1. 创建纯色图片

如果没有合适它背景图，创建纯色图片，如灰色。

import numpy

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

1. 显示图片

import cv2

import numpy

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

def main():

cv2.waitKey(0)

cv2.destroyAllWindows()

main()

1. 调整窗体大小

import cv2

import numpy

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

def main():

cv2.waitKey(0)

cv2.destroyAllWindows()

main()

1. 进入绘制矩形状态

按下R进入绘制矩形框状态，并显示R；

按下Q退出。

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def start\_draw\_rect():

global drawing\_rect

cv2.putText(background\_img, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', background\_img)

drawing\_rect = True

main()

1. 添加鼠标事件

响应点击事件和移动事件，并打印坐标。

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

print('%s: (%d, %d)' % ('Move', x, y))

elif event == cv2.EVENT\_LBUTTONUP:

print('%s: (%d, %d)' % ('Click', x, y))

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def start\_draw\_rect():

global drawing\_rect

cv2.putText(background\_img, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', background\_img)

drawing\_rect = True

main()

1. 处理绘制矩形第一次点击

绘制矩形状态下第一次点击，进入移动绘制矩形状状态。

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

print('%s: (%d, %d)' % ('Move', x, y))

elif event == cv2.EVENT\_LBUTTONUP:

draw\_click(x, y)

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def start\_draw\_rect():

global drawing\_rect

cv2.putText(background\_img, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', background\_img)

drawing\_rect = True

def draw\_click(x, y):

if drawing\_rect:

draw\_rect\_click(x, y)

def draw\_rect\_click(x, y):

global rect\_points, drawing\_rect, drawing\_rect\_move

if len(rect\_points) == 0:

rect\_points.append((x, y))

drawing\_rect\_move = True

print('drawing moving')

main()

1. 绘制过程中的矩形

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

draw\_move(x, y)

elif event == cv2.EVENT\_LBUTTONUP:

draw\_click(x, y)

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def start\_draw\_rect():

global drawing\_rect

cv2.putText(background\_img, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', background\_img)

drawing\_rect = True

def draw\_click(x, y):

if drawing\_rect:

draw\_rect\_click(x, y)

def draw\_rect\_click(x, y):

global rect\_points, drawing\_rect, drawing\_rect\_move

if len(rect\_points) == 0:

rect\_points.append((x, y))

drawing\_rect\_move = True

def draw\_move(x, y):

if drawing\_rect:

draw\_move\_rect(x, y)

def draw\_move\_rect(x, y):

if not drawing\_rect\_move:

return

img = background\_img.copy()

cv2.rectangle(img, rect\_points[0], (x, y), [255, 0, 0], 6)

cv2.imshow('window', img)

main()

1. 固定矩形

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

draw\_move(x, y)

elif event == cv2.EVENT\_LBUTTONUP:

draw\_click(x, y)

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

cv2.imshow('window', background\_img)

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def start\_draw\_rect():

global drawing\_rect

cv2.putText(background\_img, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', background\_img)

drawing\_rect = True

def draw\_click(x, y):

if drawing\_rect:

draw\_rect\_click(x, y)

def draw\_rect\_click(x, y):

global rect\_points, drawing\_rect, drawing\_rect\_move

if len(rect\_points) == 0:

rect\_points.append((x, y))

drawing\_rect\_move = True

else:

drawing\_rect = False

drawing\_rect\_move = False

rect\_points.append((x, y))

cv2.rectangle(background\_img, rect\_points[0], rect\_points[1], [255, 0, 0], 6)

cv2.imshow('window', background\_img)

rect\_points = []

def draw\_move(x, y):

if drawing\_rect:

draw\_move\_rect(x, y)

def draw\_move\_rect(x, y):

if not drawing\_rect\_move:

return

img = background\_img.copy()

cv2.rectangle(img, rect\_points[0], (x, y), [255, 0, 0], 6)

cv2.imshow('window', img)

main()

1. 清除矩形状态显示

为了清除代表绘制矩形状态的R，需要保存绘制之前的图片。

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

draw\_move(x, y)

elif event == cv2.EVENT\_LBUTTONUP:

draw\_click(x, y)

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

background\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

img\_drawing = None

cv2.imshow('window', background\_img)

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def start\_draw\_rect():

global drawing\_rect, img\_drawing

img\_drawing = background\_img.copy()

cv2.putText(img\_drawing, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', img\_drawing)

drawing\_rect = True

def draw\_click(x, y):

if drawing\_rect:

draw\_rect\_click(x, y)

def draw\_rect\_click(x, y):

global rect\_points, drawing\_rect, drawing\_rect\_move

if len(rect\_points) == 0:

rect\_points.append((x, y))

drawing\_rect\_move = True

else:

drawing\_rect = False

drawing\_rect\_move = False

rect\_points.append((x, y))

cv2.rectangle(background\_img, rect\_points[0], rect\_points[1], [255, 0, 0], 6)

cv2.imshow('window', background\_img)

rect\_points = []

def draw\_move(x, y):

if drawing\_rect:

draw\_move\_rect(x, y)

def draw\_move\_rect(x, y):

if not drawing\_rect\_move:

return

img = img\_drawing.copy()

cv2.rectangle(img, rect\_points[0], (x, y), [255, 0, 0], 6)

cv2.imshow('window', img)

main()

过程中的绘制都在img\_drawing上进行，固定操作在background\_img上进行。

1. 撤销

为了实现撤销，需要记录操作历史，这里以list存储。整体调整命名，background\_img重命名为origin\_img代表原始图片/创建的纯色图片。使用img\_before\_draw代表每次绘制前的初始状态，并为img\_drawing提供副本。

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

KEY\_Z = ord('z')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

draw\_move(x, y)

elif event == cv2.EVENT\_LBUTTONUP:

draw\_click(x, y)

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

origin\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

img\_before\_draw = origin\_img.copy()

img\_drawing = None

cv2.imshow('window', origin\_img)

op\_seq = [origin\_img]

current\_op\_seq\_idx = 0

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Z:

undo()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def undo():

global current\_op\_seq\_idx, img\_before\_draw

if current\_op\_seq\_idx != 0:

current\_op\_seq\_idx -= 1

img\_before\_draw = op\_seq[current\_op\_seq\_idx]

cv2.imshow('window', img\_before\_draw)

img\_before\_draw = img\_before\_draw.copy()

def start\_draw\_rect():

global drawing\_rect, img\_drawing

img\_drawing = img\_before\_draw.copy()

cv2.putText(img\_drawing, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', img\_drawing)

drawing\_rect = True

def draw\_click(x, y):

if drawing\_rect:

draw\_rect\_click(x, y)

def draw\_rect\_click(x, y):

global rect\_points, drawing\_rect, drawing\_rect\_move, img\_before\_draw

global op\_seq, current\_op\_seq\_idx

if len(rect\_points) == 0:

rect\_points.append((x, y))

drawing\_rect\_move = True

else:

drawing\_rect = False

drawing\_rect\_move = False

rect\_points.append((x, y))

cv2.rectangle(img\_before\_draw, rect\_points[0], rect\_points[1], [255, 0, 0], 6)

cv2.imshow('window', img\_before\_draw)

if len(op\_seq) != current\_op\_seq\_idx + 1:

op\_seq = op\_seq[:current\_op\_seq\_idx + 1]

op\_seq.append(img\_before\_draw)

current\_op\_seq\_idx += 1

img\_before\_draw = img\_before\_draw.copy()

rect\_points = []

def draw\_move(x, y):

if drawing\_rect:

draw\_move\_rect(x, y)

def draw\_move\_rect(x, y):

if not drawing\_rect\_move:

return

img = img\_drawing.copy()

cv2.rectangle(img, rect\_points[0], (x, y), [255, 0, 0], 6)

cv2.imshow('window', img)

main()

img\_before\_draw代表每一次绘制前的图片状态，当图形固定后，使用固定好的图像更新img\_before\_draw，作为下一次图形绘制成初始状态。

current\_op\_seq\_idx记录撤销，重置到的操作，在op\_seq中的位置。撤销等操作通过更新current\_op\_seq\_idx并显示对应步骤的图片实现。

每次固定图形后，清除current\_op\_seq\_idx指向步骤之后的所有步骤，并将固定图形更新后的img\_before\_draw的副本，作为最后一步加入op\_seq并调整current\_op\_seq\_idx。

img\_before\_draw在加入op\_seq或者从op\_seq取出时，要创建副本，否则绘制会在同一个img\_before\_draw中进行，会影响op\_seq中保存的状态。

1. 处理撤销与绘制的冲突

这个问题有两种方式处理：

其一，不允许在绘制过程中进行撤销等操作；

其二，立即重置绘制状态，停止绘制，进行撤销。

本代码使用方法二实现。

import cv2

import numpy

KEY\_R = ord('r')

KEY\_Q = ord('q')

KEY\_Z = ord('z')

def mouse\_callback(event, x, y, flag, param):

if event == cv2.EVENT\_MOUSEMOVE:

draw\_move(x, y)

elif event == cv2.EVENT\_LBUTTONUP:

draw\_click(x, y)

cv2.namedWindow('window', cv2.WINDOW\_NORMAL)

cv2.resizeWindow('window', 1280, 720)

cv2.setMouseCallback('window', mouse\_callback)

origin\_img = numpy.ones((2160, 3840, 3), dtype=numpy.uint8) \* 128

img\_before\_draw = origin\_img.copy()

img\_drawing = None

cv2.imshow('window', origin\_img)

op\_seq = [origin\_img]

current\_op\_seq\_idx = 0

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def main():

while True:

pass\_key = cv2.waitKey(0)

if pass\_key == KEY\_R:

start\_draw\_rect()

elif pass\_key == KEY\_Z:

undo()

elif pass\_key == KEY\_Q:

break

cv2.destroyAllWindows()

def undo():

global current\_op\_seq\_idx, img\_before\_draw

if drawing\_rect:

reset\_draw\_rect()

if current\_op\_seq\_idx != 0:

current\_op\_seq\_idx -= 1

img\_before\_draw = op\_seq[current\_op\_seq\_idx]

cv2.imshow('window', img\_before\_draw)

img\_before\_draw = img\_before\_draw.copy()

else:

cv2.imshow('window', img\_before\_draw)

def start\_draw\_rect():

global drawing\_rect, img\_drawing

img\_drawing = img\_before\_draw.copy()

cv2.putText(img\_drawing, 'R', (50, 100), cv2.FONT\_HERSHEY\_COMPLEX, 2, [255, 0, 0], 6)

cv2.imshow('window', img\_drawing)

drawing\_rect = True

def reset\_draw\_rect():

global drawing\_rect, drawing\_rect\_move, rect\_points

drawing\_rect = False

drawing\_rect\_move = False

rect\_points = []

def draw\_click(x, y):

if drawing\_rect:

draw\_rect\_click(x, y)

def draw\_rect\_click(x, y):

global rect\_points, drawing\_rect, drawing\_rect\_move, img\_before\_draw

global op\_seq, current\_op\_seq\_idx

if len(rect\_points) == 0:

rect\_points.append((x, y))

drawing\_rect\_move = True

else:

drawing\_rect = False

drawing\_rect\_move = False

rect\_points.append((x, y))

cv2.rectangle(img\_before\_draw, rect\_points[0], rect\_points[1], [255, 0, 0], 6)

cv2.imshow('window', img\_before\_draw)

if len(op\_seq) != current\_op\_seq\_idx + 1:

op\_seq = op\_seq[:current\_op\_seq\_idx + 1]

op\_seq.append(img\_before\_draw)

current\_op\_seq\_idx += 1

img\_before\_draw = img\_before\_draw.copy()

rect\_points = []

def draw\_move(x, y):

if drawing\_rect:

draw\_move\_rect(x, y)

def draw\_move\_rect(x, y):

if not drawing\_rect\_move:

return

img = img\_drawing.copy()

cv2.rectangle(img, rect\_points[0], (x, y), [255, 0, 0], 6)

cv2.imshow('window', img)

main()

注意，在撤销首步，重做末步时，虽然不需要执行对current\_op\_seq\_idx和img\_before\_draw对调整，但是要冲绘图片，清除绘制图形中的痕迹。