CHAPTER 04
인터페이스 기초

김 찬, 윤 영 선

ckim.esw@gmail.com, ysyun@hnu.kr

정보통신공학과

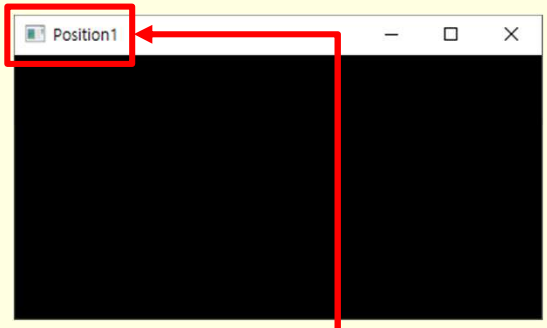
4. 인터페이스 기초

4.1 윈도우 제어

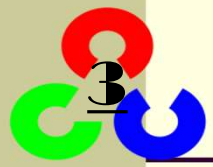
4.1.1 윈도우 생성

```
import numpy as np
import cv2

image = np.zeros((200, 400), np.uint8) # 0으로 된 200 x 400 행렬 생성
print(image)

title1 = 'Position1'                   # Window 이름 변수
cv2.namedWindow(title1)                # Window 이름 지정
cv2.imshow(title1, image)              # Window 이름과 이미지 출력
cv2.waitKey(0)                         # 키 입력 대기
cv2.destroyAllWindows()                # 실행 되어있는 창 모두 닫기
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```



RGB 색상에서 0은 검은색을 의미, 255는 흰색을 의미한다.



4. 인터페이스 기초

■ 4.1 윈도우 제어

◆ 4.1.2 윈도우 색상 변경

```
import numpy as np
import cv2

image = np.zeros((200, 400), np.uint8) # 0으로 된 200 x 400 행렬 생성
image[:] = 200                         # 변경하고 싶은 색 RGB 값 입력
print(image)

title1 = 'Changed'                     # Window 이름 변수
cv2.namedWindow(title1)                # Window 이름 지정
cv2.imshow(title1, image)              # Window 이름과 이미지 출력
cv2.waitKey(0)                         # 키 입력 대기
cv2.destroyAllWindows()                # 실행 되어있는 창 모두 닫기
```

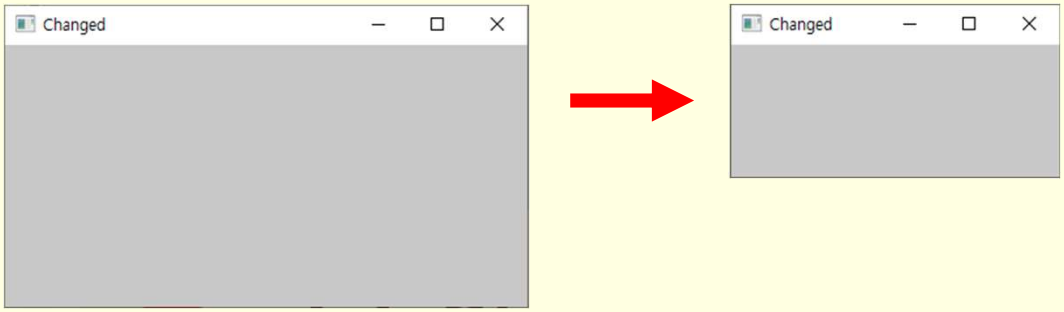
```
[[200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]
 ...
 [200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]]
```

RGB 색상에서 0은 검은색을 의미, 255는 흰색을 의미한다.

4. 인터페이스 기초

■ 4.1 윈도우 제어

◆ 4.1.2 윈도우 크기 변경




```
import numpy as np
import cv2

image = np.zeros((200, 400), np.uint8) # 0으로 된 200 x 400 행렬 생성
image[:] = 200                         # 변경하고 싶은 색 RGB 값 입력

title1 = 'Changed'                     # Window 이름 변수
cv2.namedWindow(title1)                # Window 이름 지정
                                       # Window 이름과 변환할 사이즈 입력
cv2.imshow(title1, image)              # Window 이름과 이미지 출력
cv2.waitKey(0)                         # 키 입력 대기
cv2.destroyAllWindows()                # 실행 되어있는 창 모두 닫기
```

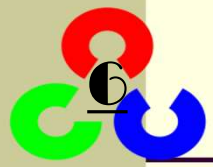
RGB 색상에서 0은 검은색을 의미, 255는 흰색을 의미한다.

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.1 키 이벤트 사용

```
import numpy as np
import cv2

image = np.zeros((200, 300), np.uint8)      # 0으로 된 200 x 300 행렬 생성
title1 = "Keyboard Event"                   # Window 이름 변수
cv2.namedWindow(title1)                     # Window 이름 지정
cv2.imshow(title1, image)                   # Window 이름과 이미지 출력

while True:                                 # 무한반복
    key = cv2.waitKeyEx(100)                # 100ms마다 키 입력 대기
    print(key)

    if key == 27:                           # esc버튼 눌렀을 때 입력되는 값
        break                               # 종료
cv2.destroyAllWindows()                     # 실행 되어있는 창 모두 닫기
```



4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.1 키 이벤트 사용

```
switch_case = {                          # 딕셔너리 형태로 키 이벤트 지정
    97: 'a키 입력',                       # ASCII 형태로 입력 받을 때
    ord('b'): 'b키 입력',                 # 98 == ASCII 형태의 'b'와 같다.
    0x41: 'A키 입력',                     # 16진수 형태로 A를 입력 받을 때
    int('0x42', 16): 'B키 입력',          # 16진수로 받은 입력을 10진수로 변환
    2424832: "왼쪽 화살표 키 입력",
    2490368: "윗쪽 화살표 키 입력",
    2555904: "오른쪽 화살표 키 입력",
    2621440: "아래쪽 화살표 키 입력"
}
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.1 키 이벤트 사용

```
while True:                                 # 무한반복
    key = cv2.waitKeyEx(100)                # 100ms마다 키 입력 받아오기

    if key == 27:                           # esc 키 이벤트가 감지되면
        break                               # while문 탈출

    try:                                    # 일단 실행
        result = switch_case[key]           # 입력한 키가 딕셔너리에 있는 값이면
        print(result)                       # 어떤 키 입력인지 출력

    except KeyError:                        # 딕셔너리에 없는 값이라면
        result = -1                         # -1 반환 (미 입력 시 어차피 -1)
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

조건

- 입력
  - 자유

- 조건
  - 1. Apple을 키 입력으로 출력하라.
  - 2. 잘못 입력했다면 “Del” 키 이벤트를 통해 모두 지워라.
  - 2. 종료 버튼은 esc가 아닌 키보드에 있는 “End” 키 이벤트로 종료 하도록 변경하라.

- 출력 예시

```
a
ap
app
appl
apple
A
Ap
App
Appl
Apple
```

소문자로 apple을 입력해서
Del 버튼으로 초기화 시킨 후
다시 입력한 모습

Hint
- ord(str) = 문자를 아스키코드로 변경
- chr(int) = 아스키코드를 문자로 변경

4. 인터페이스 기초

4.2 키보드 이벤트 제어

```
result = ""                          # 입력한 키 저장할 변수 선언
while True:                          # 무한반복
    key = cv2.waitKeyEx(100) # 100ms마다 키 입력 받아오기

    # del 키 이벤트가 감지되면
    # while문 탈출
    # end 키 이벤트가 감지되면
    # 초기화

    # 일단 실행
    # 입력한 키를 result에 저장
    # 어떤 키 입력인지 출력
    # 아스키코드에 없는 값이라면
    # 무시

cv2.destroyAllWindows()      # 반복 문 탈출 시 모든 창 닫기
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.2 마우스 이벤트 사용

```
import numpy as np
import cv2

def onMouse(event, x, y, flags, param):      # onMouse 함수에서 제공하는 이벤트
    print(event, x, y, flags, param)         # 출력을 통해 어떤 값이 나오나 보기

image = np.full((200, 300), 255, np.uint8)  # 255로 채워진 200 x 300 행렬 생성

title1 = “Mouse Event”
Cv2.imshow(title1, image)
cv2.setMouseCallback(title1, onMouse)        # 마우스 이벤트 콜백 함수
Cv2.waitKey(0)
Cv2.destroyAllWindows()
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.2 마우스 이벤트 사용

```
def onMouse(event, x, y, flags, param):

# flags
1  = 왼쪽 버튼 누르기
2  = 오른쪽 버튼 누르기
4  = 중간 버튼 누르기
8  = [Ctrl]키 누르기
16 = [Shift]키 누르기
32 = [Alt]키 누르기
```

```
# event
0  = 마우스 움직임
1  = 왼쪽 버튼 클릭
2  = 오른쪽 버튼 클릭
3  = 휠 버튼 누르기
4  = 왼쪽 버튼 떼기
5  = 오른쪽 버튼 떼기
6  = 휠 버튼 떼기
7  = 왼쪽 버튼 더블클릭
8  = 오른쪽 버튼 더블클릭
9  = 중간 버튼 더블클릭
10 = 마우스 휠
11 = 마우스 가로 휠
```

조건:

- 마우스 왼쪽, 오른쪽, 휠 클릭에 대한 이벤트 출력
  - ex) if ~~:

        print("마우스 왼쪽 버튼 누르기")

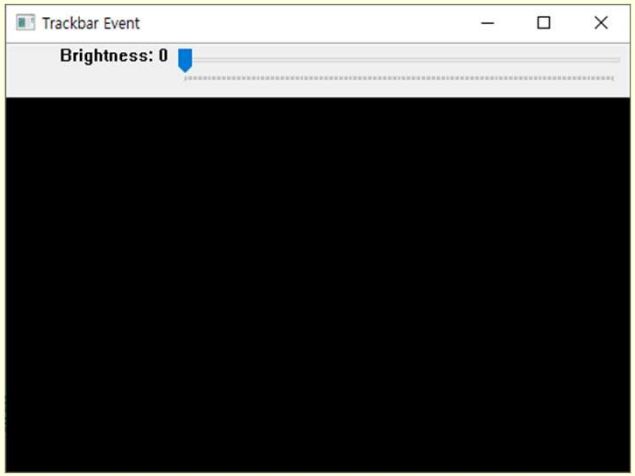
```
마우스 왼쪽 버튼 누르기
마우스 오른쪽 버튼 누르기
마우스 오른쪽 버튼 떼기
```

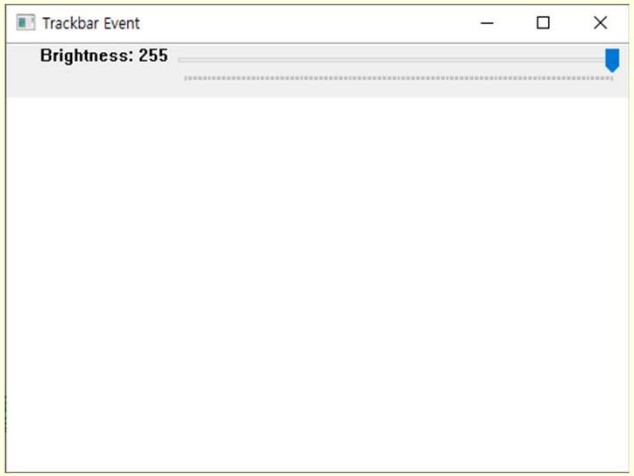
4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.3 트랙바 이벤트 사용

```
cv2.createTrackbar(trackbarName, windowName, value, count, onChange)

# trackbarName: 트랙바 이름
# windowName: 트랙바를 생성할 창 이름
# value: 트랙바 위치 초기값
# count: 트랙바 최댓값
# onChange: 트랙바 위치가 변경될 때마다 호출할 콜백 함수
```





4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.3 트랙바 이벤트 사용

```
import numpy as np
import cv2

def onChange(value):
    global image, title
    [                    ]          # 트랙바 제어로 인해 변경되는 이미지 색상 적용
    cv2.imshow(title, image)

image = np.zeros((300, 500), np.uint8)
title = "Trackbar Event"

cv2.imshow(title, image)
cv2.createTrackbar("Brightness", title, image[0][0], 255, onChange)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.3 트랙바 이벤트 사용

```
import numpy as np
import cv2

def onChange(value):
    global image, title
    image[:] = value
    cv2.imshow(title, image)

image = np.zeros((300, 500), np.uint8)
image[:] = 150                          # 현재 이미지 색상 지정
title = "Trackbar Event"

cv2.imshow(title, image)
cv2.createTrackbar("Brightness", title, image[0][0], 255, onChange)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

지정된 이미지 색상 가져와서
트랙바 위치 시작값으로 지정

4. 인터페이스 기초

4.3 그리기 함수

4.3.1 직선 및 사각형 그리기

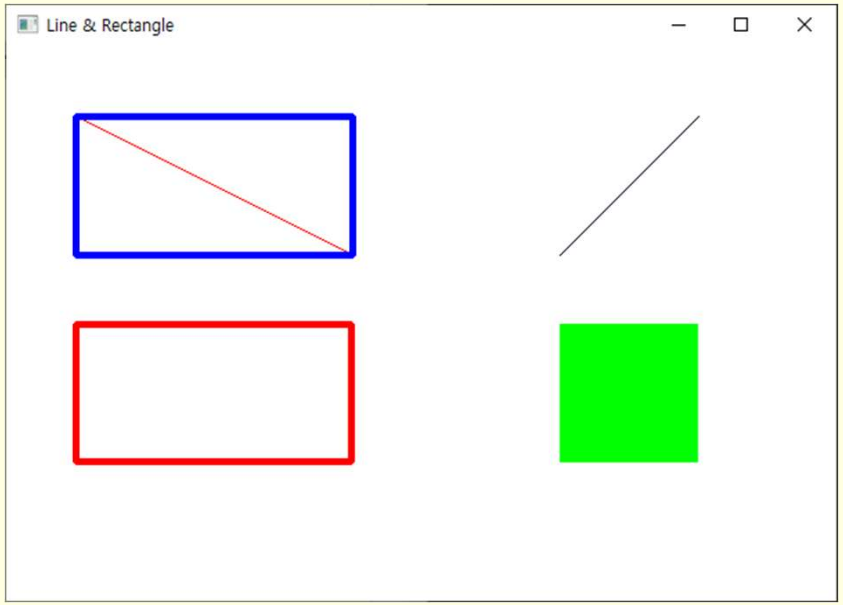
```
import numpy as np
import cv2

blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
image = np.zeros((400, 600, 3), np.uint8)
image[:] = (255, 255, 255)

pt1, pt2 = (50, 50), (250, 150)
pt3, pt4 = (400, 150), (500, 50)
roi = (50, 200, 200, 100)
```

4. 인터페이스 기초

4.3 그리기 함수

◆ 4.3.1 직선 및 사각형 그리기

```
# 사각형 그리기
cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)

# 직선 그리기
cv2.line(image, pt1, pt2, red)
cv2.line(image, pt3, pt4, cv2.LINE_AA)

cv2.imshow("Line & Rectangle", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



4. 인터페이스 기초

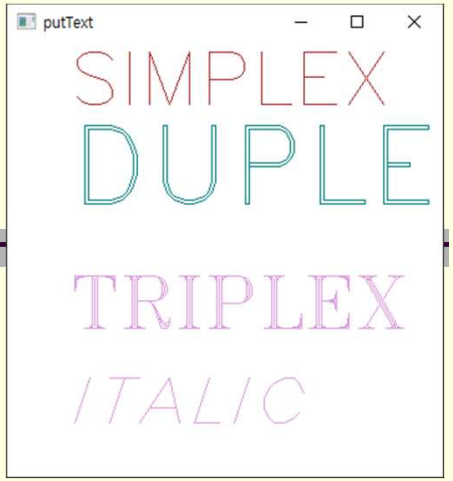
■ 4.3 그리기 함수

◆ 4.3.2 글자 쓰기

```
import numpy as np
import cv2

...

cv2.imshow('putText', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- 항상 고정이라고 생각 해 두면 편함.

4. 인터페이스 기초



4.3 그리기 함수

4.3.2 글자 쓰기

```
olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)
pt1, pt2 = (50, 230), (50, 310)

image = np.zeros((350, 350, 3), np.uint8)
image.fill(255)

cv2.putText(image, 'SIMPLEX', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, brown)
cv2.putText(image, 'DUPLEX', (50, 130), cv2.FONT_HERSHEY_DUPLEX, 3, olive)
cv2.putText(image, 'TRIPLEX', pt1, cv2.FONT_HERSHEY_TRIPLEX, 2, violet)

fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC
cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)
```

4. 인터페이스 기초

4.3 그리기 함수

4.3.3 원 그리기

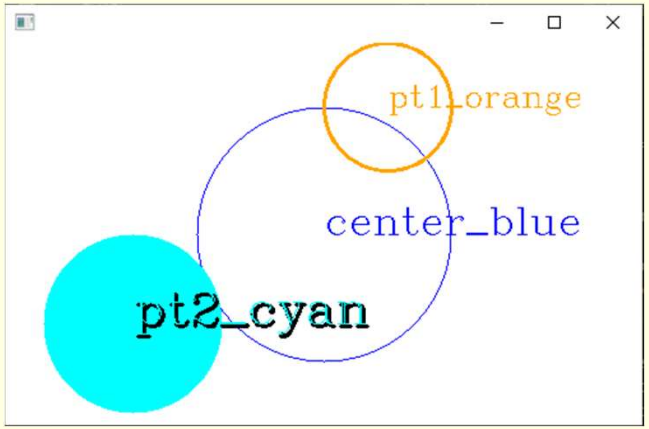
```
orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
white, black = (255, 255, 255), (0, 0, 0)

image = np.full((300, 500, 3), white, np.uint8)

center = (image.shape[1]//2, image.shape[0]//2)
pt1, pt2 = (300, 50), (100, 220)
shade = (pt2[0] + 2, pt2[1] + 2)

cv2.circle(image, center, 100, blue)
cv2.circle(image, pt1, 50, orange, 2)
cv2.circle(image, pt2, 70, cyan, -1)
```

4. 인터페이스 기초

4.3 그리기 함수

4.3.3 원 그리기

```
font = cv2.FONT_HERSHEY_COMPLEX

cv2.putText(image, 'center_blue', center, font, 1.0, blue)
cv2.putText(image, 'pt1_orange', pt1, font, 0.8, orange)
cv2.putText(image, 'pt2_cyan', shade, font, 1.2, black, 2)
# 폰트 두께를 더 두껍게하고, 위치를 약간 옮김으로 그림자 효과 생성
cv2.putText(image, 'pt2_cyan', pt2, font, 1.2, cyan, 1)
```

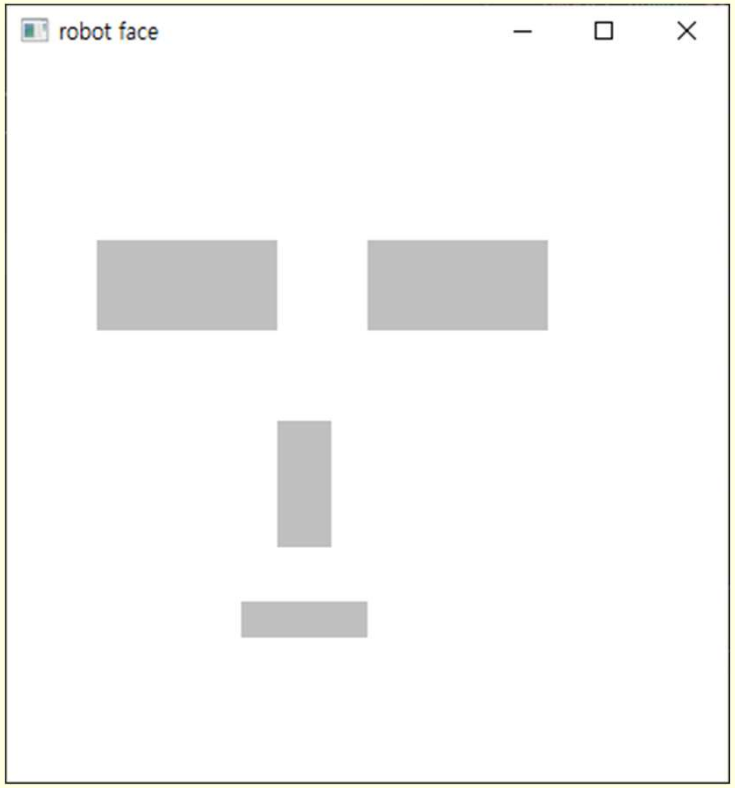


4. 인터페이스 기초

4.3 그리기 함수

◆ 4.3.4 얼굴 그리기

```
image = np.zeros((400, 400), np.uint8)
image.fill(255)

eye = np.full((50, 100), 192, np.uint8)
image[100:150, 50:150] = eye
image[100:150, 200:300] = eye

nouse = np.full((70, 30), 192, np.uint8)
image[200:270, 150:180] = nouse

mouse = np.full((20, 70), 192, np.uint8)
image[300:320, 130:200] = mouse
```



4. 인터페이스 기초

4.3 그리기 함수

4.3.5 마우스로 얼굴 그리기

```
import numpy as np
import cv2

def onMouse():
...

image = np.full((300, 300, 3), (255, 255, 255), np.uint8)
pt = (-1, -1)
title = "Draw Event"

cv2.imshow(title, image)
cv2.setMouseCallback(title, onMouse)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. 인터페이스 기초

4.3 그리기 함수

4.3.5 마우스로 얼굴 그리기(왼쪽 클릭)

```
def onMouse(event, x, y, flags, param):
    global title, pt

    if event == cv2.EVENT_LBUTTONDOWN:
        if pt[0] < 0:
            pt = (x, y)
        else:
            cv2.rectangle(image, pt, (x, y), (255, 0, 0), 2)
            cv2.imshow(title, image)
            pt = (-1, -1)
```

4. 인터페이스 기초

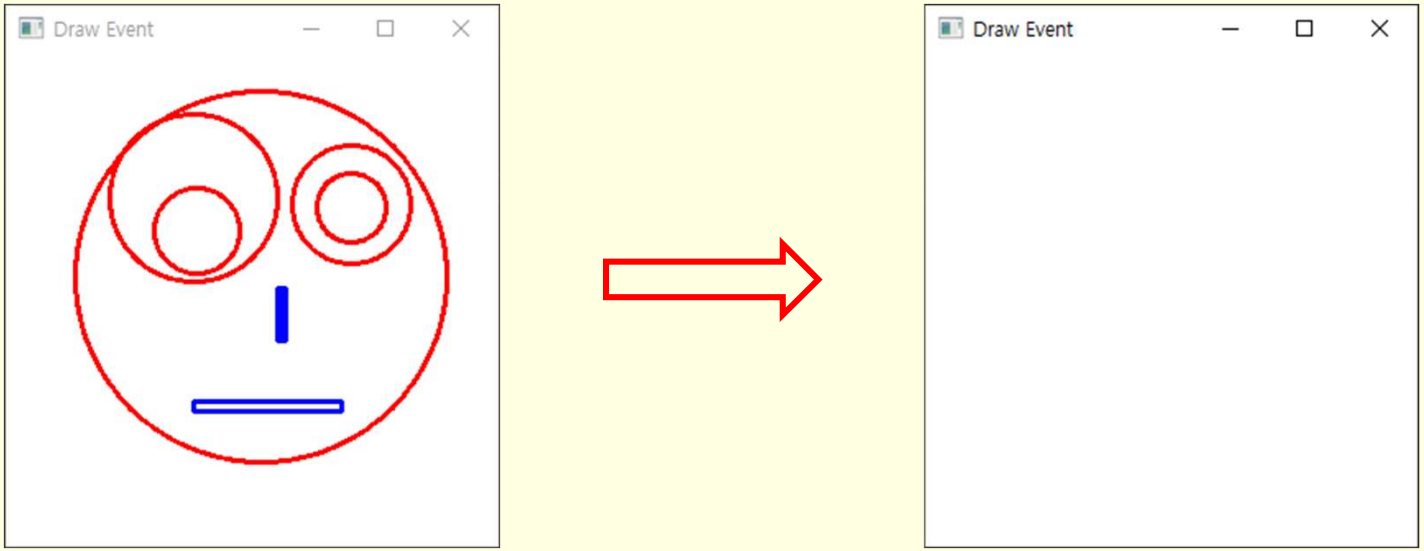
4.3 그리기 함수

4.3.5 마우스로 얼굴 그리기(오른쪽 클릭)


```
def onMouse(event, x, y, flags, param):

    ...

    elif event == cv2.EVENT_RBUTTONDOWN:
        if pt[0] < 0:
            pt = (x, y)
        else:
            dx, dy = pt[0] - x, pt[1] - y
            radius = int(np.sqrt(dx*dx + dy*dy))
            cv2.circle(image, pt, radius, (0, 0, 255), 2)
            cv2.imshow(title, image)
            pt = (-1, -1)
```

4. 인터페이스 기초

- 4.3 그리기 함수
  - 4.3.5 마우스로 얼굴 그리기
  - 추가 조건 (제한시간 5분)
    - 마우스 휠 을 조작 했을 때, 초기화 (지우개 역할)



4. 인터페이스 기초

■ 4.3 그리기 함수

◆ 4.3.5 마우스로 얼굴 그리기

```
def onMouse(event, x, y, flags, param):

    ...

    elif event == cv2.EVENT_MOUSEWHEEL:
```

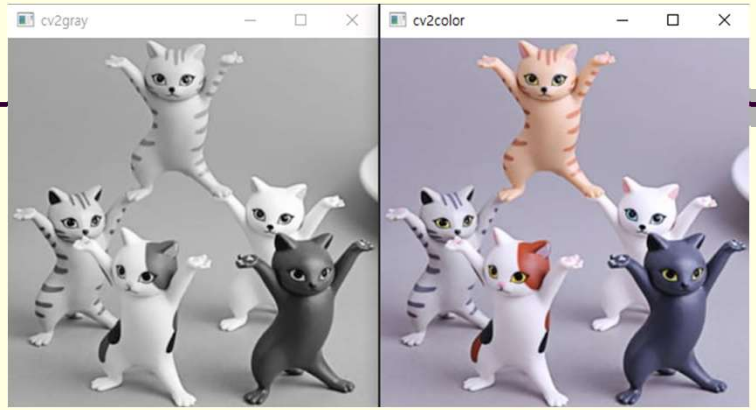
4. 인터페이스 기초

4.4 영상파일 처리

4.4.1 영상파일 읽기

```
import cv2

title1, title2 = 'cv2gray', 'cv2color'
cv2gray = cv2.imread('/images/test_image.jpg', cv2.IMREAD_GRAYSCALE)
cv2color = cv2.imread('/images/test_image.jpg', cv2.IMREAD_COLOR)

cv2.imshow(title1, cv2gray)
cv2.imshow(title2, cv2color)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. 인터페이스 기초



4.4 영상파일 처리

4.4.1 영상파일 읽기

```
import cv2

image_file_path = 'D:/github/OpenCV-Python/2022-10-04/images/test_image.jpg'

title1, title2 = 'cv2gray', 'cv2color'
cv2gray = cv2.imread(image_file_path, cv2.IMREAD_GRAYSCALE)
cv2color = cv2.imread(image_file_path, cv2.IMREAD_COLOR)

cv2.imshow(title1, cv2gray)
cv2.imshow(title2, cv2color)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

상대경로가 아닌, 절대경로를 넣어야 함

4. 인터페이스 기초

4.4 영상파일 처리

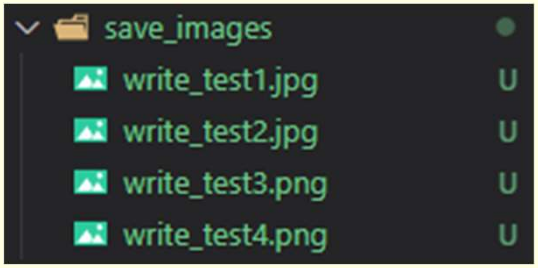
4.4.2 영상파일 저장

```
import cv2

image_file_path = 'D:/github/OpenCV-Python/2022-10-04/images/test_image.jpg'
save_file_path = 'D:/github/OpenCV-Python/2022-10-04/save_images/'

image = cv2.imread(image_file_path, cv2.IMREAD_COLOR)

params_jpg = (cv2.IMWRITE_JPEG_QUALITY, 10)     # JPEG 화질 설정
params_png = (cv2.IMWRITE_PNG_COMPRESSION, 9)   # PNG 압축율 설정
```

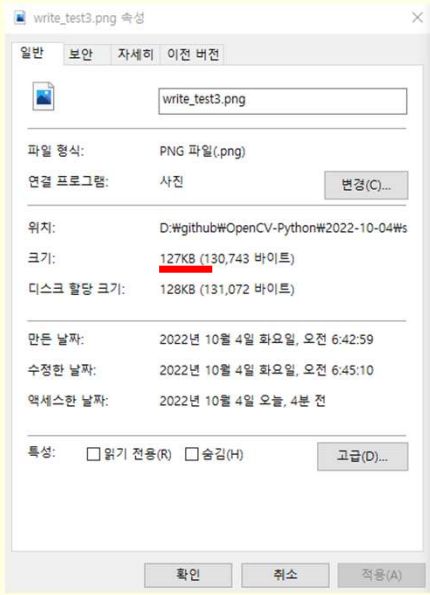
4. 인터페이스 기초

4.4 영상파일 처리

4.4.2 영상파일 저장



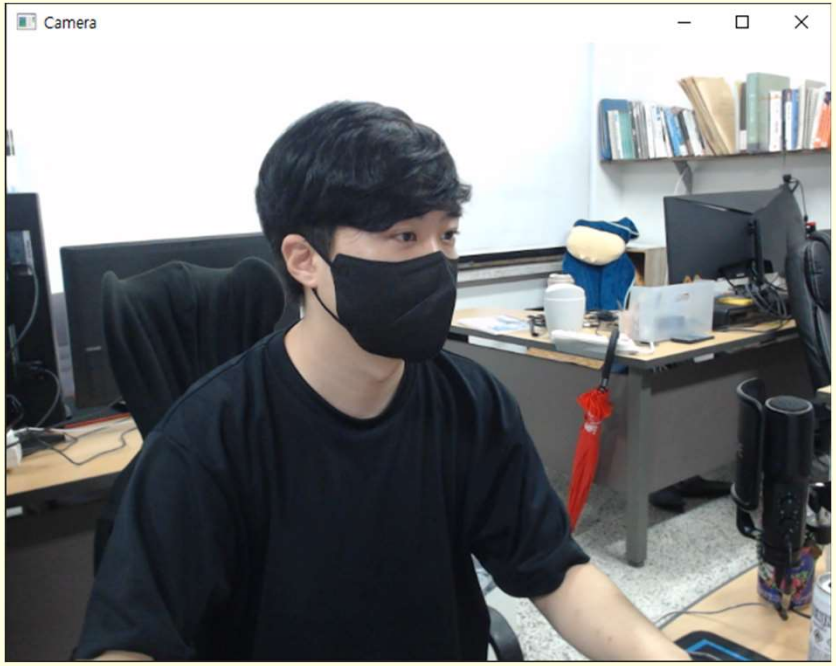

```
cv2.imwrite(f'{save_file_path}write_test1.jpg', image)
cv2.imwrite(f'{save_file_path}write_test2.jpg', image, params_jpg)
cv2.imwrite(f'{save_file_path}write_test3.png', image)
cv2.imwrite(f'{save_file_path}write_test4.png', image, params_png)

print('저장완료')
```



4. 인터페이스 기초

4.5 비디오 처리

4.5.1 카메라에서 프레임 읽기




```
import cv2

capture = cv2.VideoCapture(0)
while True:
    ret, frame = capture.read()
    if cv2.waitKey(30) >= 0:
        break # 스페이스바로 종료

    exposure = capture.get(cv2.CAP_PROP_EXPOSURE) # 노출 속성
    title = 'Camera'
    cv2.imshow(title, frame)
capture.release()
```

4. 인터페이스 기초

4.5 비디오 처리

4.5.2 카메라 프레임을 동영상 파일로 저장

```
import cv2

capture = cv2.VideoCapture(0)

fps = 29.97
delay = round(1000/fps)
size = (640, 360)
fourcc = cv2.VideoWriter_fourcc(*'DX50’)

## 카메라 속성을 실행창에 출력
print('width x height: ',size)
print('VideoWriterfourcc: ', fourcc)
print(f'delay: {delay}')
print('fps: ',fps)
```

4. 인터페이스 기초

4.5 비디오 처리

4.5.2 카메라 프레임을 동영상 파일로 저장

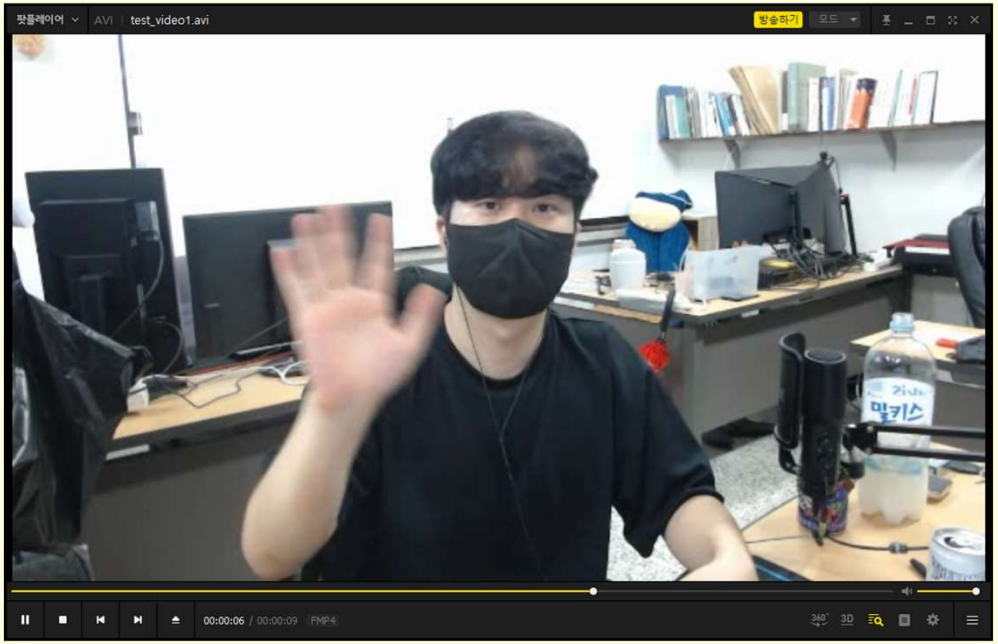
```
## 카메라 속성 지정
capture.set(cv2.CAP_PROP_ZOOM, 1)
capture.set(cv2.CAP_PROP_FOCUS, 0)
capture.set(cv2.CAP_PROP_FRAME_WIDTH, size[0])
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, size[1])


## 동영상파일 개방 및 코덱, 해상도 설정
writer = cv2.VideoWriter('D:/github/OpenCV-Python/2022-10-
04/save_videos/test_video1.avi', fourcc, fps, size)
```

4. 인터페이스 기초

4.5 비디오 처리

◆ 4.5.2 카메라 프레임을 동영상 파일로 저장

```
while True:
    ret, frame = capture.read()
    if cv2.waitKey(30) >= 0:
        break # 스페이스바로 종료

    writer.write(frame)

    title = 'Camera'
    cv2.imshow(title, frame)

writer.release()  # 쓰기 종료
capture.release() # 카메라 종료
```



4. 인터페이스 기초

4.5 비디오 처리

4.5.3 동영상 파일 읽기

```
import cv2

capture = cv2.VideoCapture('D:/github/OpenCV-Python/2022-10-
04/save_videos/test_video1.avi')

fps = capture.get(cv2.CAP_PROP_FPS)
delay = int(1000/fps)
fps_cnt = 0 # 현재 프레임 번호
```

4. 인터페이스 기초

4.5 비디오 처리

4.5.3 동영상 파일 읽기

```
while True:
    ret, frame = capture.read()
    if cv2.waitKey(30) >= 0:
        break # 스페이스바로 종료

    fps_cnt += 1
    print(fps_cnt)

    title = 'Camera'
    cv2.imshow(title, frame)

capture.release()
```