),(x,y),(0,0,0),-1)img=np.zeros((512,512,3),np.uint8)cv2.namedWindow("windows")cv2.setMouseCallback("windows",draw\_cicle)#将回调函数与窗口绑定在一块while(1): cv2.imshow("windows",img) k=cv2.waitKey(1)&0xFF if k==ord('1'):#转换模式 mode=mode if k==ord('2'): mode=not mode elif k==27: breakcv2.destroyAllWindows()



用三种颜色来画圆，填充的矩形，不填填充的矩形  
  
  
 0:23:28  
import cv2  
import numpy as np  
img=np.zeros((512,512,3),np.uint8)  
  
cv2.circle(img,(250,250),62,(0,0,255),0)  
cv2.rectangle(img,(384,0),(510,128),(0,255,0),-1)  
cv2.rectangle(img,(0,268),(180,342),(255,0,),0)  
while(1):  
    cv2.imshow("image",img)  
    k=cv2.waitKey(1)&0xFF  
    if k==27:  
        break  
cv2.destroyAllWindows()



附加题：使画出的矩形为最新帧一的，鼠标没松之前就就显示图形的轮廓了

import cv2import numpy as npdrawing=Falseix,iy=-1,-1def drawing\_rectangle(event,x,y,flags,param):#设置·一个回调函数 global ix,iy,drawing#找到鼠标点击的位置，也就是说起始地点的位置， if event==cv2.EVENT\_LBUTTONDOWN:#鼠标点击之后，设置命令 drawing=True ix,iy=x,y elif event == cv2.EVENT\_MOUSEMOVE and flags == cv2.EVENT\_FLAG\_LBUTTON: cv2.rectangle(img, (ix,iy),(x,y),(0,255,0),2)#一下的两行是用背景底色来覆盖原有的圆 cv2.rectangle(img,(ix,iy),(x,y),(0,0,0),-1) elif event == cv2.EVENT\_LBUTTONUP:#当鼠标再次点过的时候，drawing变为False，说明此时的鼠标已经松开 # cv2.rectangle(img, (ix, iy), (x, y), (0, 255, 0), 2) # cv2.rectangle(img, (ix, iy), (x, y), (0, 0, 0), -1) drawing=Falseimg=np.zeros((512,521,3),np.uint8)#生成一个图片cv2.namedWindow("windows")#创建一个名字为Windows的窗口cv2.setMouseCallback("windows",drawing\_rectangle)#将回调函数与窗口绑定在一块while(1): cv2.imshow("windows",img)#持续显示 k = cv2.waitKey(1) & 0xFF if k==27: breakcv2.destroyAllWindo

