رویداد کلیک ماوس:

قبل از شروع کار بیاید کمی در مورد list comprehension در پایتون صحبت کنیم:

list comprehension یا (خلاصه لیست) روشی ساده برای ایجاد و فیلتر کردن یک لیست بر اساس معیارهای خاصی در پایتون است. سینتکس list comprehension به شکل زیر است:

```
[expression for item in iterable if condition]
```

که در آن:

expression عبارتی که برای هر عنصر در لیست جدید محاسبه می‌شود .
item متغیری که در هر بار تکرار حلقه، به یک عنصر از iterableاشاره می‌کند .
iterable یک شیء قابل پیمایش مانند لیست، تاپل، رشته یا محدوده .
Condition یک عبارت شرطی اختیاری که مشخص می‌کند کدام عناصر باید در لیست جدید قرار گیرند.

بنابراین با استفاده از این نوع عملیات می توانیم به انواع event هایی که برای کار کردن با موس می توان از آنها استفاده نمود دسترسی داشته باشیم:

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

Callback :

در کتابخانهOpenCV ، رویداد "callback" (بازخوانی) به تابعی اشاره دارد که در پاسخ به یک رویداد خاص، مانند کلیک ماوس، فراخوانی می‌شود. این تابع به شما امکان می‌دهد تا اقدامات دلخواه خود را در هنگام وقوع رویداد انجام دهید.

نحوه عملکرد رویداد callback در OpenCV:

1. **تعریف تابع callback:** ابتدا، شما تابعی را تعریف می‌کنید که می‌خواهید در پاسخ به رویداد فراخوانی شود. این تابع معمولاً پارامترهایی را دریافت می‌کند که اطلاعات مربوط به رویداد را ارائه می‌دهند، مانند مختصات ماوس، نوع دکمه کلیک شده و غیره.

2. **ثبت تابع callback:** سپس، شما تابع callback خود را با استفاده از تابع `setMouseCallback` در OpenCV ثبت می‌کنید. این تابع دو پارامتر اصلی را دریافت می‌کند: نام پنجره‌ای که می‌خواهید رویدادها را در آن نظارت کنید و نام تابع callback شما.
3. **وقوع رویداد:** هنگامی که یک رویداد ماوس در پنجره نظارت شده رخ می‌دهد، OpenCV به طور خودکار تابع callback ثبت شده را فراخوانی می‌کند.
4. **اجرای تابع callback:** تابع callback شما اجرا می‌شود و می‌تواند اقدامات دلخواه شما را انجام دهد، مانند رسم شکل، تغییر رنگ پیکسل‌ها، نمایش اطلاعات و غیره.

شما همیشه می توانید با استفاده از دستور زیر یک callback تعریف کرده و سپس در یک عکس سیاه رنگ دایره رسم نمایید:

```
# mause callback func
def draw_circle(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(img,(x,y),20,(255,0,0), -1)

# create a black img
img = np.zeros((512,512,3), np.uint8)

# create a window
cv2.namedWindow('My Image')

# bind the function to window
cv2.setMouseCallback('My Image', draw_circle)

# lets do it
while True:
    cv2.imshow('My Image', img)
    if cv2.waitKey(1) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```

بیایید اجزای این کد را با هم بررسی کنیم:

`def draw_circle(event, x, y, flags, param)` این خط یک تابع به نام `draw_circle` تعریف می‌کند که به عنوان تابع callback برای رویدادهای ماوس استفاده خواهد شد .

- `Event` نوع رویداد ماوس (مانند کلیک دکمه).
- `x, y` مختصات ماوس.

- `Flags` اطلاعات اضافی در مورد رویداد.
- `Param` پارامتر اختیاری.

`if event == cv2.EVENT_LBUTTONDOWN` این خط بررسی می‌کند که آیا رویداد کلیک دکمه چپ ماوس است یا خیر .
`cv2.circle(img, (x, y), 20, (255, 0, 0), -1)` اگر دکمه چپ ماوس کلیک شده باشد، این خط یک دایره آبی رنگ در محل کلیک رسم می‌کند .

- `Img` تصویر مورد نظر.
- `(x, y)` مرکز دایره.
- `20` شعاع دایره.
- `(255, 0, 0)` رنگ آبی.
- `-1` دایره توپر.

`img = np.zeros((512, 512, 3), np.uint8)` این خط یک تصویر سیاه با ابعاد 512 x512 پیکسل ایجاد می‌کند .

- `np.zeros(...)` یک آرایه NumPy با مقادیر صفر ایجاد می‌کند.
- `(512, 512, 3)` ابعاد تصویر (عرض، ارتفاع، کانال‌های رنگی).
- `np.uint8` نوع داده (اعداد صحیح 8 بیتی بدون علامت).

`cv2.namedWindow('image')` این خط یک پنجره با نام "image" ایجاد می‌کند که تصویر در آن نمایش داده می‌شود.

`cv2.setMouseCallback('image', draw_circle)` این خط تابع `draw_circle`را به عنوان تابع callback برای رویدادهای ماوس در پنجره "image" ثبت می‌کند.

`while True` این یک حلقه بی‌نهایت است که تا زمانی که کاربر دکمه ESC را فشار دهد، ادامه خواهد داشت .
`cv2.imshow('image', img)` این خط تصویر `img`را در پنجره "image" نمایش می‌دهد .
`if cv2.waitKey(20) & 0xFF == 27` این خط منتظر می‌ماند تا کاربر یک کلید را فشار دهد .

- `cv2.waitKey(20)` منتظر می‌ماند تا یک کلید به مدت 20 میلی‌ثانیه فشرده شود.
- `& 0xFF` مقدار کلید فشرده شده را به یک مقدار 8 بیتی محدود می‌کند.

- `27 ==` بررسی می‌کند که آیا کلید فشرده شده ESC (کد اسکی 27) است یا خیر.

`Break` اگر کلید ESC فشرده شود، از حلقه خارج می‌شود.

`cv2.destroyAllWindows()` این خط تمام پنجره‌های باز شده توسط OpenCV را می‌بندد.

به طور خلاصه، این کد یک تصویر سیاه ایجاد می‌کند، یک پنجره نمایش برای آن باز می‌کند، و هر زمان که کاربر دکمه چپ ماوس را در پنجره کلیک کند، یک دایره آبی رنگ در محل کلیک رسم می‌کند.