

CHAPTER 04

OpenCV 인터페이스 기초 /
사용자 인터페이스 및 I/O
처리

contents

- 4.1 윈도우 제어
- 4.2 이벤트 처리 함수
- 4.3 그리기 함수
- 4.4 영상파일 처리
- 4.5 비디오 처리
- 4.6 Matplotlib 패키지 활용

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 의미를 이해하기 쉬움
- OpenCV에서는 윈도우(window, 창)가 활성화된 상태에서만 마우스나 키보드 이벤트 감지

4.1 윈도우 제어

함수 설명

cv2.namedWindow(winname[, flags]) → None

- 설명: 윈도우 이름을 설정한 후, 해당 이름으로 윈도우 생성

인수 설명

■ winname(str)

원도우 이름

■ flags(int)

원도우의 크기 조정

옵션	값	설명
cv2.WINDOW_NORMAL	0	원도우 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

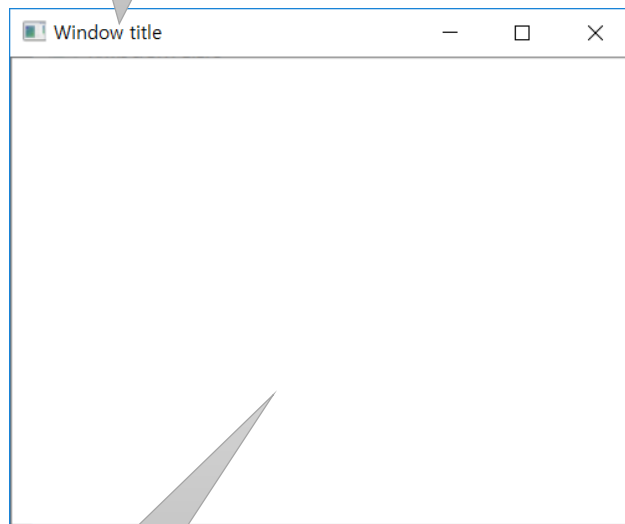
인수 설명	■ x, y	모니터 안에서 이동하려는 위치의 x, y 좌표
----------	--------	---------------------------

cv2.resizeWindow(winname, width, height) → None

- 설명: 윈도우의 크기를 재조정한다.

인수 설명	■ width, height	변경 윈도우의 가로, 세로 크기
----------	-----------------	-------------------

윈도우 이름



영상 표시 영역

4.1 윈도우 제어

라이브러리 импорт

예제 4.1.1 윈도우 이동 - 01.move_window.py

0 원소 행렬 생성

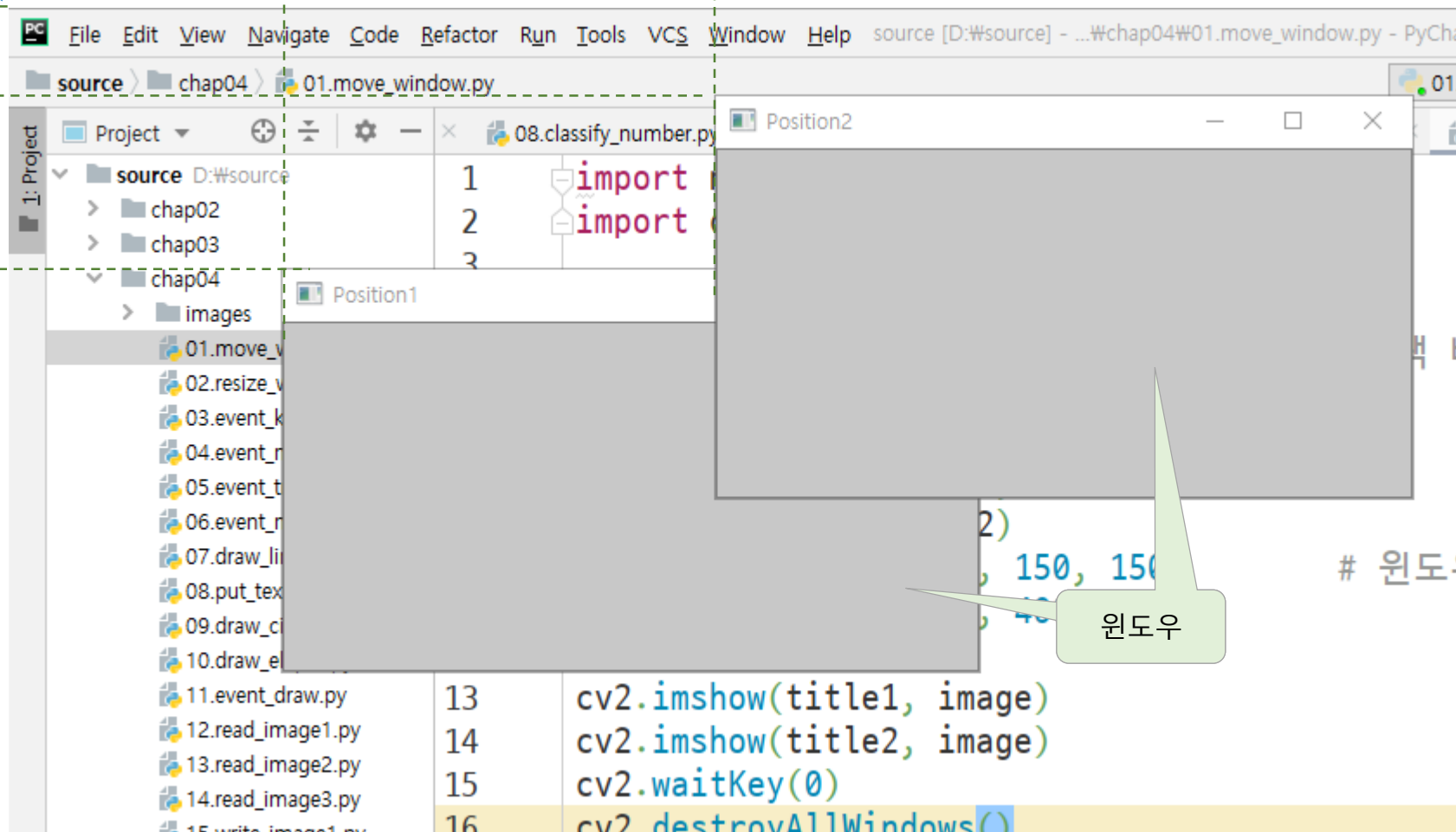
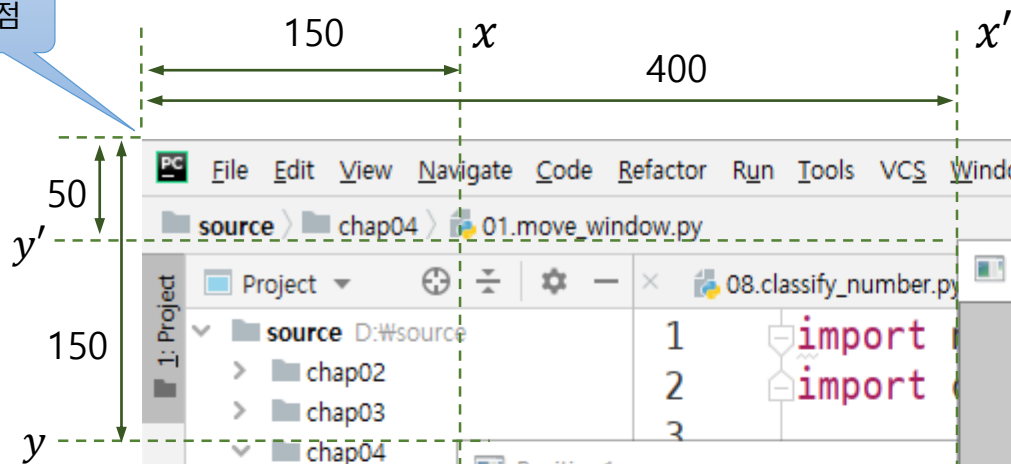
슬라이스 연산자로 행렬 원소값 지정

```
01 import numpy as np
02 import cv2
03
04 image = np.zeros((200, 400), np.uint8)
05 image[:] = 200
06
07 title1, title2 = 'Position1', 'Position2'
08 cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)
09 cv2.namedWindow(title2)
10 cv2.moveWindow(title1, 150, 150)
11 cv2.moveWindow(title2, 400, 50)
12
13 cv2.imshow(title1, image)
14 cv2.imshow(title2, image)
15 cv2.waitKey(0)
16 cv2.destroyAllWindows()
```

넘파이 라이브러리 импорт
OpenCV 라이브러리 импорт
행렬 생성
밝은 회색(200) 바탕 영상 생성
윈도우 이름
윈도우 생성 및 크기 조정 옵션
윈도우 이동 - 위치 지정
행렬 원소를 영상으로 표시
키 이벤트(key event) 대기
열린 모든 윈도우 파괴

4.1 윈도우 제어

원점



윈도우

4.1 윈도우 제어

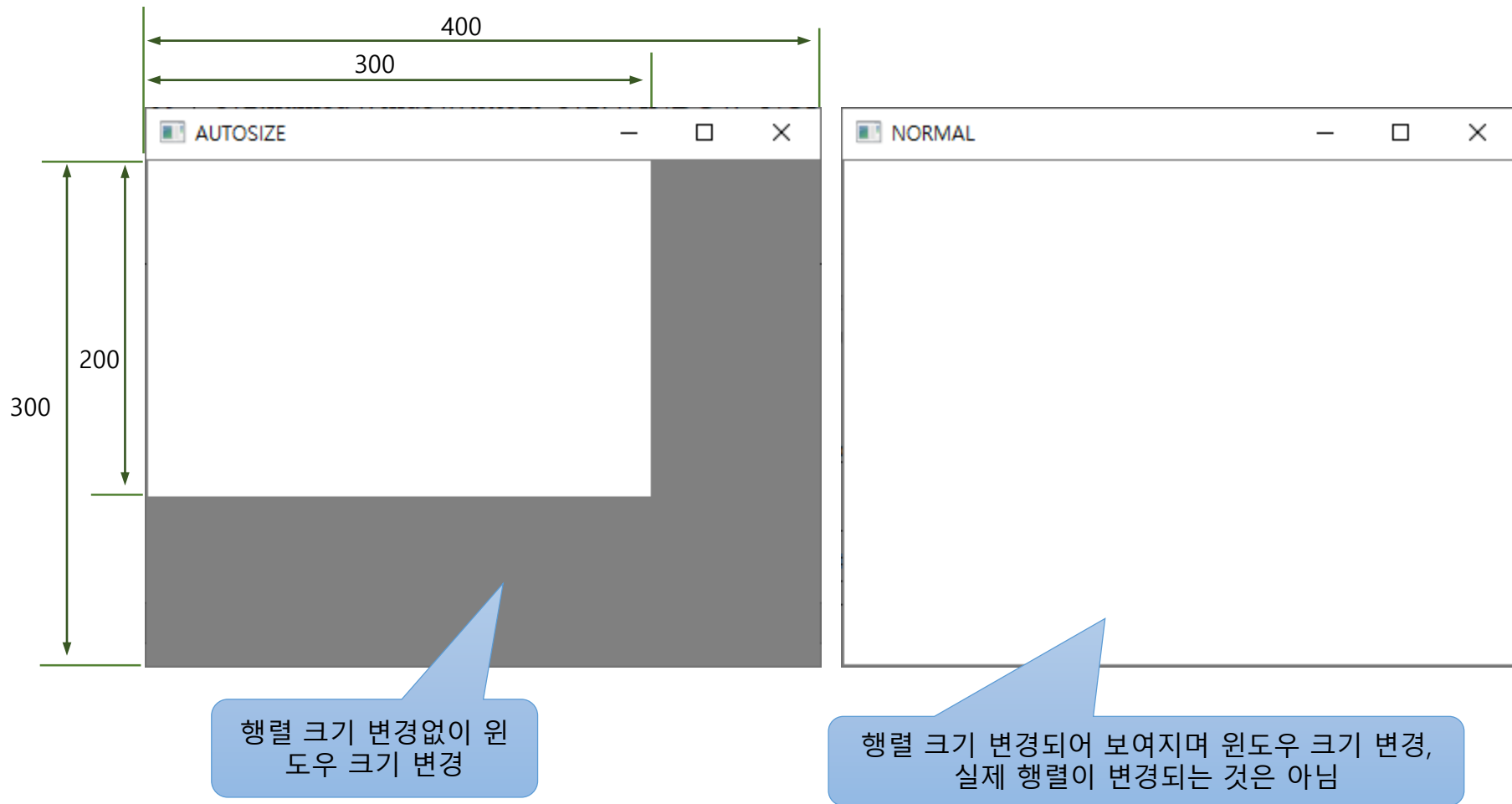
• WINDOW_AUTOSIZE vs. WINDOW_NORMAL

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()
```

np.ndarray.fill() 함수로 원소값 지정

4.1 윈도우 제어



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

4.2.1 키보드 이벤트 제어

함수 설명

`cv2.waitKey([, delay])` → `retval`

- 설명: `delay(ms: miliscond)` 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

4.2.1 키보드 이벤트 제어

예제 4.2.1 키 이벤트 사용 - 03.event_key.py

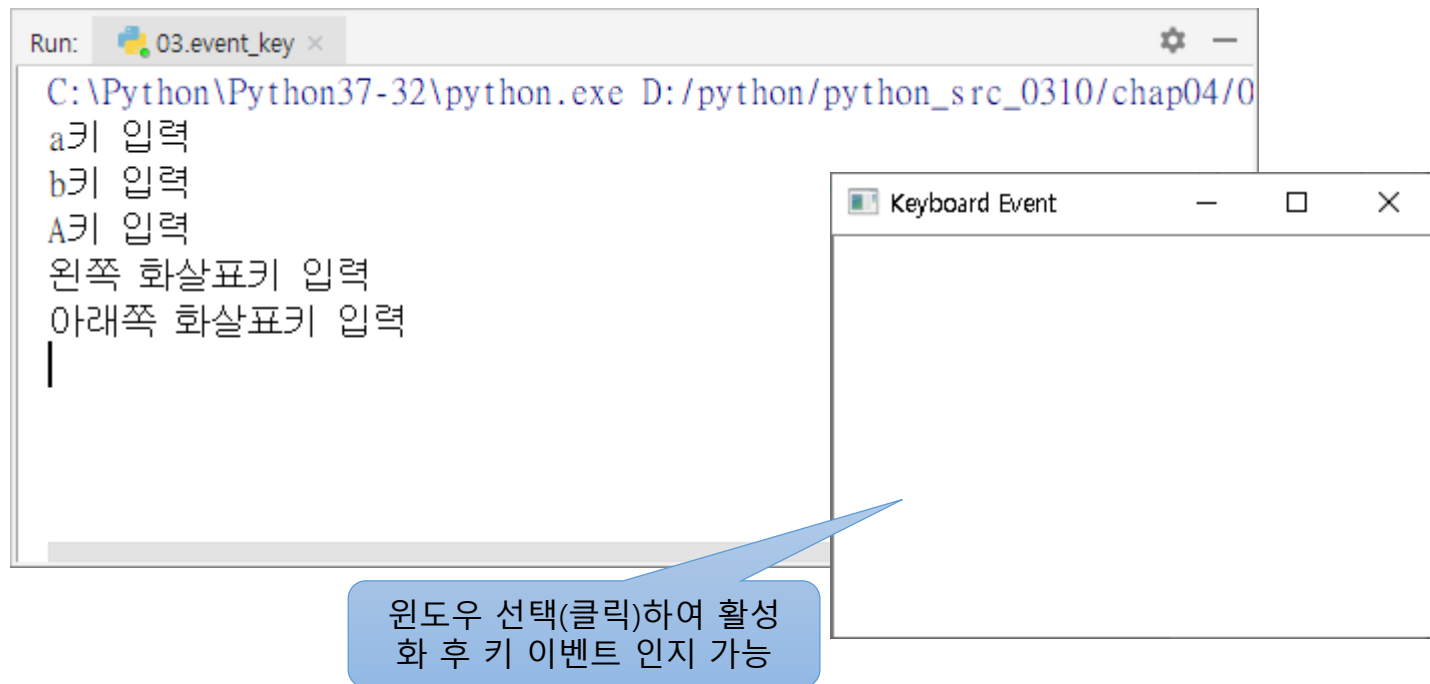
```
01 import numpy as np
02 import cv2
03
04 ## switch case문을 사전(dictionary)으로 구현
05 switch_case = {
06     ord('a'): "a키 입력",           # ord() 함수: 문자 → 아스키코드 변환
07     ord('b'): "b키 입력",
08     0x41: "A키 입력",
09     int("0x42", 16): "B키 입력",    # 0x42(16진수) → 10진수 변환
10     2424832: "왼쪽 화살표키 입력",  # 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

4.2.1 키보드 이벤트 제어

- 결과

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



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.		

〈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

4.2.2 마우스 이벤트 제어

예제 4.2.2

마우스 이벤트 사용 - 04.event_mouse.py

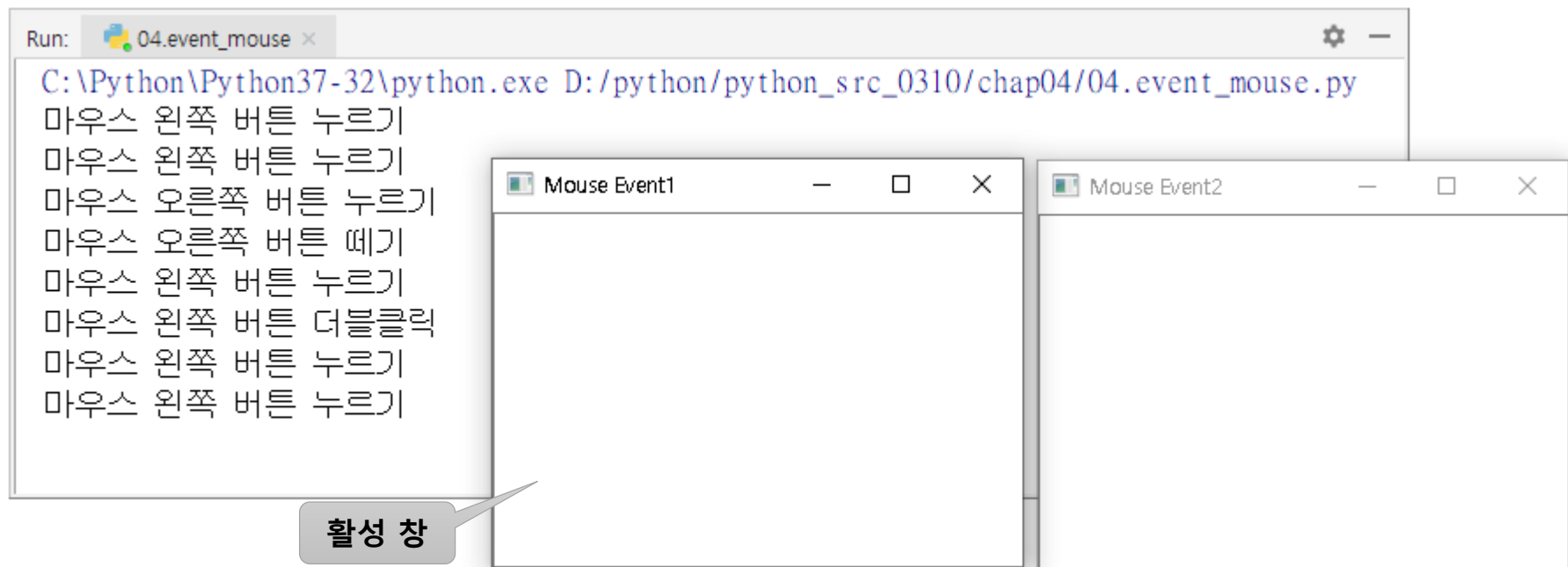
```
01 import numpy as np
02 import cv2
03
04 def onMouse(event, x, y, flags, param):           # 콜백 함수 - 이벤트 내용 출력
05     if event == cv2.EVENT_LBUTTONDOWN:          # 마우스 왼쪽 버튼
06         print("마우스 왼쪽 버튼 누르기")
07     elif event == cv2.EVENT_RBUTTONDOWN:
08         print("마우스 오른쪽 버튼 누르기")
09     elif 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

4.2.2 마우스 이벤트 제어

- 실행 결과



4.2.3 트랙바 이벤트 제어

- 트랙바(trackbar)

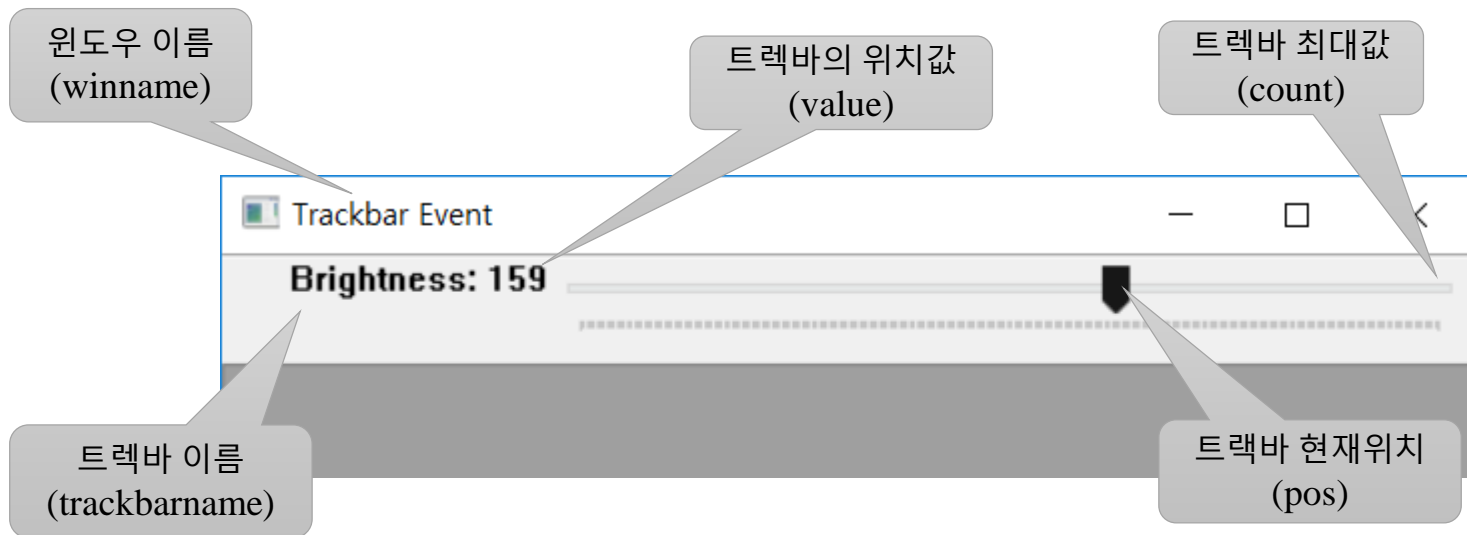
- 일정한 범위에서 특정한 값을 선택할 때 사용하는 일종의 스크롤바 혹은 슬라이더바

함수 설명	
cv2.createTrackbar(trackbarmame, winname, value count, onChange) → None	
■ 설명: 트랙바를 생성한 후, 지정한 윈도우에 추가하는 함수이다.	
인수 설명	<ul style="list-style-type: none">■ trackbarmame 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	<ul style="list-style-type: none">■ pos 트랙바 슬라이더 위치
cv2.getTrackbarPos(trackbarmame, winname) → retval	
■ 설명: 지정한 트랙바의 슬라이더 위치를 반환한다.	
cv2.setTrackbarPos(trackbarmame, winname, pos) → None	
■ 설명: 지정한 트랙바의 슬라이더 위치를 설정한다.	

4.2.3 트랙바 이벤트 제어

- 형식

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



4.2.3 트랙바 이벤트 제어 (실습)

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
```

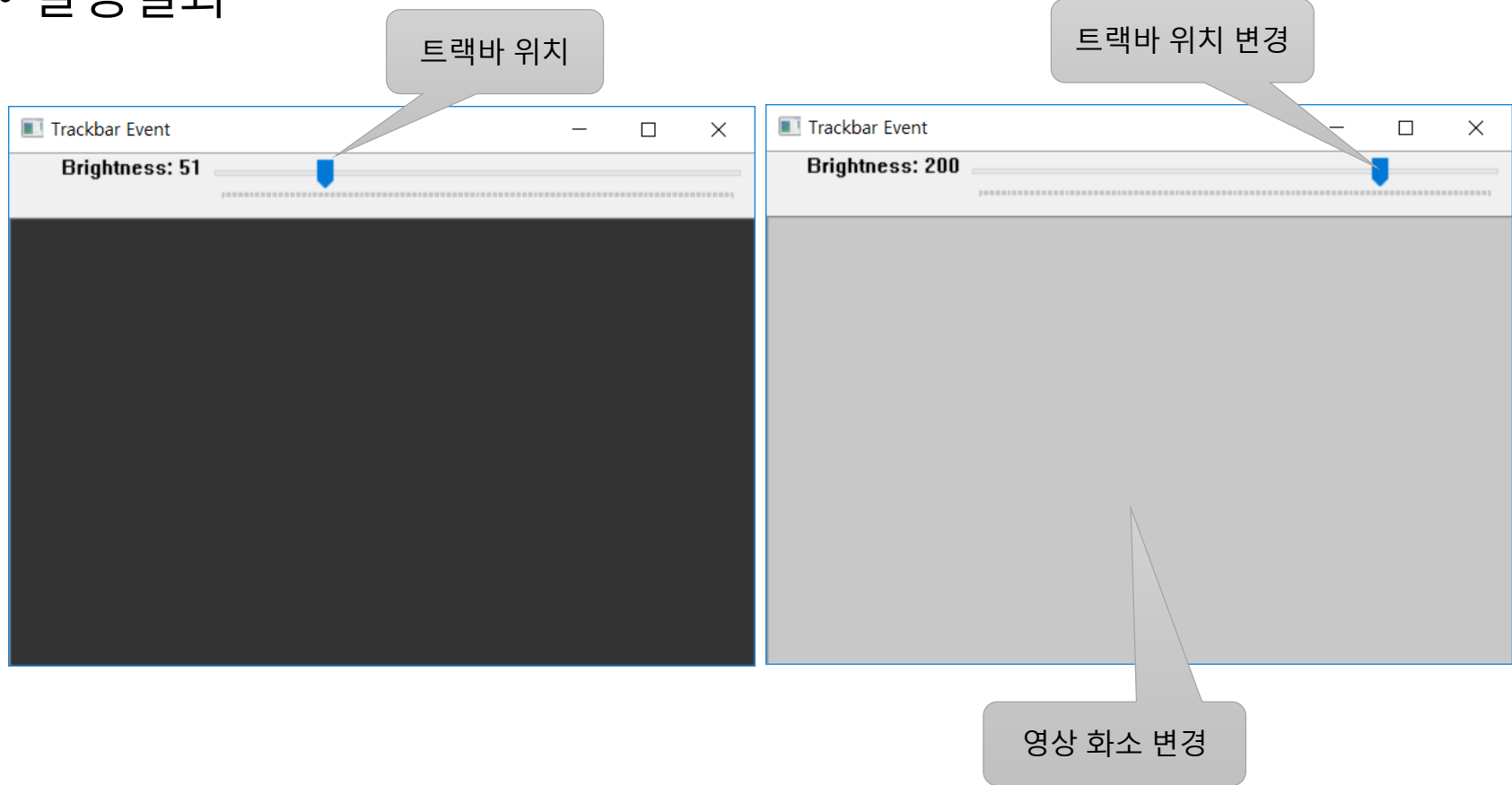
현재값

최대값

콜백함수

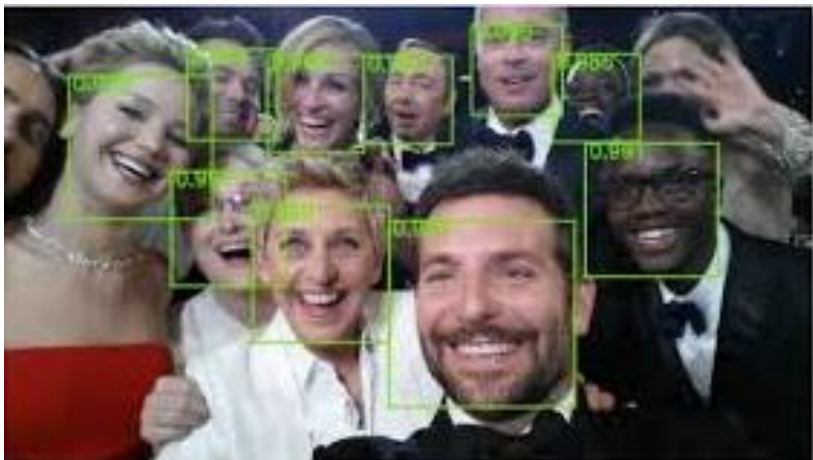
4.2.3 트랙바 이벤트 제어

- 실행결과



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



4.3.1 직선 및 사각형 그리기

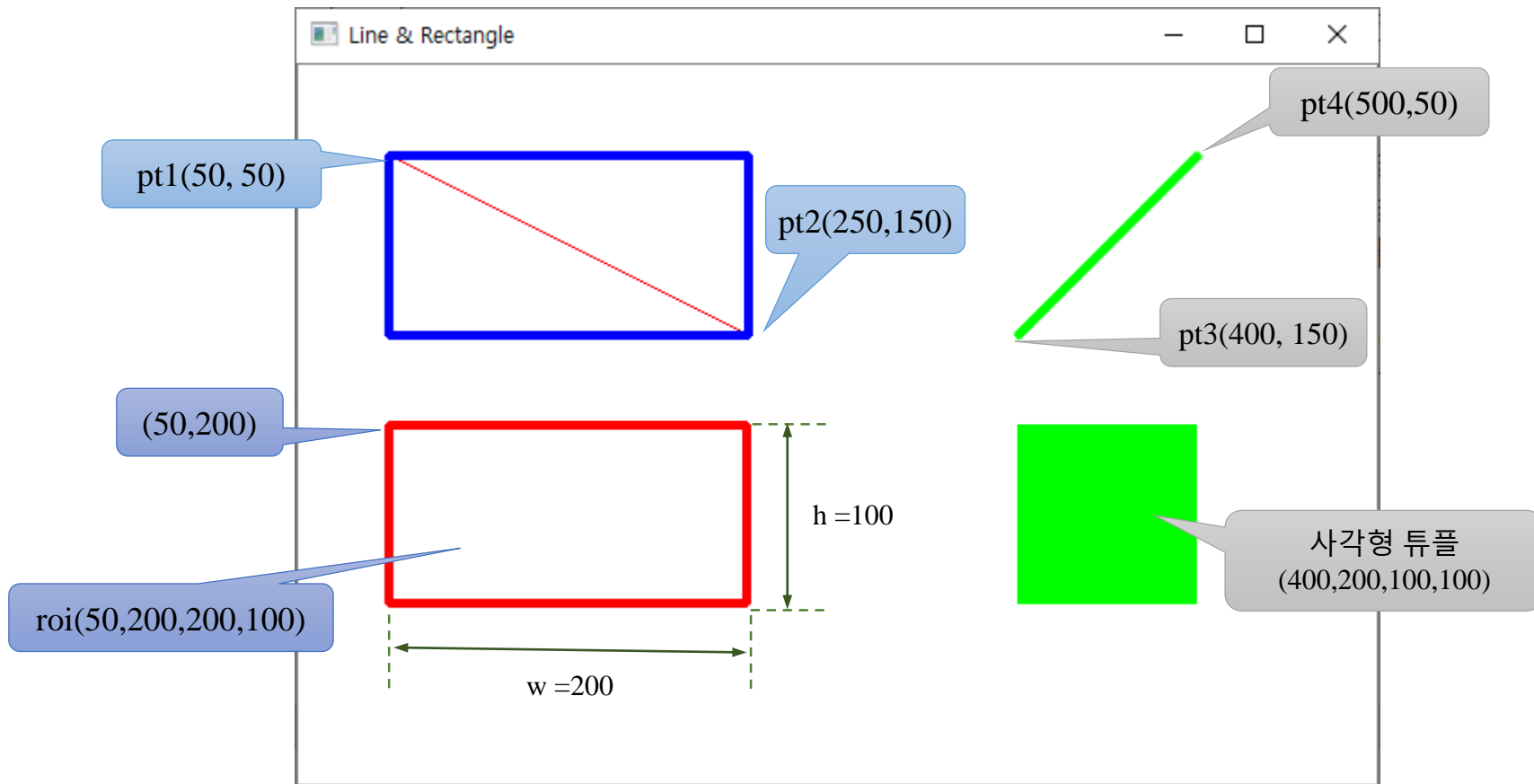
예제 4.3.1

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, 150)                             # 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
```

4.3.1 직선 및 사각형 그리기

- 실행결과



4.3. 그외 기능

- 교제 참고:
 - 글자 쓰기
 - 원 그리기
 - 타원 그리기

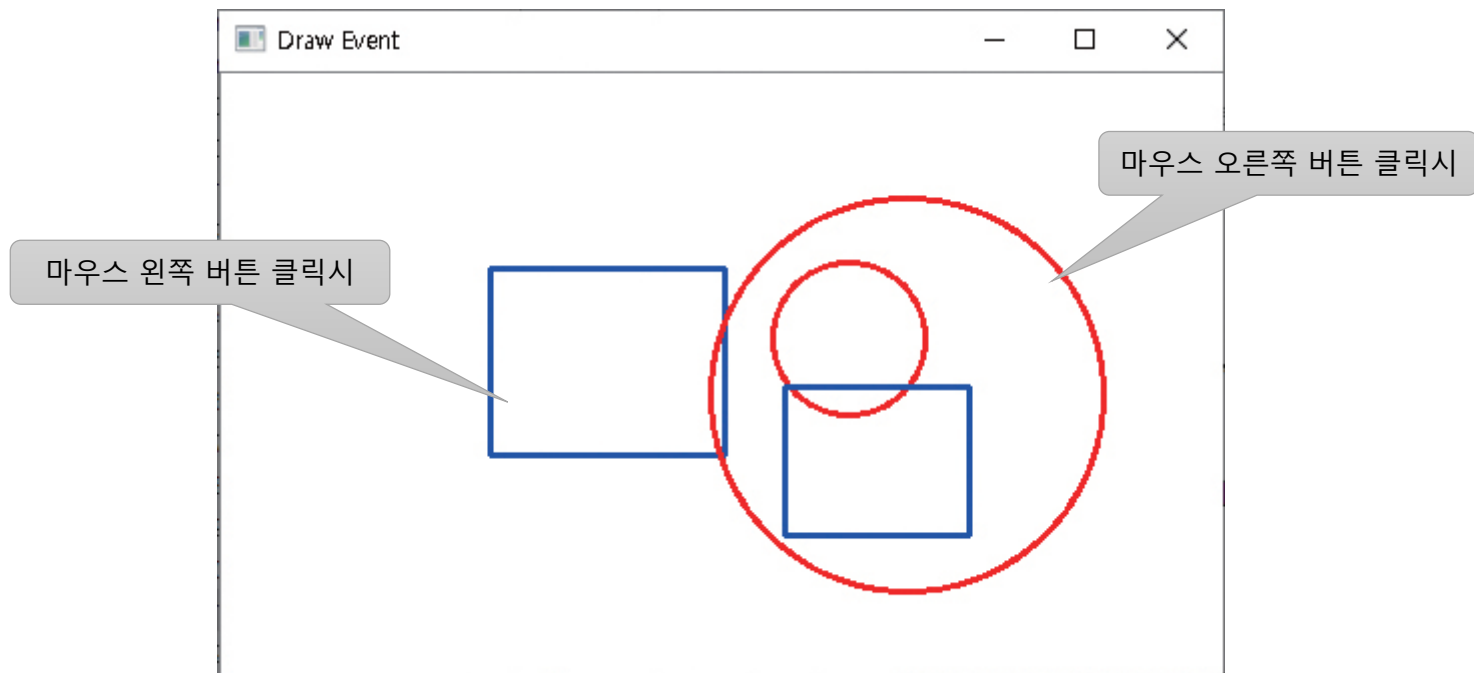
마우스 이벤트 및 그리기 종합 예제 (실습)

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)            # 시작 좌표 초기화
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)
```


- 실행 결과



4.4 영상파일 처리

- 영상처리 과정
 - 영상 파일을 읽어 들여 행렬에 저장
 - 행렬 연산 과정에서 행렬의 원소를 필요할 때마다 눈으로 직접 확인
 - 처리된 결과 행렬을 영상 파일로 저장 →
- 영상파일을 처리해 주는 함수와 활용 방법 필수

함수 설명

`cv2.imread(filename[, flags]) → retval`

■ 설명: 지정된 영상파일로부터 영상을 적재한 후, 행렬로 반환한다.

인수	■ filename	적재할 영상파일 이름(디렉터리 구조 포함)
설명	■ flags	적재한 영상을 행렬로 반환될 때 컬러 타입을 결정하는 상수

`cv2.imwrite(filename, img[, params]) → retval`

■ 설명: img 행렬을 지정된 영상파일로 저장한다.

인수 설명	■ filename	저장할 영상파일 이름(디렉터리 구조 포함), 확장자명에 따라 영상파일 형식 결정
	■ img	저장하고자 하는 행렬(영상)
	■ params	압축 방식에 사용되는 인수 쌍(paramId, paramValue)

4.4 영상파일 처리

〈표 4.4.1〉 행렬의 컬러 타입 결정 상수

옵션	값	설명
cv2.IMREAD_UNCHANGED	-1	입력 파일에 정의된 타입의 영상을 그대로 반환(알파(alpha) 채널 포함)
cv2.IMREAD_GRAYSCALE	0	명암도(grayscale) 영상으로 변환하여 반환
cv2.IMREAD_COLOR	1	컬러 영상으로 변환하여 반환
cv2.IMREAD_ANYDEPTH	2	입력 파일에 정의된 깊이(depth)에 따라 16비트/32비트 영상으로 변환, 설정되지 않으면 8비트 영상으로 변환
cv2.IMREAD_ANYCOLOR	4	입력 파일에 정의된 타입의 영상을 반환

4.4.1 영상파일 읽기

예제 4.4.1

영상파일 읽기1 - 12.read_image1.py

```
01 import cv2
02
03 def print_matInfo(name, image):                # 행렬 정보 출력 함수
04     if image.dtype == 'uint8':    mat_type = 'CV_8U'
05     elif image.dtype == 'int8':   mat_type = 'CV_8S'
06     elif image.dtype == 'uint16': mat_type = 'CV_16U'
07     elif image.dtype == 'int16':  mat_type = 'CV_16S'
08     elif image.dtype == 'float32': mat_type = 'CV_32F'
09     elif image.dtype == 'float64': mat_type = 'CV_64F'
10     nchannel = 3 if image.ndim == 3 else 1
11
12     ## depth, channel 출력
13     print("%12s: depth(%s), channels(%s) -> mat_type(%sC%d)"
14           % (name, image.dtype, nchannel, mat_type, nchannel))
15
16 title1, title2 = 'gray2gray', 'gray2color'    # 윈도우 이름
17 gray2gray = cv2.imread("images/read_gray.jpg", cv2.IMREAD_GRAYSCALE) # 명암도
18 gray2color = cv2.imread("images/read_gray.jpg", cv2.IMREAD_COLOR)    # 컬러 영상
19
20 ## 예외처리 -영상파일 읽기 여부 조사
21 if gray2gray is None or gray2color is None :
22     raise Exception("영상파일 읽기 에러")
23
```

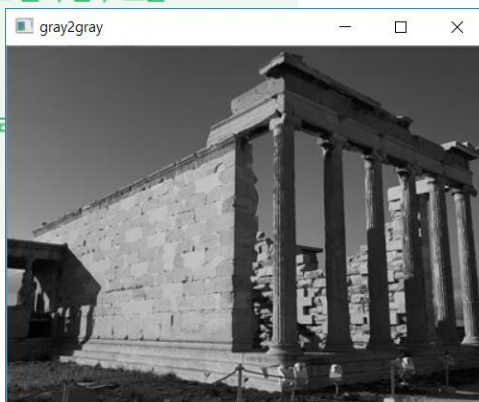
Run: 12.read_image1

```
C:\Python\python.exe D:/source/chap04/12.read_image1.py
Traceback (most recent call last):
  File "D:/source/chap04/12.read_image1.py", line 23, in <module>
    raise Exception("영상파일 읽기 에러")
Exception: 영상파일 읽기 에러
```

4.4.1 영상파일 읽기

```
24 print("행렬 좌표 (100, 100) 화소값")
25 print("%s %s" % (title1, gray2gray[100, 100])) # 행렬 내 한 화소값 표시
26 print("%s %s\n" % (title2, gray2color[100, 100]))
27
28 print_matInfo(title1, gray2gray) # 행렬 정보 출력 함수 호출
29 print_matInfo(title2, gray2color)
30
31 cv2.imshow(title1, gray2gray) # 행렬 정보
32 cv2.imshow(title2, gray2color)
33 cv2.waitKey(0)
```

• 실행결과



Run: 12.read_image1 x

C:\Python\python.exe D:/source/chap04/12.read_image1.py
행렬 좌표 (100, 100) 화소값
gray2gray 106
gray2color [106 106 106]

명암도 영상
(1채널 화소)

3채널 화소: 명암도 영상에서
변환해서 동일한 값임

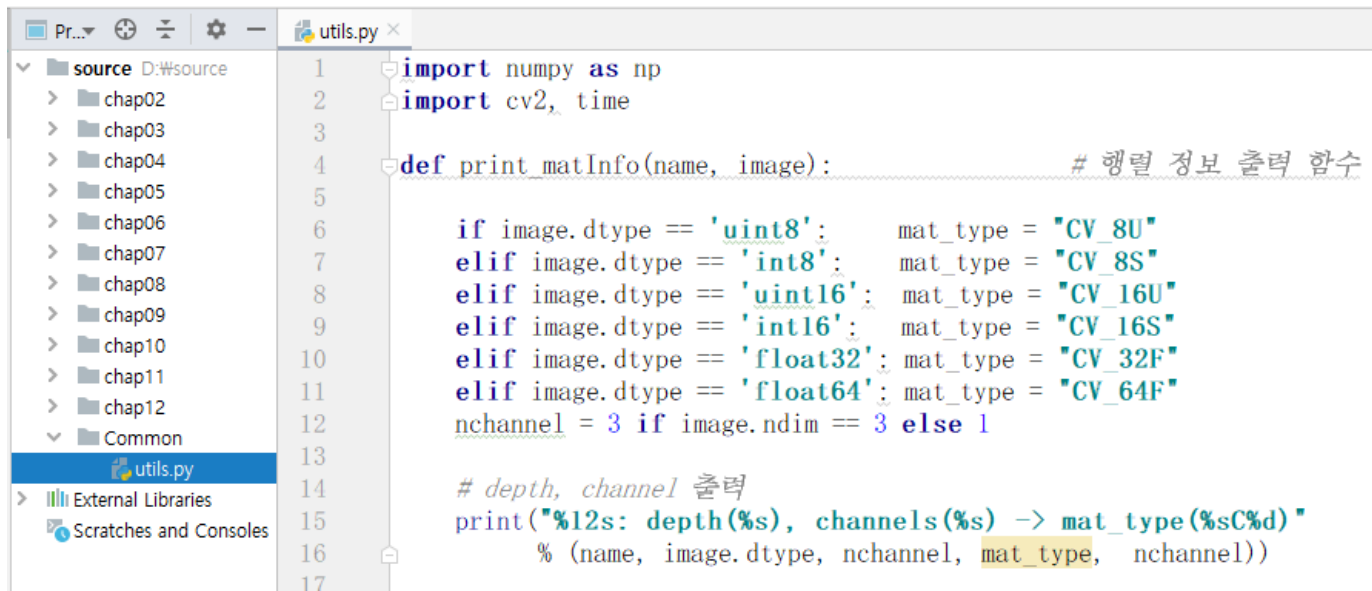
gray2gray: depth(uint8), channels(1) -> mat_type(CV_8UC1)
gray2color: depth(uint8), channels(3) -> mat_type(CV_8UC3)

행렬 자료형

모듈 импорт 하기

- 모듈 라이브러리 파일 만들기

- 프로젝트 루트 폴더(source)에서 마우스 우버튼 눌러 'Common' 이름으로 폴더 생성
 - 루트 폴더에 'Common' 폴더를 두는 것 - 다른 챕터의 소스 파일들에서도 모듈의 쉬운 사용위해
- 'Common' 폴더에서 파이썬 소스 파일(utils.py) 추가



```
1 import numpy as np
2 import cv2, time
3
4 def print_matInfo(name, image): # 행렬 정보 출력 함수
5
6     if image.dtype == 'uint8': mat_type = "CV_8U"
7     elif image.dtype == 'int8': mat_type = "CV_8S"
8     elif image.dtype == 'uint16': mat_type = "CV_16U"
9     elif image.dtype == 'int16': mat_type = "CV_16S"
10    elif image.dtype == 'float32': mat_type = "CV_32F"
11    elif image.dtype == 'float64': mat_type = "CV_64F"
12    nchannel = 3 if image.ndim == 3 else 1
13
14    # depth, channel 출력
15    print("%12s: depth(%s), channels(%s) -> mat_type(%sC%d)"
16          % (name, image.dtype, nchannel, mat_type, nchannel))
17
```

4.4.1 영상파일 읽기

예제 4.4.2

영상파일 읽기(컬러) - 13.read_image2.py

저자 생성 모듈의 함수 импорт

```
01 import cv2
02 from Common.utils import print_matInfo      # 행렬 정보 출력 함수 импорт
03
04 title1, title2 = 'color2gray', 'color2color'
05 color2gray = cv2.imread("images/read_color.jpg", cv2.IMREAD_GRAYSCALE)
06 color2color = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
07 if color2gray is None or color2color is None:      # 예외처리
08     raise Exception("영상파일 읽기 에러")
09
10 print("행렬 좌표(100, 100) 화소값")
11 print("%s %s" % (title1, color2gray[100, 100]))      # 한 화소값 표시
12 print("%s %s\n" % (title2, color2color[100, 100]))
13
14 print_matInfo(title1, color2gray)      # 행렬 정보 출력
15 print_matInfo(title2, color2color)
16 cv2.imshow(title1, color2gray)      # 행렬 정보 영상으로 띄우기
17 cv2.imshow(title2, color2color)
18 cv2.waitKey(0)
```

4.4.1 영상파일 읽기

• 실행결과

컬러 영상도 명암도 타입으로
읽으면 1채널 영상이 됨

Run: 13.read_image2 x

C:\Python\python.exe D:/source/chap04/13.read_image2.py

행렬 좌표 (100, 100) 화소값

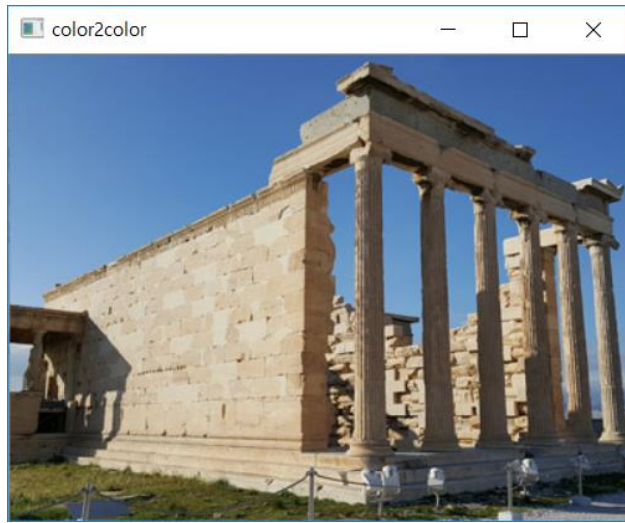
color2gray 137

color2color [197 145 98]

3채널 컬러 영상 화소값

color2gray: depth(uint8), channels(1) -> mat_type(CV_8UC1)

color2color: depth(uint8), channels(3) -> mat_type(CV_8UC3)



4.4.2 행렬을 영상파일로 저장

- cv2.imwrite() 함수
 - 행렬을 영상파일로 저장
 - 확장자에 따라서 JPG, BMP, PNG, TIF, PPM 등의 영상파일 포맷 저장 가능

〈표 4.4.3〉 압축 방식에 사용되는 params 인수 튜플(paramId, paramValue)의 예시

paramId	paramValue (기본값)	설명
cv2.IMWRITE_JPEG_QUALITY	0~100 (95)	JPG 파일 화질, 높은 값일수록 화질 좋음
cv2.IMWRITE_PNG_COMPRESSION	0~9 (3)	PNG 파일 압축 레벨, 높은 값일수록 용량은 적어지고, 압축 시간이 길어짐
cv2.IMWRITE_PXM_BINARY	0 or 1 (1)	PPM, PGM 파일의 이진 포맷 설정

4.4.2 행렬을 영상파일로 저장

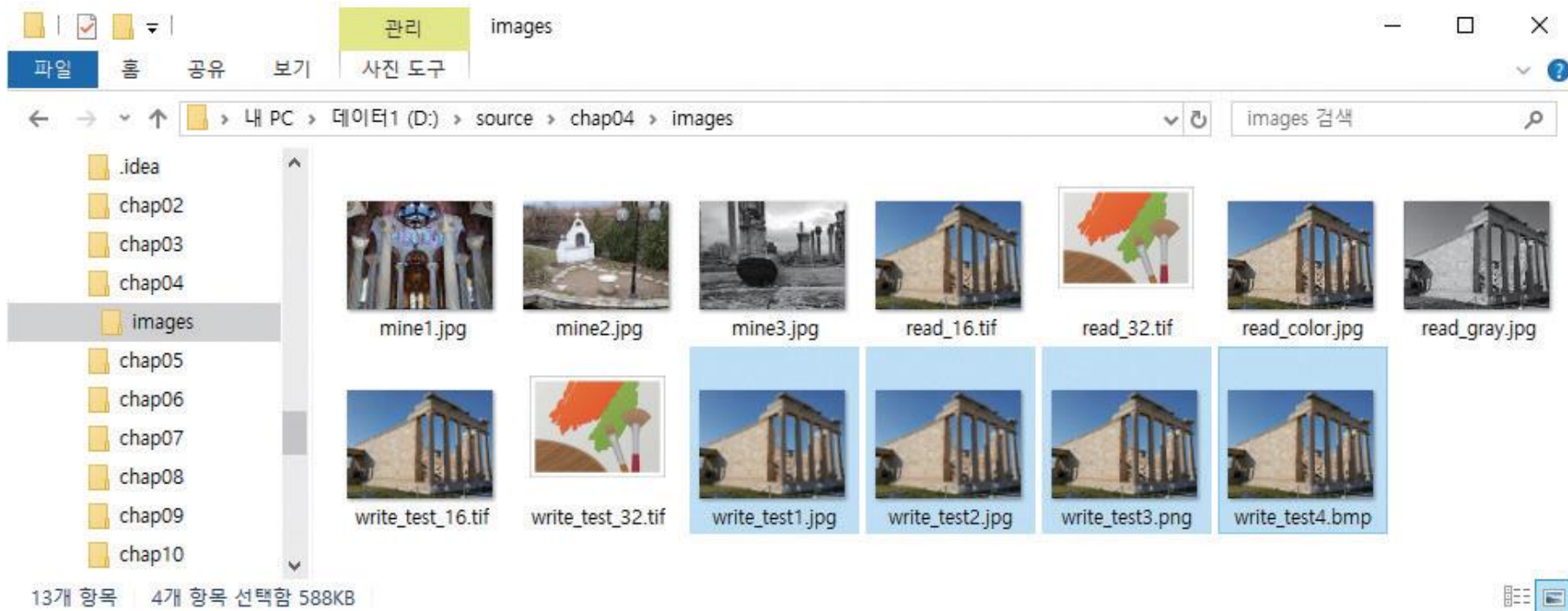
예제 4.4.3

행렬 영상 저장1 - 15.write_image1.py

```
01 import cv2
02
03 image = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
04 if image is None: raise Exception("영상파일 읽기 에러")      # 예외처리
05
06 params_jpg = (cv2.IMWRITE_JPEG_QUALITY, 10)                # JPEG 화질 설정
07 params_png = [cv2.IMWRITE_PNG_COMPRESSION, 9]              # PNG 압축 레벨 설정
08
09 ## 행렬을 영상파일로 저장
10 cv2.imwrite("images/write_test1.jpg", image)                # 디폴트는 95
11 cv2.imwrite("images/write_test2.jpg", image, params_jpg)    # 지정한 화질로 저장
12 cv2.imwrite("images/write_test3.png", image, params_png)
13 cv2.imwrite("images/write_test4.bmp", image)                # BMP 파일로 저장
14 print("저장 완료")
```

4.4.2 행렬을 영상파일로 저장

• 실행결과



4.5 비디오 처리

- 동영상 파일
 - 초당 30프레임 저장, 압축 필요 → 압축 코덱(codec) 사용함
- OpenCV는 동영상을 처리할 수 있는 클래스 제공
 - VideoCapture 클래스
 - 카메라에서 비대로 프레임 읽기
 - 카메라 속성 설정
 - 카메라 프레임을 동영상 파일로 저장
- (분량 상 실습 진행하지 않습니다. 관심있는 학생은 교재 p130 참고.)

4.6 Matplotlib 패키지 활용

- Matplotlib 라이브러리
 - 파이썬에서 데이터를 차트나 그래프로 그려주는 라이브러리
- pyplot 모듈
 - Matplotlib 라이브러리의 하나의 모듈
 - 매트랩의 수치해석 소프트웨어의 시각화 명령을 거의 그대로 사용가능한 명령어 집합 제공
 - 주피터 노트북이나 구글의 Colab 에서 간단히 결과 확인할 때
 - OpenCV의 cv2.imshow() 함수를 사용하지 않고 Matplotlib 패키지를 사용하는 것이 편리

4.6 Matplotlib 패키지 활용

- 함수 설명 (p145 참조)

함수 설명		
pyplot.figure(num=None, figsize=None, dpi=None) → matplotlib.figure.Figure		
■ 설명: 그림(Figure) 객체를 생성해서 플롯(plot)을 그릴 수 있게 한다.		
인수 설명	■ num	그림 이름, 정수 or 문자열(지정 안하면 자동 증가 숫자)
	■ figsize	그림 크기(가로, 세로), 인치 단위(실수형)
	■ dpi	그림 해상도(정수형)
pyplot.imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None) → (matplotlib.image.AxesImage)		
■ 설명: 데이터 X를 그래프에 영상으로 그려준다.		
인수 설명	■ X	그리려는 데이터(ndarray 객체), 2차원 혹은 3차원
	■ cmap	컬러 맵, 정해진 컬러 조합에 따라서 색상 매칭
	■ aspect	그리지는 영상의 종횡비
	■ interpolation	그리지는 영상의 보간 방법(예: 'nearest', 'bilinear', 'bicubic')
	■ alpha	투명도(0: 완전 투명, 1: 완전 불투명)

4.6 Matplotlib 패키지 활용

예제 4.6.1 Matplotlib 라이브러리로 영상 표시 - 21.matplotlib.py

```
01 import cv2
02 import matplotlib.pyplot as plt          # pyplot 모듈 임포트
03
04 image = cv2.imread("images/matplot.jpg", cv2.IMREAD_COLOR)  # 영상 읽기
05 if image is None: raise Exception("영상파일 읽기 에러")      # 예외처리
06
07 rows, cols = image.shape[:2]              # 영상 크기 정보
08 rgb_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # 컬러 공간변환
09 gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # 명암도 영상 변환
10
11 fig = plt.figure(num=1, figsize=(3,4))      # 그림 생성
12 plt.imshow(image), plt.title('figure1- original(bgr)') # 그림표시 및 제목
13 plt.axis('off'), plt.tight_layout()         # 축 없음, 여백 없음
14
15 fig = plt.figure(figsize=(6,4))            # 그림 생성
16 plt.suptitle('figure2- pyplot image display') # 전체 제목 지정
17 plt.subplot(1,2,1), plt.imshow(rgb_img)     # 서브 플롯 그림
18 plt.axis([0,cols, rows,0]), plt.title('rgb color') # 축 범위, 서브 플롯 제목
19 plt.subplot(1,2,2), plt.imshow(gray_img, cmap='gray') # 서브 플롯 그림, 명암도로 표시
20 plt.title('gray_img2')
21 plt.show()                                  # 전체 그림 띄우기
```

1번 윈도우

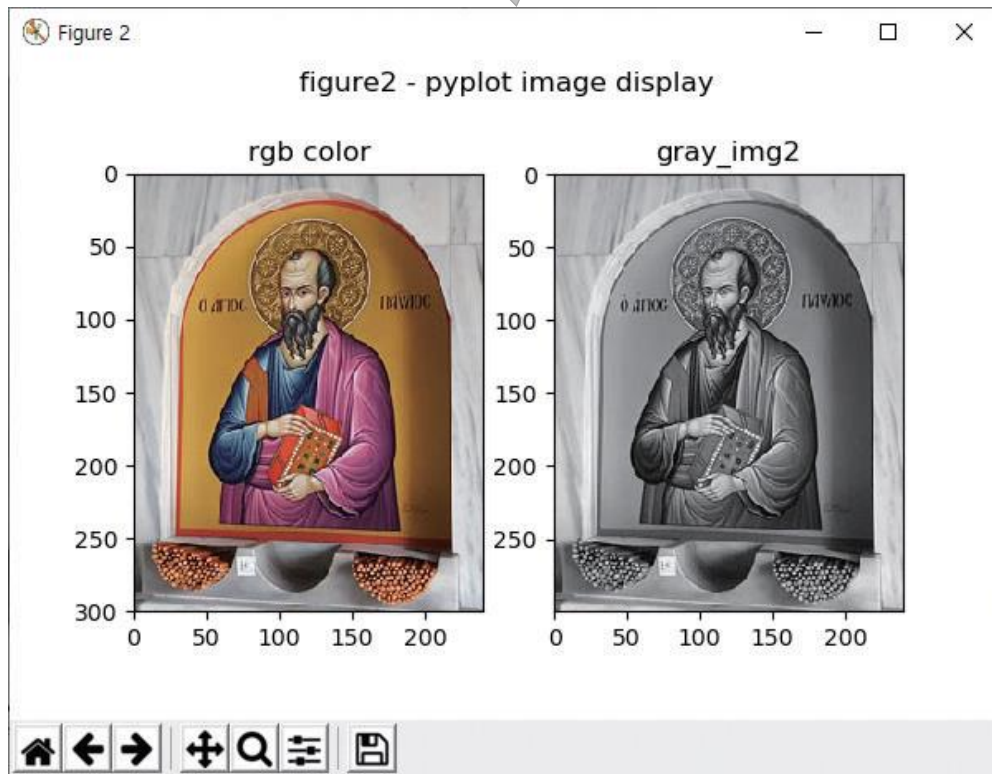
2번 윈도우 - 서브
플롯(2개 그림)

4.6 Matplotlib 패키지 활용

- 실행결과

색상 보색으로 나옴
Opencv - BGR 순
Matplotlib - RGB 순

서브플롯 - 1개 창에 2개 영상 표시



4. 실습 과제

- (과제) p157. 10번.
 - 다음의 마우스 이벤트 제어 프로그램을 작성하시오.
 - 1) 마우스 오른쪽 클릭 시 원(클릭좌표에서 반지름 20화소)을 그린다.
 - 2) 마우스 왼쪽 버튼 클릭 시 사각형(크기 30x30)을 그린다.
 - PPT 23쪽 실습 참조
- (보너스) p157. 11번.
 - 위 과제 문제에 다음을 추가하여 프로그램을 작성하시오.
 - 1) 트랙바를 추가해서 선의 굵기를 1~10픽셀로 조절한다.
 - 2) 트랙바를 추가해서 원의 반지름을 1~50픽셀로 조절한다.

4. 실습 규칙

- 실습 과제는 실습 시간내로 해결해야 합니다.
 - 해결 못한경우 실습 포인트를 얻지 못합니다.
 - -> 집에서 미리 연습하고 오길 권장합니다.
- 코드 공유/보여주기 금지. 의논 가능.
- 보너스문제까지 해결한 학생은 조기 퇴실 가능