CHAPTER 04
OPENCV 인터페이스 기초 / 사용자 인터페이스 및 I/O 처리

- 4.1 윈도우 제어
- 4.2 이벤트 처리 함수
- 4.3 그리기 함수

1

4.1 윈도우 제어

◆영상처리
- ❖ 2차원 행렬에 대한 연산
- ❖ 연산과정에서 행렬 원소 변경
- ❖ 전체 영상에 대한 변화 인지하기 어려움

◆윈도우 영상 표시
- ❖ 영상처리로 적용된 행렬 연산의 의미 이해하기 쉬움

◆OpenCV에서는 윈도우(window, 창)가 활성화된 상태에서만 마우스나 키보드 이벤트 감지

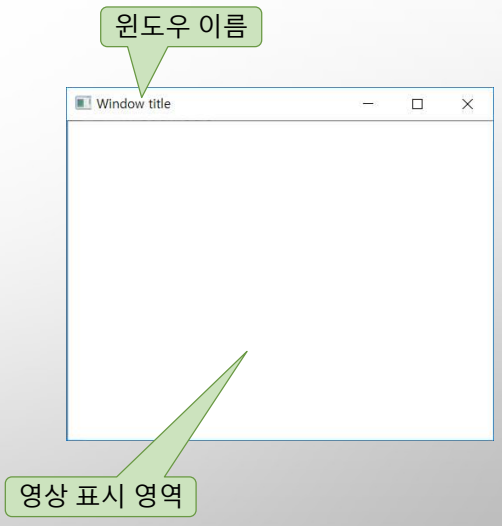
2

4.1 윈도우 제어

함수 설명	
cv2.namedWindow(winname[, flags]) → None ■설명: 윈도우 이름을 설정한 후, 해당 이름으로 윈도우 생성	
인수 설명	■ winname(str) 윈도우 이름 ■ flags(int) 윈도우의 크기 조정

옵션	값	설명
cv2.WINDOW_NORMAL	0	윈도우 크기 재조정 가능
cv2.WINDOW_AUTOSIZE	1	표시될 행렬의 크기에 맞춰 자동 조정

cv2.imshow(winname, mat) → None ■설명: winname 이름의 윈도우에 mat 행렬을 영상으로 표시함. 생성된 윈도우가 없으면, winname 이름으로 윈도우를 생성하고, 영상을 표시한다.	
인수 설명	■ mat(numpy.ndarray) 윈도우에 표시되는 영상(행렬이 화소값을 밝기로 표시)
cv2.destroyAllWindows() → None ■설명: 인수로 지정된 타이틀 윈도우 파괴	
cv2.destroyAllWindows() → None ■설명: HighGUI로 생성된 모든 윈도우 파괴	
cv2.moveWindow(winname, x, y) → None ■설명: winname 이름의 윈도우를 지정된 위치인 (x, y)로 이동. 이동되는 윈도우의 기준 위치는 좌측 상단임	
인수 설명	■ x, y 모니터 안에서 이동하려는 위치의 x, y 좌표
cv2.resizeWindow(winname, width, height) → None ■설명: 윈도우의 크기를 재조정한다.	
인수 설명	■ width, height 변경 윈도우의 가로, 세로 크기



3

4.1 윈도우 제어

라이브러리 임포트

0 원소 행렬 생성

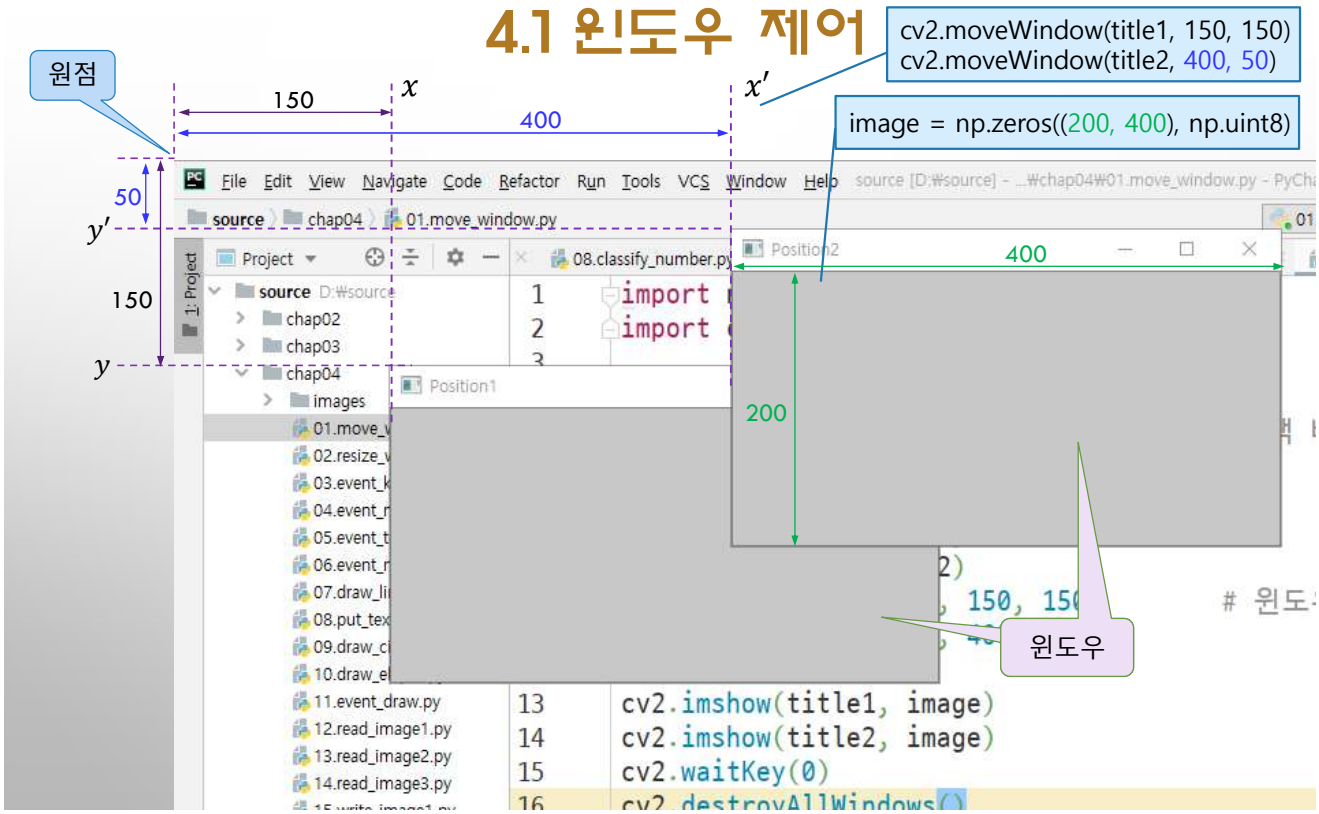
슬라이스 연산자로 행렬원소 값 지정

예제 4.1.1 윈도우 이동 – 01.move_window.py

```
01  import numpy as np                          # 넘파이 라이브러리 임포트
02  import cv2                                  # OpenCV 라리브러리 임포트
03
04  image = np.zeros((200, 400), np.uint8)      # 행렬 생성
05  image[:] = 200                              # 밝은 회색(200) 바탕 영상 생성
06
07  title1, title2 = 'Position1', 'Position2'   # 윈도우 이름
08  cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE) # 윈도우 생성 및 크기 조정 옵션
09  cv2.namedWindow(title2)
10  cv2.moveWindow(title1, 150, 150)            # 윈도우 이동 – 위치 지정
11  cv2.moveWindow(title2, 400, 50)
12
13  cv2.imshow(title1, image)                   # 행렬 원소를 영상으로 표시
14  cv2.imshow(title2, image)
15  cv2.waitKey(0)                              # 키 이벤트(key event) 대기
16  cv2.destroyAllWindows()                     # 열린 모든 윈도우 파괴
```

4

4.1 윈도우 제어



5

4.1 윈도우 제어

◆WINDOW_AUTOSIZE vs. WINDOW_NORMAL

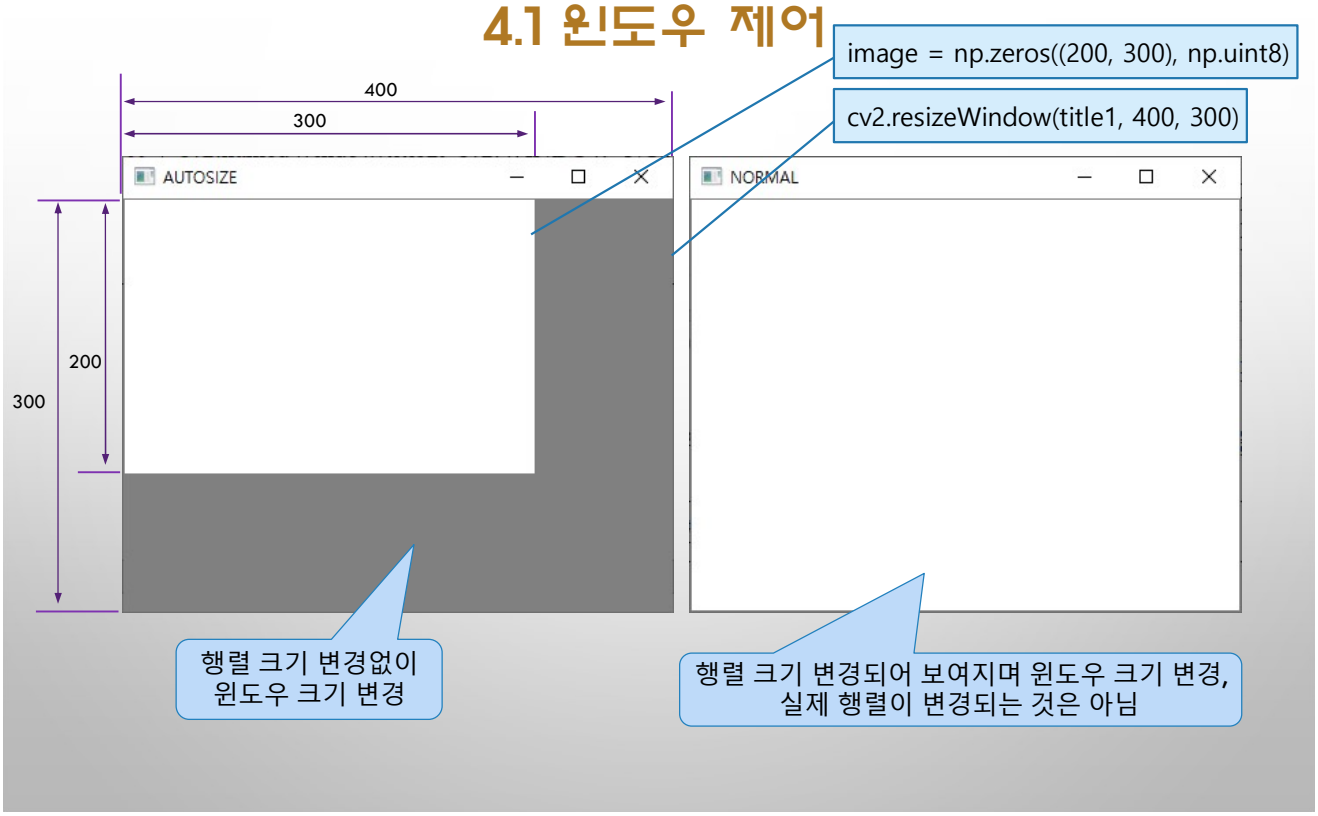
np.ndarray.fill() 함수로 원소값 지정

예제 4.1.2 윈도우의 크기 변경 – 02.window_resize.py

```
01  import numpy as np
02  import cv2
03
04  image = np.zeros((200, 300), np.uint8)          # ndarray 행렬 생성
05  image.fill(255)                                 # 모든 원소에 255(흰색) 지정
06
07  title1, title2 = 'AUTOSIZE', 'NORMAL'           # 윈도우 이름 변수
08  cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)    # 윈도우 생성 – 크기변경 불가
09  cv2.namedWindow(title2, cv2.WINDOW_NORMAL)      # 크기 변경 가능
10
11  cv2.imshow(title1, image)                       # 행렬 원소를 영상으로 표시
12  cv2.imshow(title2, image)
13  cv2.resizeWindow(title1, 400, 300)              # 윈도우 크기 변경
14  cv2.resizeWindow(title2, 400, 300)
15  cv2.waitKey(0)                                  # 키 이벤트(key event) 대기
16  cv2.destroyAllWindows()                         # 열린 모든 윈도우 제거
```

6

4.1 윈도우 제어



7

4.2 이벤트 처리 함수

◆이벤트
- ❖ 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건
- ❖ 예
  - ●사용자가 키보드의 키를 누르는 것
  - ●마우스를 움직인다거나 마우스 버튼을 누르는 것
  - ●깊이 들어가면 타이머(timer)와 같은 하드웨어 장치가 발생시키는 이벤트
  - ●사용자가 자체적으로 정의하는 이벤트

◆일반적으로 이벤트를 처리하기 위해 콜백(callback) 함수 사용

◆콜백 함수
- ❖ 개발자가 시스템 함수를 직접 호출하는 방식
- ❖ 이벤트가 발생하거나 특정 시점에 도달했을 때 시스템이 개발자가 등록한 함수 호출

◆OpenCV에서도 기본적인 이벤트 처리 함수 지원
- ❖ 키보드 이벤트, 마우스 이벤트, 트랙바(trackbar) 이벤트

8

4.2.1 키보드 이벤트 제어

함수 설명	
cv2.waitKey([, delay]) → retval ■ 설명: delay(ms: milisccond) 시간만큼 키 입력을 대기하고, 키 이벤트가 발생하면 해당 키 값 반환	
인수 설명	■ delay 지연 시간, ms 단위 - delay≤0 키 이벤트 발생까지 무한 대기 - delay>0 지연 시간 동안 키 입력 대기, 지연 시간 안에 키 이벤트 없으면 -1 반환
cv2.waitKeyEx([, delay]) → retval ■ 설명: cv2.waitKey()와 동일하지만, 전체 키 코드(full key code)를 반환한다. 화살표 키 등을 입력 받을 때 사용 가능 (OpenCV 3.4 이상에서 지원)	

◆delay 인수에 따라서 두 가지 모드 동작

9

4.2.1 키보드 이벤트 제어

예제 4.2.1 키 이벤트 사용 – 03.event_key.py

```
01  import numpy as np
02  import cv2
03
04  ## switch case문을 사전(dictionary)으로 구현
05  switch_case = {
06        ord('a'): "a키 입력",                  # ord() 함수: 문자 → 아스키코드 변환
07        ord('b'): "b키 입력",
08        0x41: "A키 입력",
09        int('0x42', 16): "B키 입력",           # 0x42(16진수) → 10진수 변환
10        2424832: "왼쪽 화살표키 입력",          # 0x250000
11        2490368: "윗쪽 화살표키 입력",          # 0x260000
12        2555904: "오른쪽 화살표키 입력",        # 0x270000
13        2621440: "아래쪽 화살표키 입력"         # 0x280000
14  }
15
16  image = np.ones((200, 300), np.float)        # 원소값 1인 행렬 생성
17  cv2.namedWindow('Keyboard Event')            # 윈도우 이름
18  cv2.imshow("Keyboard Event", image)
19
20  while True:                                  # 무한 반복
21      key = cv2.waitKeyEx(100)                 # 100ms 동안 키 이벤트 대기
22      if key == 27: break                      # ESC 키 누르면 종료
23
24      try:
25          result = switch_case[key]
26          print(result)
27      except KeyError:
28          result = -1
29
30  cv2.destroyAllWindows()                      # 열린 모든 윈도우 제거
```

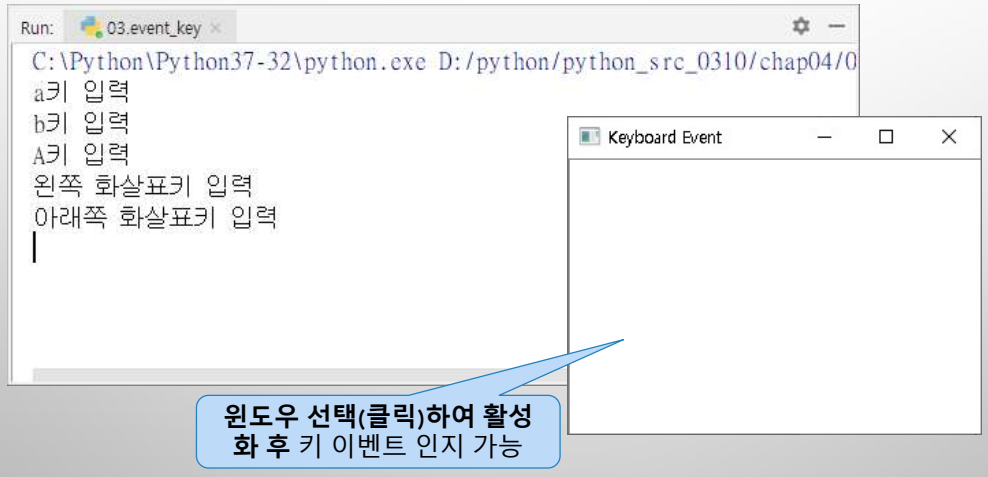
화살표키 등 특수 키 인지 위해 사용

10

4.2.1 키보드 이벤트 제어

◆결과

❖ 키 이벤트를 발생시키려면 "Keyboard Event" 윈도우를 반드시 선택(클릭)하여 활성화시킨 후, 키보드의 키를 눌려야 한다



11

4.2.2 마우스 이벤트 제어

함수 설명

def setMouseCallback(windowName, onMouse, param=None) → None

■설명: 사용자가 정의한 마우스 콜백 함수를 시스템에 등록

인수 설명		
	■ winname	이벤트 발생을 확인할 윈도우 이름, 문자열
	■ onMouse	마우스 이벤트를 처리하는 콜백 함수 이름(콜백함수)
	■ param	이벤트 처리 함수로 전달할 추가적인 사용자 정의 인수

onMouse(event, x, y, flags, param=None)

■설명: 발생한 마우스 이벤트에 대한 처리와 제어를 구현하는 콜백 함수. cv2.setMouseCallback() 함수의 두 번째 인수(onMouse)의 구현부, 따라서 이름이 같아야 함. onMouse() 함수의 인수 구조(인수 타입, 인수 순서 등)를 유지해야 함.

인수 설명

■ event — 발생한 마우스 이벤트의 종류

■ x, y — 이벤트 발생 시 마우스 포인터의 x, y 좌표

■ flags — 마우스 버튼과 동시에 특수키([Shift], [Alt], [Ctrl])를 눌렀는지 여부

옵션	값	설명
cv2.EVENT_FLAG_LBUTTON	1	왼쪽 버튼 누르기
cv2.EVENT_FLAG_RBUTTON	2	오른쪽 버튼 누르기
cv2.EVENT_FLAG_MBUTTON	4	중간 버튼 누르기
cv2.EVENT_FLAG_CTRLKEY	8	[Ctrl] 키 누르기
cv2.EVENT_FLAG_SHIFTKEY	16	[Shift] 키 누르기
cv2.EVENT_FLAG_ALTKEY	32	[Alt] 키 누르기

■ param — 콜백 함수로 전달하는 추가적인 사용자 정의 인수

〈표 4.2.1〉 마우스 이벤트 종류

옵션	값	설명
cv2.EVENT_MOUSEMOVE	0	마우스 움직임
cv2.EVENT_LBUTTONDOWN	1	왼쪽 버튼 누르기
cv2.EVENT_RBUTTONDOWN	2	오른쪽 버튼 누르기
cv2.EVENT_MBUTTONDOWN	3	중간 버튼 누르기
cv2.EVENT_LBUTTONUP	4	왼쪽 버튼 떼기
cv2.EVENT_RBUTTONUP	5	오른쪽 버튼 떼기
cv2.EVENT_MBUTTONUP	6	중간 버튼 떼기
cv2.EVENT_LBUTTONDBLCLK	7	왼쪽 버튼 더블클릭
cv2.EVENT_RBUTTONDBLCLK	8	오른쪽 버튼 더블클릭
cv2.EVENT_MBUTTONDBLCLK	9	중간 버튼 더블클릭
cv2.EVENT_MOUSEWHEEL	10	마우스 휠
cv2.EVENT_MOUSEHWHEEL	11	마우스 가로 휠

12

4.2.2 마우스 이벤트 제어

예제 4.2.2 마우스 이벤트 사용 – 04.event_mouse.py

```
01  import numpy as np
02  import cv2
03
04  def onMouse(event, x, y, flags, param):                # 콜백 함수 – 이벤트 내용 출력
05      if event == cv2.EVENT_LBUTTONDOWN:
06          print("마우스 왼쪽 버튼 누르기")
07      elif event == cv2.EVENT_RBUTTONDOWN:
08          print("마우스 오른쪽 버튼 누르기")
09      elif event == cv2.EVENT_RBUTTONUP:
10          print("마우스 오른쪽 버튼 떼기")
11      elif event == cv2.EVENT_LBUTTONDBLCLK:
12          print("마우스 왼쪽 버튼 더블클릭")
13
14  image = np.full((200, 300), 255, np.uint8)             # 초기 영상 생성
15
16  title1, title2 = "Mouse Event1", "Mouse Event2"        # 윈도우 이름
17  cv2.imshow(title1, image)                              # 윈도우 보기
18  cv2.imshow(title2, image)
19
20  cv2.setMouseCallback(title1, onMouse)                  # 마우스 콜백 함수
21  cv2.waitKey(0)                                         # 키 이벤트 대기
22  cv2.destroyAllWindows()                                # 열린 모든 윈도우 제거
```

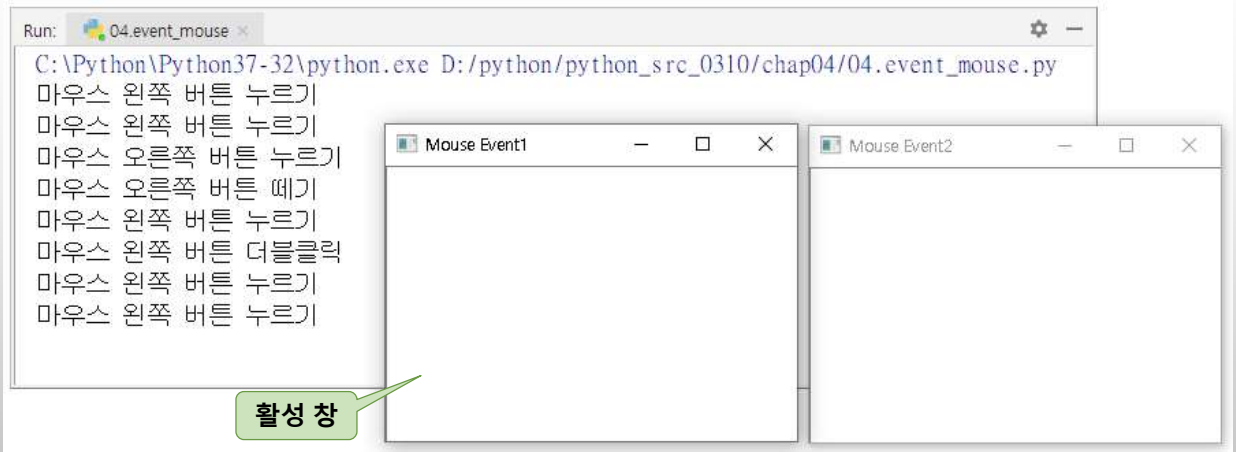
마우스 왼쪽 버튼

함수 이름

13

4.2.2 마우스 이벤트 제어

◆실행 결과



14

4.2.3 트랙바 이벤트 제어

◆트랙바(trackbar)

❖ 일정한 범위에서 특정한 값을 선택할 때 사용하는 일종의 스크롤바 혹은 슬라이더바

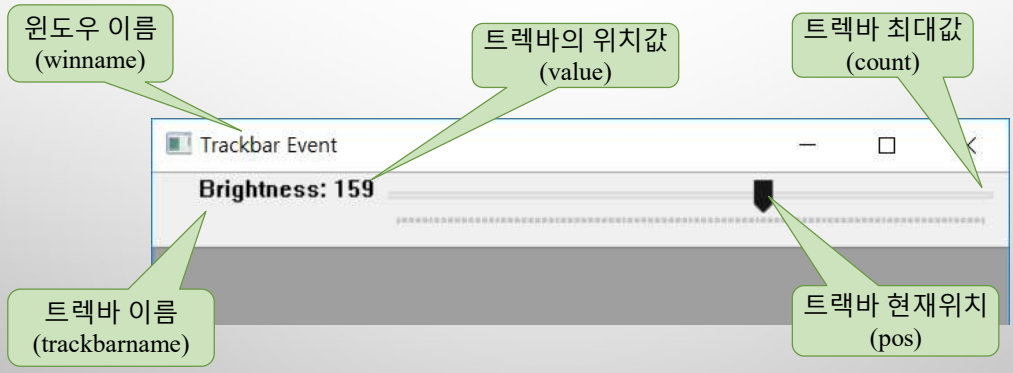
함수 설명		
cv2.createTrackbar(trackbarname, winname, value count, onChange) → None ■ 설명: 트랙바를 생성한 후, 지정한 윈도우에 추가하는 함수이다.		
인수 설명	■ trackbarname	윈도우에 생성되는 트랙바 이름
	■ winname	트랙바의 부모 윈도우 이름(트랙바 이벤트 발생을 체크하는 윈도우)
	■ value	트랙바 슬라이더의 위치를 반영하는 값(정수)
	■ count	트랙바 슬라이더의 최댓값, 최솟값은 항상 0
	■ onChange	트랙바 슬라이더의 값이 변경될 때 호출되는 콜백 함수
onChange(pos) → None ■ 설명: 트랙바 슬라이더의 위치가 변경될 때마다 호출되는 콜백 함수. cv2.createTrackbar()의 마지막 인수와 이름이 같아야 한다.		
인수 설명	■ pos	트랙바 슬라이더 위치
cv2.getTrackbarPos(trackbarname, winname) → retval ■ 설명: 지정한 트랙바의 슬라이더 위치를 반환한다.		
cv2.setTrackbarPos(trackbarname, winname, pos) → None ■ 설명: 지정한 트랙바의 슬라이더 위치를 설정한다.		

15

4.2.3 트랙바 이벤트 제어

◆형식

```
createTrackbar(trackbarname , winname, value , count , onChange , userdata );
```



16

4.2.3 트랙바 이벤트 제어

예제 4.2.3 트랙바 이벤트 사용 – 05.event_trackbar.py

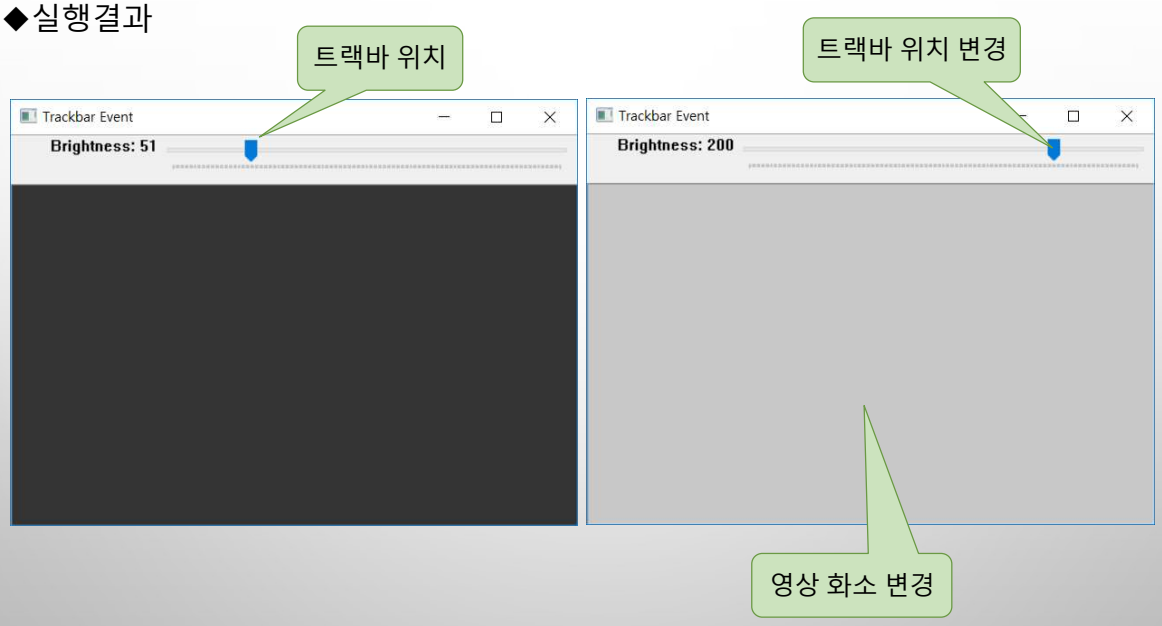
```
01  import numpy as np
02  import cv2
03
04  def onChange(value):                              # 트랙바 콜백 함수
05      global image, title                           # 전역 변수 참조
06
07      add_value = value - int(image[0][0])          # 트랙바 값과 영상 화소값 차분
08      print("추가 화소값:", add_value)
09      image = image + add_value                     # 행렬과 스칼라 덧셈 수행
10      cv2.imshow(title, image)
11
12  image = np.zeros((300, 500), np.uint8)            # 영상 생성
13
14  title = 'Trackbar Event'
15  cv2.imshow(title, image)
16
17  cv2.createTrackbar('Brightness', title, image[0][0], 255, onChange)  # 트랙바 콜백 함수 등록
18  cv2.waitKey(0)
19  cv2.destroyAllWindows()                           # 열린 모든 윈도우 제거
```

현재값 최대값 콜백함수

17

4.2.3 트랙바 이벤트 제어

◆실행결과



18

4.2.3 트랙바 이벤트 제어

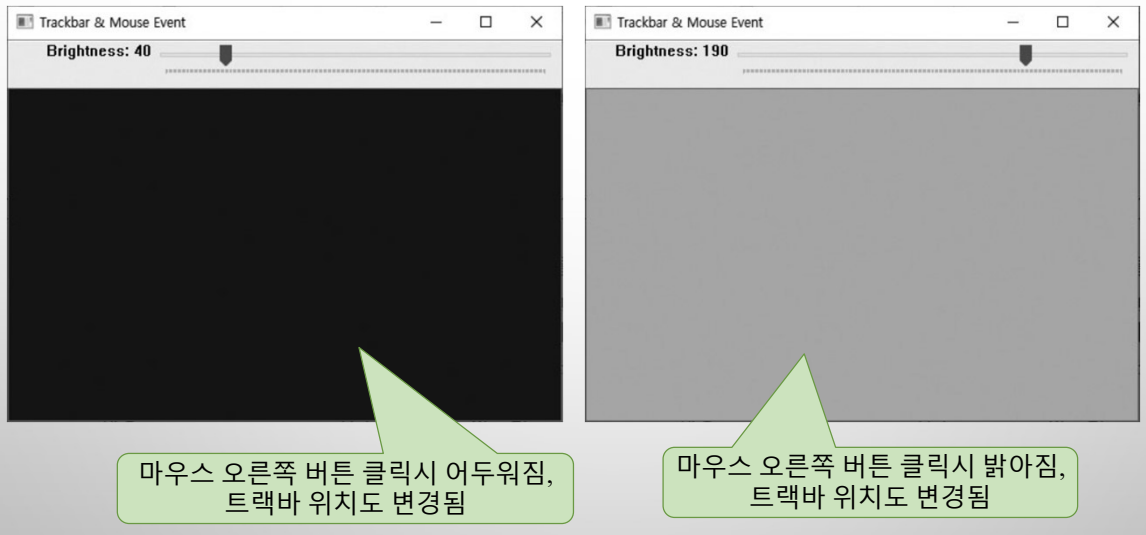
심화예제 4.2.4 마우스 및 트랙바 이벤트 사용 – 06.event_mouse_trackbar.py

```
01  import numpy as np
02  import cv2
03
04  def onChange(value): ···                                  # 트랙바 콜백 함수 – 소스 표시 생략
10
11  def onMouse(event, x, y, flags, param):                   # 마우스 콜백 함수
12      global image, bar_name                                # 전역 변수 참조
13
14      if event == cv2.EVENT_RBUTTONDOWN:                    # 마우스 우버튼
15          if (image[0][0] < 246): image = image + 10
16          cv2.setTrackbarPos(bar_name, title, image[0][0])  # 트랙바 위치 변경
17          cv2.imshow(title, image)
18
19      elif event == cv2.EVENT_LBUTTONDOWN:
20          if (image[0][0] >= 10):
21              image = image - 10
22          cv2.setTrackbarPos(bar_name, title, image[0][0])  # 트랙바 위치 변경
23          cv2.imshow(title, image)
24
25  image = np.zeros((300, 500), np.uint8)
26  title = "Trackbar & Mouse Event"                          # 윈도우 이름
27  bar_name = 'Brightness'                                   # 트랙바 이름
28  cv2.imshow(title, image)
29
30  cv2.createTrackbar(bar_name, title, image[0][0], 255, onChange)  # 트랙바 콜백 함수
31  cv2.setMouseCallback(title, onMouse)                      # 마우스 콜백 함수 등록
32  cv2.waitKey(0)                                            # 키 입력 대기
33  cv2.destroyAllWindows()                                   # 모든 윈도우 닫기
```

19

4.2.3 트랙바 이벤트 제어

◆실행결과



20

4.3 그리기 함수

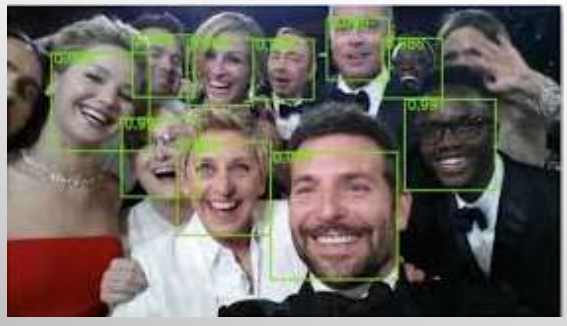
◆영상처리 프로그래밍 과정에서 해당 알고리즘 적용시 결과 확인 필요

❖ 얼굴 검출 알고리즘을 적용했을 때,

●전체 영상 위에 검출한 얼굴 영역을 사각형이나 원으로 표시

❖ 차선 확인하고자 직선 검출 알고리즘을 적용했을 때,

●차선을 정확하게 검출했는지 확인하기 위해 도로 영상 위에 선으로 표시



21

4.3.1 직선 및 사각형 그리기

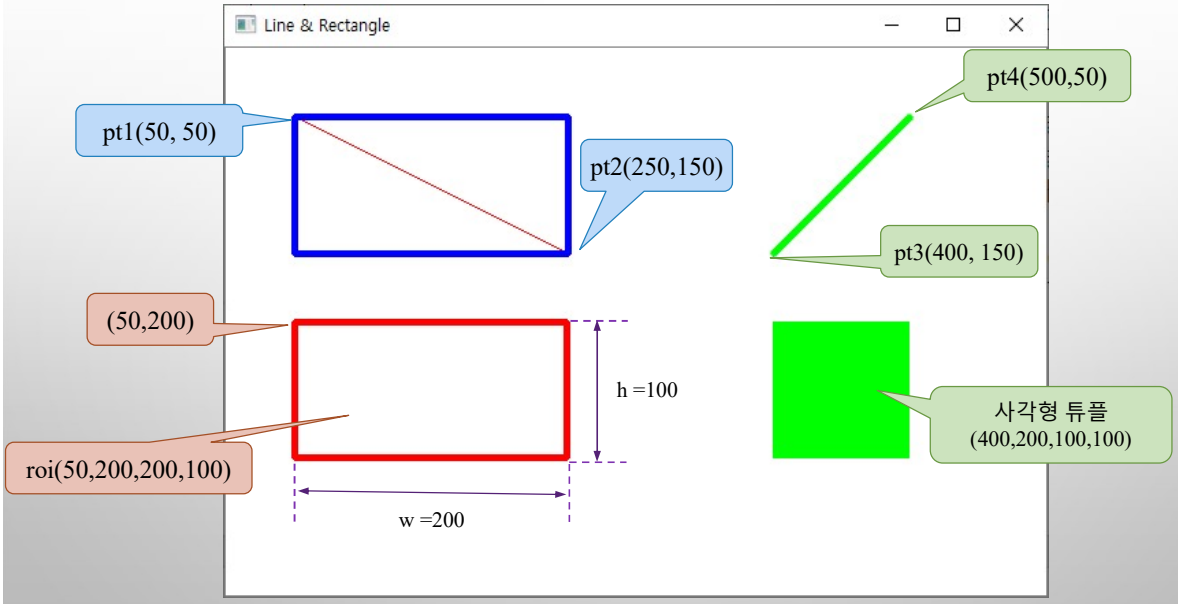
예제 4.3.1 직선 & 사각형 그리기 – 07.draw_line_rect.py

```
01  import numpy as np
02  import cv2
03
04  blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)          # 색상 선언
05  image = np.zeros((400, 600, 3), np.uint8)                         # 3채널 컬러 영상 생성
06  image[:] = (255, 255, 255)                                        # 3채널 흰색
07
08  pt1, pt2 = (50, 50),  (250, 350)                                  # 좌표 선언 – 정수형 튜플
09  pt3, pt4 = (400, 150), (500, 50)
10  roi = (50, 200, 200, 100)                                         # 사각형 영역 – 4원소 튜플
11
12  ## 직선 그리기
13  cv2.line(image, pt1, pt2, red)
14  cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)                  # 계단 현상 감소선
15
16  ## 사각형 그리기
17  cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)               # 4방향 연결선
18  cv2.rectangle(image, roi, red, 3, cv2.LINE_8 )                    # 8방향 연결선
19  cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)     # 내부 채움
20
21  cv2.imshow("Line & Rectangle", image)                             # 윈도우에 영상 표시
22  cv2.waitKey(0)
23  cv2.destroyAllWindows()                                           # 모든 열린 윈도우 닫기
```

22

4.3.1 직선 및 사각형 그리기

◆실행결과



23

4.3.2 글자 쓰기

◆cv2.putText()

❖ 행렬의 특정 위치에 원하는 글자를 써서 영상으로 표시하고 싶을 때 사용

◆형식

24

4.3.2 글자 쓰기

함수 설명		
cv2.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]) → img ■ 설명: text 문자열을 org 좌표에 color 색상으로 그린다.		
인수 설명	■ img	문자열을 작성할 대상 행렬(영상)
	■ text	작성할 문자열
	■ org	문자열의 시작 좌표, 문자열에서 가장 왼쪽 하단을 의미
	■ fontFace	문자열의 폰트
	■ fontScale	글자 크기 확대 비율
	■ color	글자의 색상
	■ thichness	글자의 굵기
	■ lineTpye	글자 선의 형태
	■ bottomLeftOrigin	영상의 원점 좌표 설정(True- 좌하단 왼쪽,

〈표 4.3.1〉 문자열의 폰트(fontFace)에 대한 옵션과 의미

옵션	값	설명
cv2.FONT_HERSHEY_SIMPLEX	0	중간 크기 산세리프 폰트
cv2.FONT_HERSHEY_PLAIN	1	작은 크기 산세리프 폰트
cv2.FONT_HERSHEY_DUPLEX	2	2줄 산세리프 폰트
cv2.FONT_HERSHEY_COMPLEX	3	중간 크기 세리프 폰트
cv2.FONT_HERSHEY_TRIPLEX	4	3줄 세리프 폰트
cv2.FONT_HERSHEY_COMPLEX_SMALL	5	COMPLEX 보다 작은 크기
cv2.FONT_HERSHEY_SCRIPT_SIMPLEX	6	필기체 스타일 폰트
cv2.FONT_HERSHEY_SCRIPT_COMPLEX	7	복잡한 필기체 스타일
cv2.FONT_ITALIC	16	이탤릭체를 위한 플래그

◆세리프 체
- ❖ 글자의 획 끝에 날카롭게 튀어나온 글자체
- ❖ 명조체, 궁서체 등

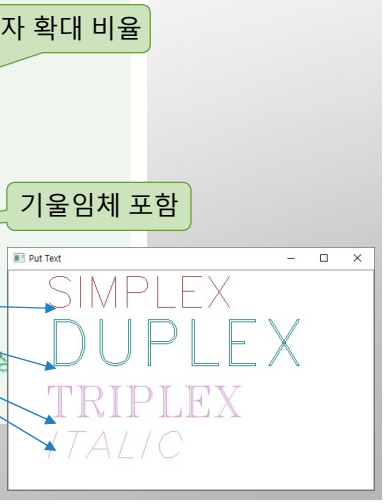
◆산세리프 체
- ❖ 날카로운 장식선이 없는 글자체
- ❖ 돋움체 , 고딕체

25

4.3.2 글자 쓰기

예제 4.3.2 글자 쓰기 – 08.put_text.py

```
01  import numpy as np
02  import cv2
03
04  olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)        # 색상 지정
05  pt1, pt2 =  (50, 230), (50, 310)                                            # 문자열 위치 좌표
06
07  image = np.zeros((350, 500, 3), np.uint8)
08  image.fill(255)
09
10  cv2.putText(image, 'SIMPLEX', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, brown)
11  cv2.putText(image, 'DUPLEX', (50, 130), cv2.FONT_HERSHEY_DUPLEX, 3, olive)
12  cv2.putText(image, 'TRIPLEX', pt1, cv2.FONT_HERSHEY_TRIPLEX, 2, violet)
13  fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC                         # 글자체 상수
14  cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)
15
16  cv2.imshow('Put Text', image)                                               # 윈도우 이름 지정 및 영상
17  cv2.waitKey(0)                                                              # 키이벤트 대기
```



26

4.3.3 원 그리기

◆cv2.circle() 함수

❖ 원의 중심 좌표(center), 반지름(radius), 선의 색상(color)은 반드시 지정

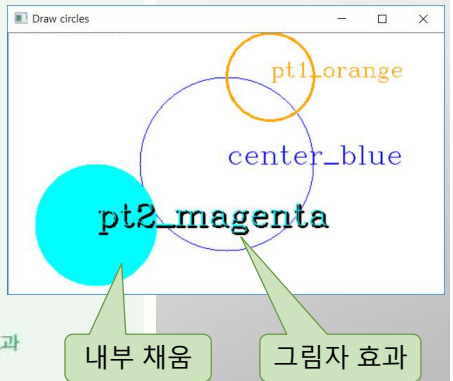
함수 설명		
cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]]) → img ■ 설명: center를 중심으로 radius 반지름의 원을 그린다.		
인수 설명	■ img	원을 그릴 대상 행렬(영상)
	■ center	원의 중심 좌표
	■ radius	원의 반지름
	■ color	선의 색상
	■ thickness	선의 두께
	■ lineType	선의 형태 , cv2.line() 함수의 인수와 동일
	■ shift	좌표에 대한 비트 시프트 연산

27

4.3.3 원 그리기

예제 4.3.3 원 그리기 – 09.draw_circle.py

```
01  import numpy as np
02  import cv2
03
04  orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
05  white, black = (255, 255, 255), (0, 0, 0)
06  image = np.full((300, 500, 3), white, np.uint8)          # 컬러 영상 생성 및 초기화
07
08  center = (image.shape[1]//2, image.shape[0]//2)          # 영상 중심 좌표 – 역순 구성
09  pt1, pt2 = (300, 50), (100, 220)
10  shade = (pt2[0] + 2, pt2[1] + 2)                         # 그림자 좌표
11
12  cv2.circle(image, center, 100, blue)                     # 원 그리기
13  cv2.circle(image, pt1, 50, orange, 2)
14  cv2.circle(image, pt2, 70, cyan, -1)                     # 원 내부 채움
15
16  font = cv2.FONT_HERSHEY_COMPLEX;
17  cv2.putText(image, 'center_blue', center, font, 1.0, blue)
18  cv2.putText(image, 'pt1_orange', pt1, font, 0.8, orange)
19  cv2.putText(image, 'pt2_cyan', shade, font, 1.2, black, 2)      # 그림자 효과
20  cv2.putText(image, 'pt2_cyan', pt2, font, 1.2, cyan,    1)
21
22  cv2.imshow("Draw circles", image)
23  cv2.waitKey(0)
```

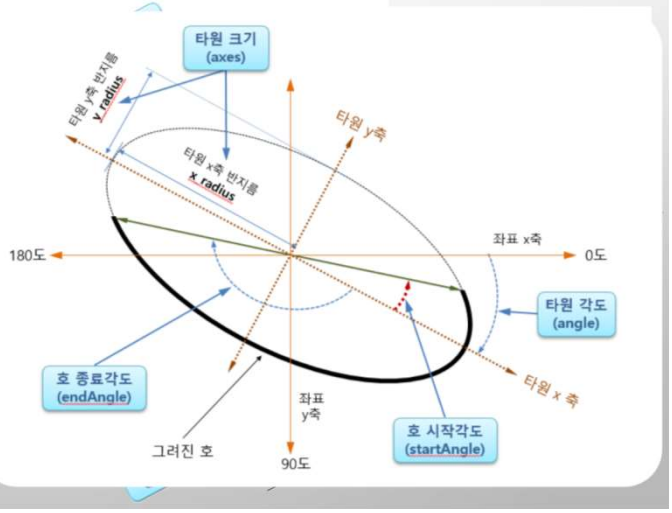


28

4.3.4 타원 그리기

◆타원 그리기 인수 의미

함수명과 반환형 및 인수 구조		
cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color[, thickness[, lineType[, shift]]]) →img ■설명: center를 중심으로 axes 크기의 타원을 그린다.		
인수 설명	■img	그릴 대상 행렬(영상)
	■center	원의 중심 좌표
	■axes	타원의 절반 크기(x축 반지름, y축 반
	■angle	타원의 각도 (3시 방향이 0도, 시계방
	■startAngle	호의 시작 각도
	■endAngle	호의 종료 각도
	■color	선의 색상
	■thickness	선의 두께
	■lineType	선의 형태
	■shift	좌표에 대한 비트 시프트

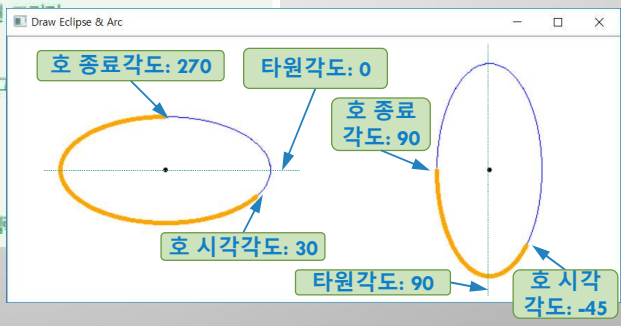


29

4.3.4 타원 그리기

예제 4.3.4 타원 및 호 그리기 – 10.draw_ellipse.py

```
01  import numpy as np
02  import cv2
03
04  orange, blue, white = (0, 165, 255), (255, 0, 0), (255,255,255) # 색상 지정
05  image = np.full((300, 700, 3), white, np.uint8)          # 3채널 행렬 생성 및 초기화
06
07  pt1, pt2 = (180, 150), (550, 150)                        # 타원 중심점
08  size = (120, 60)                                         # 타원 크기 – 반지름 값임
09
10  cv2.circle(image, pt1, 1, 0, 2)                          # 타원의 중심점(2화소 원) 표시
11  cv2.circle(image, pt2, 1, 0, 2)
12
13  cv2.ellipse(image, pt1, size,  0, 0, 360, blue, 1)       # 타원
14  cv2.ellipse(image, pt2, size, 90, 0, 360, blue, 1)
15  cv2.ellipse(image, pt1, size,  0, 30, 270, orange, 4)    # 호
16  cv2.ellipse(image, pt2, size, 90, -45, 90, orange, 4)
17
18  cv2.imshow("문자열", image)
19  cv2.waitKey()                                            # 키입
```

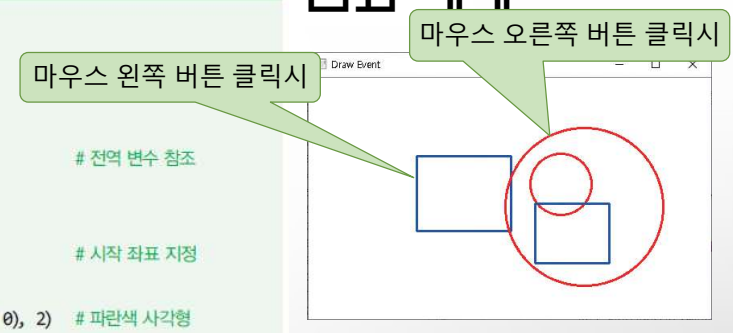


30

심화 예제

심화예제 4.3.5 마우스 이벤트 및 그리기 종합 – 11.event_draw.py

```
01  import numpy as np
02  import cv2
03
04  def onMouse(event, x, y, flags, param):
05      global title, pt                                    # 전역 변수 참조
06
07      if event == cv2.EVENT_LBUTTONDOWN:
08          if pt[0] < 0: pt = (x, y)                       # 시작 좌표 지정
09          else:
10              cv2.rectangle(image, pt, (x, y), (255, 0, 0), 2)  # 파란색 사각형
11              cv2.imshow(title, image)
12              pt = (-1, -1)
13
14      elif event == cv2.EVENT_RBUTTONDOWN:
15          if pt[0] < 0: pt = (x, y)
16          else:
17              dx, dy = pt[0] - x, pt[1] - y               # 두 좌표 간 간격
18              radius = int(np.sqrt(dx*dx + dy*dy))
19              cv2.circle(image, pt, radius, (0, 0, 255), 2)   # 빨간색 원
20              cv2.imshow(title, image)
21              pt = (-1, -1)                               # 시작 좌표 초기화
22
23  image = np.full((300, 500, 3), (255, 255, 255), np.uint8)   # 흰색 배경 영상
24
25  pt = (-1, -1)                                           # 시작 좌표 초기화
26  title = "Draw Event"
27  cv2.imshow(title, image)
28  cv2.setMouseCallback(title, onMouse)                    # 마우스 콜백 함수 등록
29  cv2.waitKey(0)
```



31

단원 요약

◆1. 콜백 함수는 개발자가 함수를 직접 호출하는 것이 아니라, 어떤 이벤트가 발생하거나 특정 시점에 도달했을 때, 시스템에서 개발자가 등록한 함수를 호출하는 방식이다. OpenCV의 cv2.setMouseCallback() 함수와 cv2.createTrackbar() 함수를 사용해서 마우스와 트렉바 이벤트를 처리하는 콜백 함수를 등록할 수 있다.

◆2. OpenCV에서 윈도우의 이름을 지정하는 함수는 cv2.namedWindow()이고, cv2.imshow() 함수로 지정된 윈도우에 행렬을 영상으로 표시할 수 있다.

◆3. cv2.waitKey() 함수와 cv2.waitKeyEx() 함수는 지정된 대기 시간 동안 키보드 키를 입력 받을 수 있는 함수로서, 키 이벤트를 처리하거나 윈도우 창을 바로 닫지 않고 대기시킬 때 사용된다.

◆4. OpenCV는 선과 사각형을 그려주는 cv2.line() 함수와 cv2.rectangle() 함수를 제공하며, 원과 타원을 그려주는 cv2.circle() 함수와 cv2.ellipse() 함수도 제공한다.

32