2장. 기본 입출력

세부목차

Image and Video I/O

- **Image show**

```
import cv2


img_file =”../img/model.jpg"
img = cv2.imread(img_file)


if img is not None:
    cv2.imshow('model', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
     print(“Noe Image file”)
```

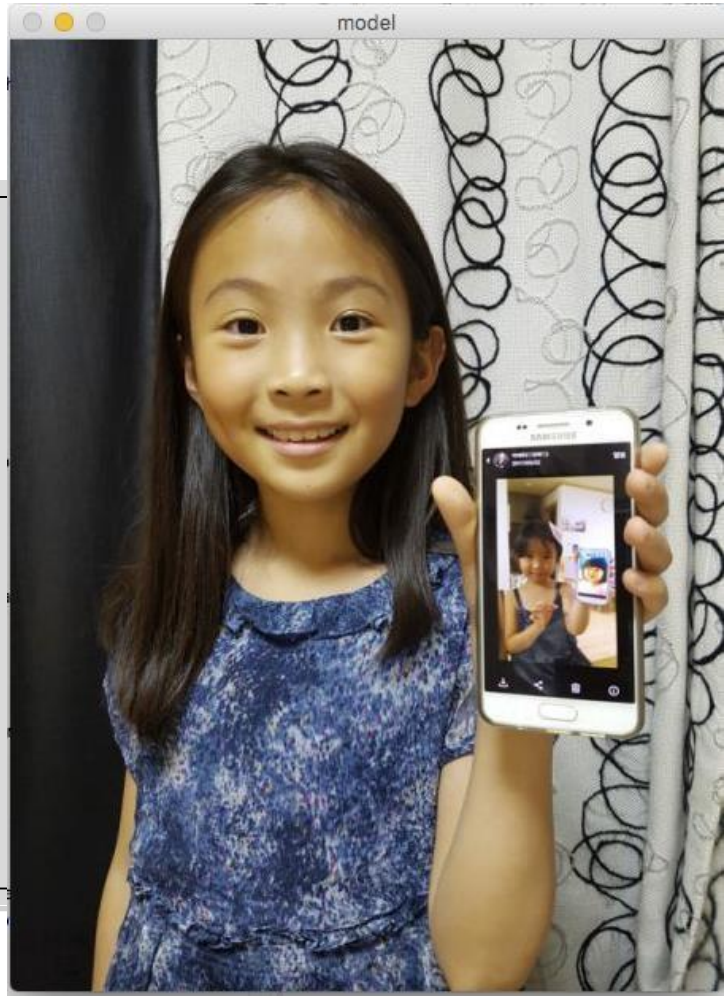


Image and Video I/O

❖ Image show

- img = cv2.imread(file_name [,mod_flag]) : 파일로 부터 이미지 읽기
  - file_name : 이미지 경로, 문자열
  - mode_flag : 읽기 모드 지정 플래그
    - cv2.IMREAD_COLOR : 컬러(BGR) 스케일, 기본 값, 1
    - cv2.IMREAD_UNCHANGED : 파일에 저장된 스케일, -1
    - cv2.IMREAD_GRAYSCALE : 그레일 스케일, 0
  - img : 읽은 이미지, NumPy ndarray
- cv2.imshow(title, img) : 이미지 화면에 표시
  - title : 창에 표시할 이름, 문자열
  - img : 표시할 이미지, NumPy ndarray
- key = cv2.waitKey([delay]) : 키보드 입력 대기
  - delay : 대기 시간, ms
  - key : 입력한 키보드 값, -1: timeout
- cv2.destroyAllWindows() : 모든 열린 창 닫기

Image and Video I/O

- **Read in GrayScale**

```
import cv2

img_file = "../img/model.jpg"
img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)

if not img is None:
    cv2.imshow(img_file, img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("no file:", img_file)
```

Image and Video I/O

❖ Image Write

- cv2.imwrite(file_hame, img)
  - file_name : 저장할 파일 이름, 문자열
  - img : 저장할 이미지(NumPy array)

```
import cv2

img_file =”../img/model.jpg"
img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
cv2.imshow(" img_file ", img)

print('saving file.')
cv2.imwrite(”./gray.jpg", img)
Cv2.waitKey()
cv2.destroyAllWindows()
```

Image and Video I/O

❖ Video Capture

- cap = cv2.VideoCapture(file_path 또는 index) : 비디오 캡쳐 객체 생성자
  - file_path : 동영상 파일 경로, str
  - index : 카메라 장치 번호, int, 0부터 순차적으로 증가(예:0,1,2,....)
- ret = cap.isOpened() : 연결 초기화 확인
  - ret : 성공 여부, Boolean
- cap.set()/get() : Property 변경 및 확인
  - cv2.CAP_PROP_FRAME_WIDTH(3), cv2.CAP_PROP_FRAME_HEIGHT(4)
  - cv2.CAP_PROP_FPS : 초당 프레임 수
  - cv2.CAP_PROP_POS_MSEC : 동영상 파일의 프레임 위치(ms)
  - cv2.CAP_PROP_POS_AVI_RATIO : 동영상 파일의 상대 위치 (0-시작, 1-끝)
  - cv2.CAP_PROP_FOURCC : 동영상 파일 코덱 문자
  - cv2.CAP_PROP_AUTOFOCUS : 카메라 자동 초점 조절
  - cv2.CAP_PROP_ZOOM : 카메라 줌
- ret, img = cap.read() : 다음 프레임 읽기
  - ret : 성공 여부, boolean
  - img : 영상 프레임 데이타(numpy array)

Image and Video I/O

❖ Video File Play

```
import cv2
video_file = "../img/big_buck.avi" # 동영상 파일 경로
cap = cv2.VideoCapture(video_file) # 동영상 캡쳐 객체 생성 ---①
if cap.isOpened(): # 캡쳐 객체 초기화 확인
    while True:
        ret, img = cap.read() # 다음 프레임 읽기 --- ②
        if ret: # 프레임 읽기 정상
            cv2.imshow(video_file, img) # 화면에 표시 --- ③
            cv2.waitKey(25) # 25ms 지연(40fps로 가정) --- ④
        else: # 다음 프레임 읽을 수 없슴,
            break # 재생 완료
else:
    print("can't open video.") # 캡쳐 객체 초기화 실패
cap.release() # 캡쳐 자원 반납
cv2.destroyAllWindows()
```

Image and Video I/O

❖ Video File Play with FPS

- delay = 1000/FPS

```
import cv2
video_file = "../img/big_buck.avi“          # 동영상 파일 경로
cap = cv2.VideoCapture(video_file)          # 동영상 캡쳐 객체 생성
if cap.isOpened(): # 캡쳐 객체 초기화 확인
    fps = cap.get(cv2.CAP_PROP_FPS) # 프레임 수 구하기
    delay = int(1000/fps)
    print("FPS: %f, Delay: %dms" %(fps, delay))
    while True:
        ret, img = cap.read() # 다음 프레임 읽기
        if ret: # 프레임 읽기 정상
            cv2.imshow(video_file, img) # 화면에 표시
            cv2.waitKey(delay) # fps에 맞게 시간 지연
        else:
            break # 다음 프레임 읽을 수 없슴, 재생 완료
```

Image and Video I/O

❖ Video File Play with FPS

- delay = 1000/FPS

```
else:
        print("can't open video.") # 캡쳐 객체 초기화 실패
cap.release() # 캡쳐 자원 반납
cv2.destroyAllWindows()
```

Image and Video I/O

❖ Camera Frame

```
import cv2

cap = cv2.VideoCapture(0) #0 or -1
while cap.isOpened():
    ret, img = cap.read()
    if ret:
        cv2.imshow('camera-0', img)
        if cv2.waitKey(1) &0xFF == 27: #esc
            break
    else:
        print('no camera!')
        break

cap.release()
cv2.destroyAllWindows()
```

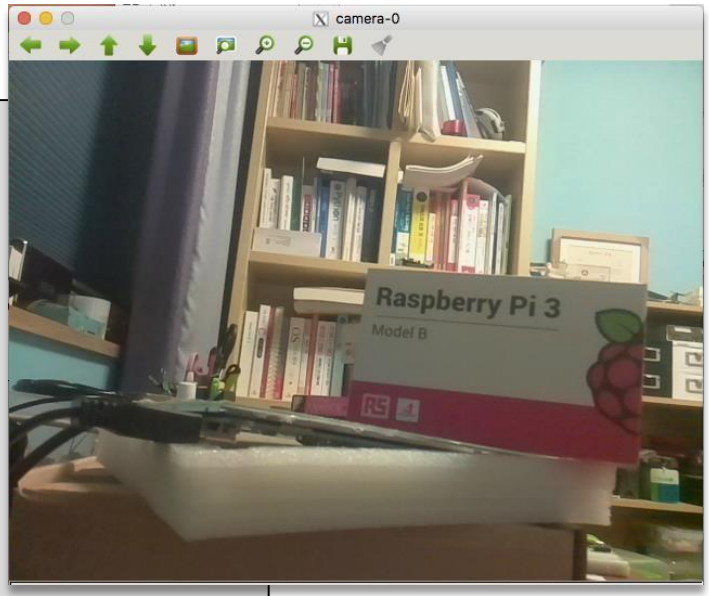


Image and Video I/O

❖ Camera Frame Resize

```
import cv2

cap = cv2.VideoCapture(0) #0 or -1
print("width: %d, height:%d" %(cap.get(3), cap.get(4)) )
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
print("resized width: %d, height:%d" %(cap.get(3), cap.get(4)) )
while cap.isOpened(): ret
    , img = cap.read() if
    ret:
        cv2.imshow('camera-0', img)
        if cv2.waitKey(1) &0xFF == 27: #esc
            break
    else:
        print('no camera!')
        break cap.rele
ase() cv2.destroyAllWi
ndows()
```

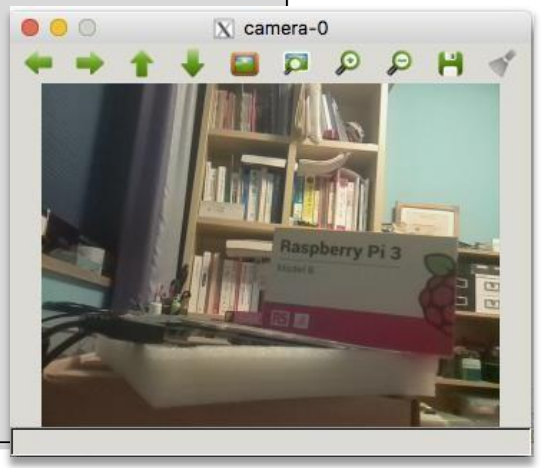


Image and Video I/O

❖ Save a Camera Fra

```
import cv2

cap = cv2.VideoCapture(0)                          #0번 카메라 연결
if cap.isOpened() :
    while True:                                    # 카메라 프레임 읽기
        ret, frame = cap.read()
        if ret:                                    # 프레임 화면에 표시
            cv2.imshow('camera',frame)             #아무 키나 누르면
            if cv2.waitKey(1) != -1 :
                cv2.imwrite('photo.jpg',           frame) #프레임을 'photo.jpg'에 저장
                break
        else:
            print('no frame!')
            break
else:
    print('no camera!')
cap.release()
cv2.destroyAllWindows()
```

Image and Video I/O

❖ Video Recording with Camera

- writer = cv2.VideoWriter(file_path, fourcc, fps, (width, height))
  - file_path : 저장할 경로
  - fourcc : 저장 형식, 4글자
    - cv2.VideoWriter_fourcc(*'ABCD')
      - ABCD : 'MJPG', 'DMX'
    - ord('D')+ (ord('I')<<8) + (ord('V')<<16) + (ord('X')<<24)
  - fps : Frame per Sec, float
  - (width, height) : 프레임 폭, 높이, turple
- writer.write(frame)
  - frame : numpy array
- writer.set()/get()
  - Property 변경 및 확인
  - cv2.CAP_PROP_*
    - FRAME_WIDTH(3),
    - FRAME_HEIGHT(4)

Image and Video I/O

❖ Video Recording with Camera

```
import cv2
cap = cv2.VideoCapture(0)                                    # 0번 카메라 연결
if cap.isOpened:
        file_path = './record.avi' # 저장할 파일 경로 이름 ---①
        fps = 30.0 # FPS, 초당 프레임 수
        fourcc = cv2.VideoWriter_fourcc(*'DIVX') # 인코딩 포맷 문자
        width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
        height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
        size = (int(width), int(height)) # 프레임 크기
        out = cv2.VideoWriter(file_path, fourcc, fps, size) # VideoWriter 객체 생성
        while True:
            ret, frame = cap.read()
            if ret:
                    cv2.imshow('camera-recording',frame)
                    out.write(frame) # 파일 저장
```

Image and Video I/O

❖ Video Recording with Camera

```
                    if cv2.waitKey(int(1000/fps)) != -1:
                        break
                else:
                    print("no frame!")
                    break
        out.release() # 파일 닫기
else:
    print("can't open camera!")
cap.release()
cv2.destroyAllWindows()
```

세부목차

1. Image and Video I/O
2. **Drawing**
3. Window Management
4. Event Handling

Drawing

❖ Line

- cv2.line(img, start, end, color [, thickness, lineType]): 직선을 그리기
  - img : 그림 그릴 대상 이미지, Numpy array
  - start : 선 시작 지점 좌표, 튜플(x,y)
  - end : 선 끝 지점 좌표, 튜플(x, y)
  - color : 선 색상, 튜플(Blue, Green, Red), 0~255
  - thickness : 선 두께
  - lineType : 선 그리기 형식,
    - cv2.LINE_4 : 4 연결 선 알고리즘
    - cv2.LINE_8 : 8 연결 선 알고리즘
    - cv2.LINE_AA : 안티에일리어싱(Antialiasing,계단현상없는 선)

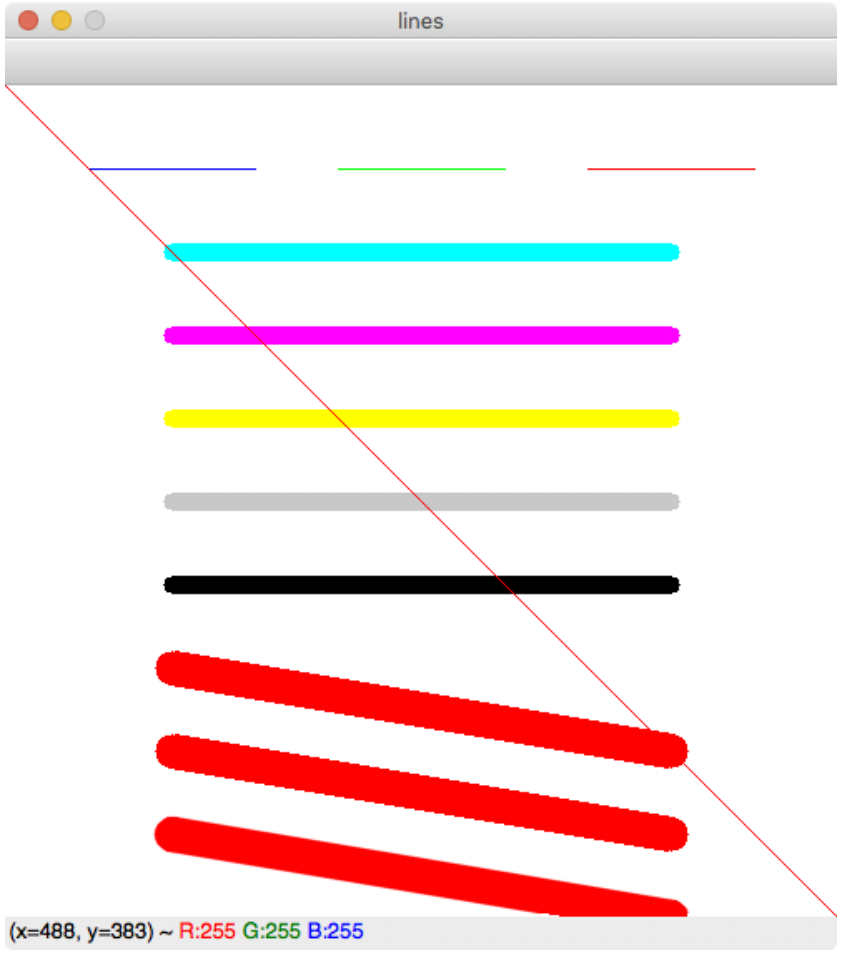
Drawing

❖ Line

```
import cv2

img = cv2.imread('./img/blank_500.jpg')

cv2.line(img, (50, 50), (150, 50), (255,0,0))                    # 파란색 1픽셀 선
cv2.line(img, (200, 50), (300, 50), (0,255,0)) # 초록색 1픽셀 선
cv2.line(img, (350, 50), (450, 50), (0,0,255)) # 빨간색 1픽셀 선
cv2.line(img, (100, 100), (400, 100), (255,255,0), 10)
cv2.line(img, (100, 150), (400, 150), (255,0,255), 10)
cv2.line(img, (100, 200), (400, 200), (0,255,255), 10)
cv2.line(img, (100, 250), (400, 250), (200,200,200), 10)
cv2.line(img, (100, 300), (400, 300), (0,0,0), 10)
cv2.line(img, (100, 350), (400, 350), (0,0,255), 20, cv2.LINE_4 )
cv2.line(img, (100, 400), (400, 400), (0,0,255), 20, cv2.LINE_8)
cv2.line(img, (100, 450), (400, 450), (0,0,255), 20, cv2.LINE_AA)
cv2.line(img, (0,0), (500,500), (0,0,255))              #이미지 전체에 대각
                                                        선

cv2.imshow('lines', img)
cv2.waitKey(0) cv2.destr
oyAllWindows()
```

Drawing

- **Line**



Drawing

❖ Rectangle

- cv2.rectangle(img, start, end, color[, thickness, lineType]) : 사각형 그리기
  - img : 그림 그릴 대상 이미지
  - start : 사각형 시작 꼭짓점, 튜플(x, y)
  - end : 사각형 끝 꼭짓점, 튜플(x, y)
  - color : 색상, 튜플(Blue, Green, Red)
  - thickness : 선 두께, -1: 채우기
  - lineType : 선 타입,
    - cv2.LINE_4
    - cv2.LINE_8
    - cv2.LINE_AA

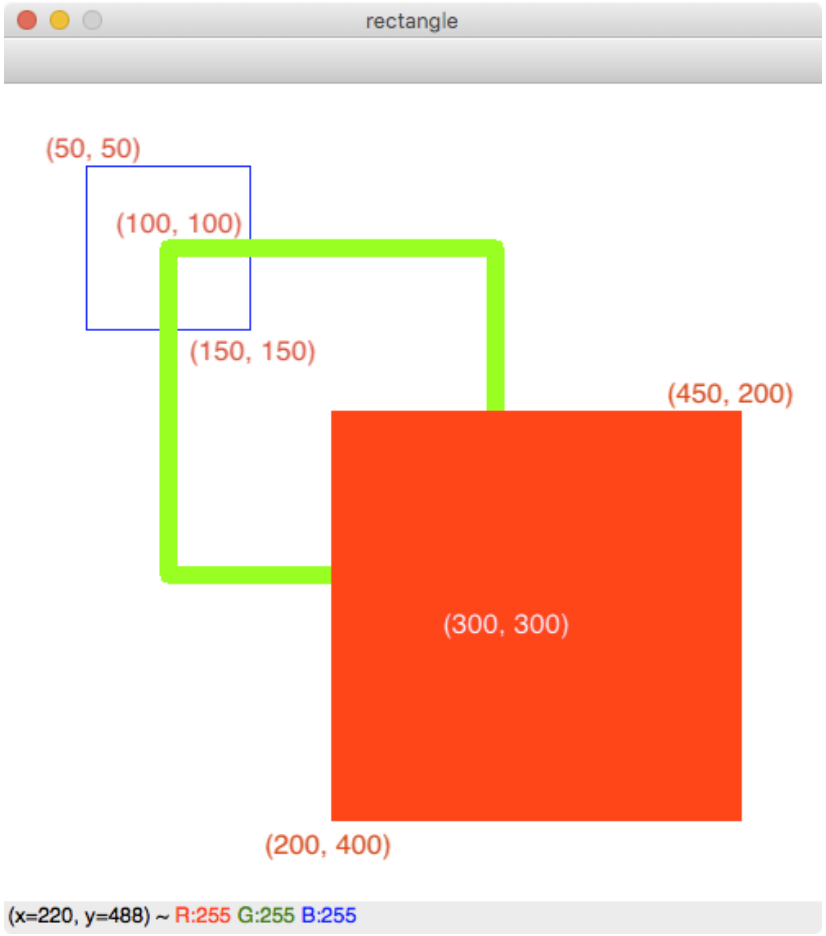
Drawing

- **Rectangl**

```
import cv2

img = cv2.imread('./img/blank_500.jpg')

cv2.rectangle(img, (50, 50), (150, 150), (255,0,0) )
cv2.rectangle(img, (300, 300), (100, 100),(0,255,0), 10 )
cv2.rectangle(img, (450, 200), (200, 450), (0,0,255), -1 )

cv2.imshow('rectangle', img)
cv2.waitKey(0) cv2.destroyAl
lWindows()
```

Drawing

- **Rectangle**



Drawing

❖ Polylines

- cv2.polylines(img, points, isClosed, color[, thickness, lineType]) : 다각형 그리기
  - img : 그림 그릴 대상 이미지
  - points : 꼭짓점 좌표, Numpy array의 리스트
  - isClosed : 닫힌 도형 여부, Boolean
  - color : 색상, 튜플(Blue, Green, Red)
  - thickness : 선 두께
  - lineType : 선 타입
    - cv2.LINE_4
    - cv2.LINE_8
    - cv2.LINE_AA

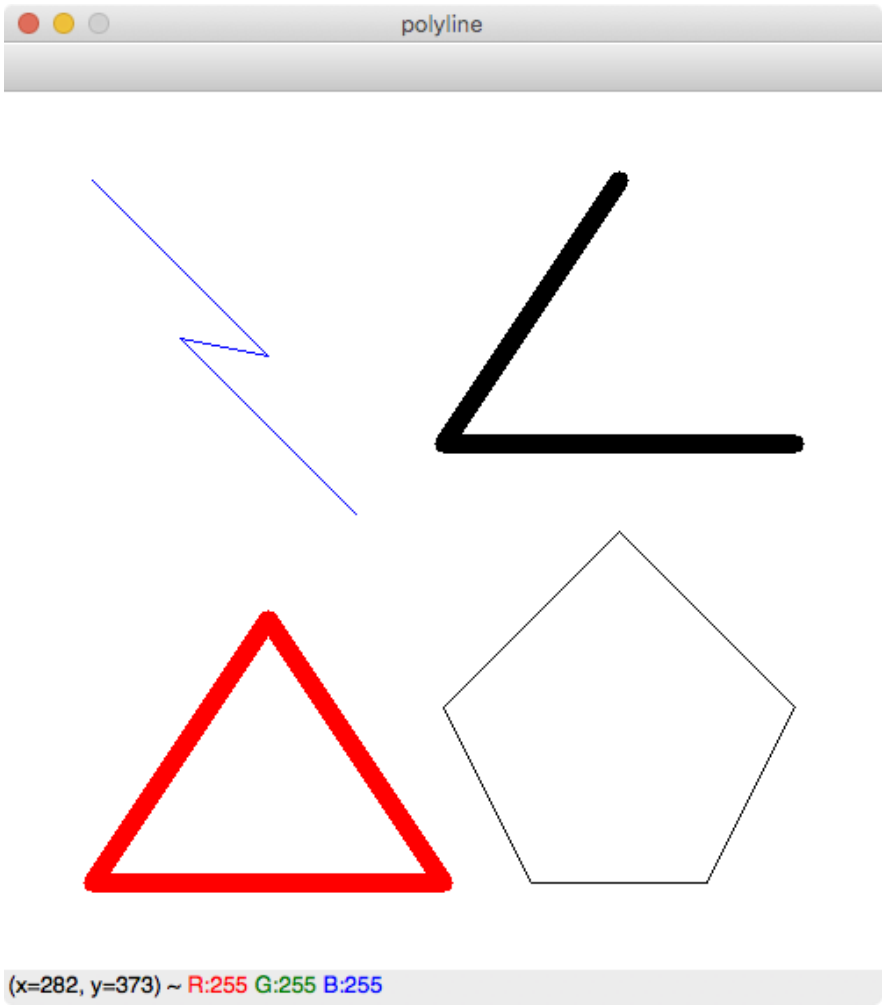
Drawing

- **Polylines**

```
import cv2
import numpy as np

img = cv2.imread('./img/blank_500.jpg')


pts1 = np.array([[50,50], [150,150], [100,140],[200,240]], dtype=np.int32)
pts2 = np.array([[350,50], [250,200], [450,200]], dtype=np.int32)
pts3 = np.array([[150,300], [50,450], [250,450]], dtype=np.int32)
pts4 = np.array([[350,250], [450,350], [400,450],[300,450], [250,350]], \
                                                        dtype=np.int32)

cv2.polylines(img, [pts1], False, (255,0,0))
cv2.polylines(img, [pts2], False, (0,0,0), 10)
cv2.polylines(img, [pts3], True, (0,0,255), 10)
cv2.polylines(img, [pts4], True, (0,0,0))

cv2.imshow('polyline', img)
cv2.waitKey(0) cv2.destroy
AllWindows()
```

Drawing

- **Polylines**



Drawing

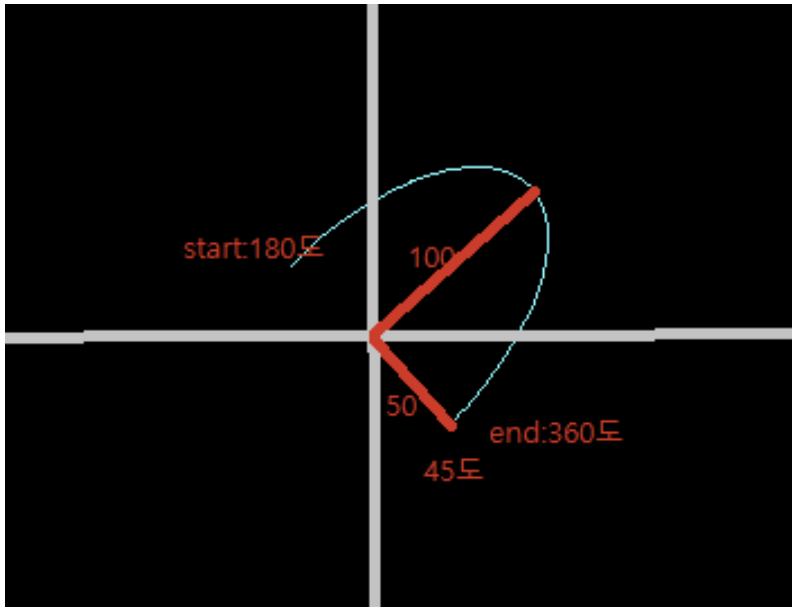
❖ Circle, Ellipse

- cv2.circle(img, center, radius, color [, thickness, lineType]) : 원 그리기 함수
  - img : 그림 그림 대상 이미지
  - center : 원점, 튜플(x, y)
  - radius : 반지름
  - color : 색상, 튜플(Blue, Green, Red)
  - thickness : 선 두께, -1: 채우기
  - lineType : 선 타입, cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA
- cv2.ellipse(img, center, axes, angle, from, to, color[, thickness, lineType])
  - img : 그림 그림 대상 이미지
  - center : 원점, 튜플(x, y)
  - axes : 기준 축 길이
  - angle : 기준 축 회전 각도
  - from, to : 선을 그릴 시작 지점과 끝 지점 각도
  - color : 색상, 튜플(Blue, Green, Red)
  - thickness : 선 두께, -1: 채우기
  - lineType : 선 타입, cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA

❖ Ellipse Example

- cv2.ellipse(img, center, axes, angle, from, to, color[, thickness, lineType])

```
ellipse(img, (256,256), (50,100), 45, 180,360, 1)
```



Drawing

❖ Circle, Ellipse

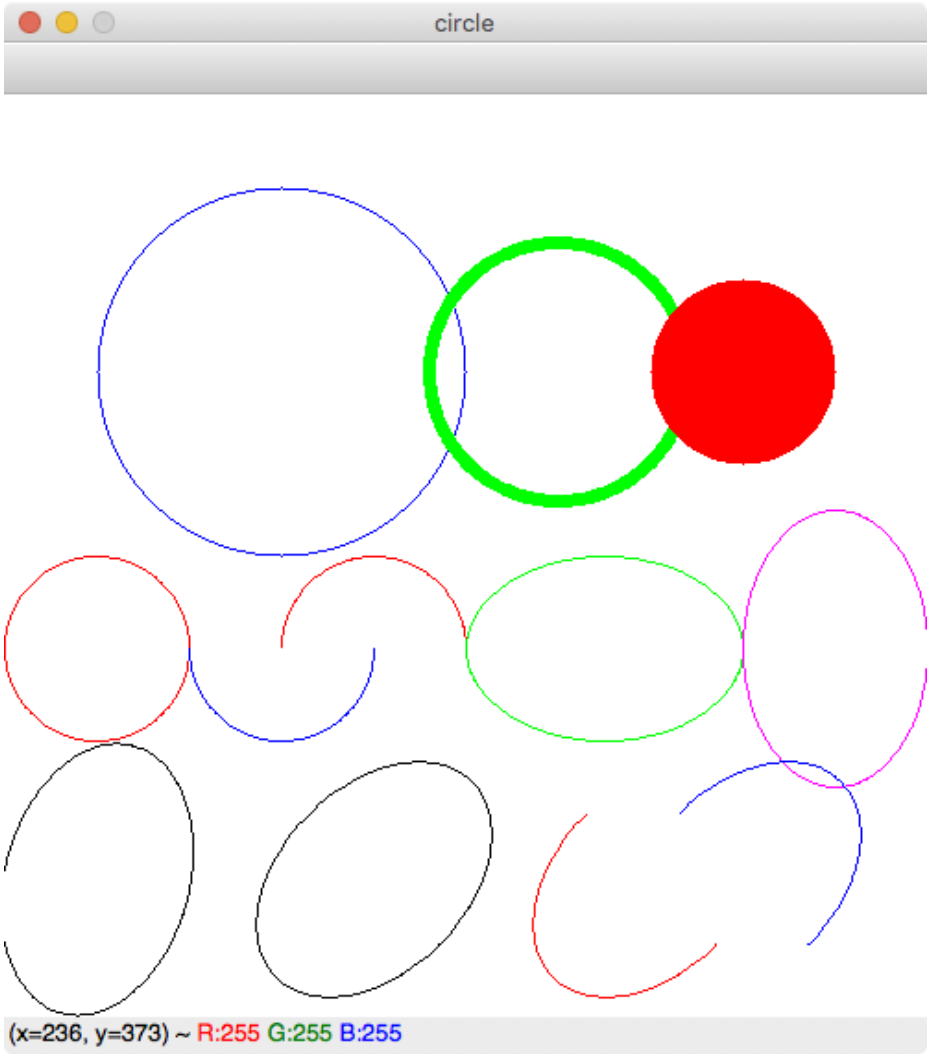
```
import cv2
img = cv2.imread('../img/blank_500.jpg')
# 원점(150,150), 반지름 100 ---①
cv2.circle(img, (150, 150), 100, (255,0,0))
# 원점(300,150), 반지름 70 ---②
cv2.circle(img, (300, 150), 70, (0,255,0), 5)
# 원점(400,150), 반지름 50, 채우기 ---③
cv2.circle(img, (400, 150), 50, (0,0,255), -1)
# 원점(50,300), 반지름(50), 회전 0, 0도 부터 360도 그리기 ---④
cv2.ellipse(img, (50, 300), (50, 50), 0, 0, 360, (0,0,255))
# 원점(150, 300), 아래 반원 그리기 ---⑤
cv2.ellipse(img, (150, 300), (50, 50), 0, 0, 180, (255,0,0))
#원점(200, 300), 윗 반원 그리기 ---⑥
cv2.ellipse(img, (200, 300), (50, 50), 0, 181, 360, (0,0,255))
# 원점(325, 300), 반지름(75,50) 납작한 타원 그리기 ---⑦
cv2.ellipse(img, (325, 300), (75, 50), 0, 0, 360, (0,255,0))
```

❖ Circle, Ellipse

```
# 원점(450,300), 반지름(50,75) 홀쭉한 타원 그리기 ---⑧
cv2.ellipse(img, (450, 300), (50, 75), 0, 0, 360, (255,0,255))
# 원점(50, 425), 반지름(50,75), 회전 15도 ---⑨
cv2.ellipse(img, (50, 425), (50, 75), 15, 0, 360, (0,0,0))
# 원점(200,425), 반지름(50,75), 회전 45도 ---⑩
cv2.ellipse(img, (200, 425), (50, 75), 45, 0, 360, (0,0,0))
# 원점(350,425), 홀쭉한 타원 45도 회전 후 아랫 반원 그리기 ---⑪
cv2.ellipse(img, (350, 425), (50, 75), 45, 0, 180, (0,0,255))
# 원점(400,425), 홀쭉한 타원 45도 회전 후 윗 반원 그리기 ---⑫
cv2.ellipse(img, (400, 425), (50, 75), 45, 181, 360, (255,0,0))
cv2.imshow('circle', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Drawing

- **Circle, Ellipse**



Drawing

❖ Text

- cv2.putText(img, text, point, fontFace, fontSize, color [, thickness, line Type])
  - img : 글씨를 표시할 이미지
  - text : 표시할 문자열
  - point : 글씨 표시할 좌표(좌측 하단 기준점), 튜플(x,y),
  - fontFace : 글꼴
    - cv2.FONT_HERSHEY_PLAIN : 산세리프 작은 글꼴
    - cv2.FONT_HERSHEY_SIMPLEX : 산세리프 일반 글꼴
    - cv2.FONT_HERSHEY_DUPLEX : 산세리프 진한 글꼴
    - cv2.FONT_HERSHEY_COMPLEX_SMALL : 세리프 작은 글꼴
    - cv2.FONT_HERSHEY_COMPLEX : 세리프 일반 글꼴
    - cv2.FONT_HERSHEY_TRIPLEX : 세리프 진한 글꼴
    - cv2.FONT_HERSHEY_SCRIPT_SIMPLEX : 필기체 산세리프 글꼴
    - cv2.FONT_HERSHEY_SCRIPT_COMPLEX : 필기체 세리프 글꼴
    - cv2.FONT_ITALIC : 이탤릭체 플래그
  - fontSize : 글꼴 크기
  - color : 색상, 튜플(Blue, Green, Red)
  - thickness : 선 두께, -1: 채우기
  - lineType : 선 타입, cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA

Drawing

❖ Text (1/2)

```
import cv2

img = cv2.imread('./img/blank_500.jpg')

cv2.putText(img, "Plain", (50, 30), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 0))
cv2.putText(img, "Simplex", (50, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0))
cv2.putText(img, "Duplex", (50, 110), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0))
cv2.putText(img, "Simplex", (200, 110), cv2.FONT_HERSHEY_SIMPLEX, 2, \
                                                        (0, 0, 250))

cv2.putText(img, "Complex Small", (50, 180), cv2.FONT_HERSHEY_COMPLEX_SMALL,\
                                                        1, (0, 0, 0))
cv2.putText(img, "Complex", (50, 220), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 0))
cv2.putText(img, "Triplex", (50, 260), cv2.FONT_HERSHEY_TRIPLEX, 1, (0, 0, 0))
cv2.putText(img, "Complex", (200, 260), cv2.FONT_HERSHEY_TRIPLEX, 2, \
                                                        (0, 0, 255))
```

Drawing

❖ Text (2/2)

```
cv2.putText(img, "Script  Simplex", (50, 330), \
                              cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, 1, (0, 0,0))
cv2.putText(img, "Script  Complex", (50, 370), \
                              cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 1, (0, 0,0))

cv2.putText(img, "Plain Italic", (50, 430), cv2.FONT_HERSHEY_PLAIN| \
                                                cv2.FONT_ITALIC, 1, (0,  0,0))
cv2.putText(img, "Complex Italic", (50, 470), cv2.FONT_HERSHEY_COMPLEX| \
                                                cv2.FONT_ITALIC, 1, (0,  0,0))

cv2.imshow('draw text', img)
cv2.waitKey()  cv2.destroyAll
Windows()
```

Drawing

- **Text**

세부목차

1. Image and Video I/O
2. Drawing
3. **Window Management**
4. Event Handling

Window Management

❖ Window Management

- cv2.namedWindow(title [, option]) : 이름을 갖는 창 열기
  - title : 창 이름, 제목 줄에 표시
  - option : 창 옵션, "cv2.WINDOW_"로 시작
    - cv2.WINDOW_NORMAL : 임의 크기, 사용자 창 크기 조정 가능
    - cv2.WINDOW_AUTOSIZE : 이미지와 같은 크기, 창 크기 재조정 불가능
- cv2.moveWindow(title, x, y) : 창 위치 이동
  - title : 위치 변경할 대상 창의 이름
  - x, y : 이동 할 창의 위치
- cv2.resizeWindow(title, width, height) : 창 크기 변경
  - title : 창 크기 변경할 창의 이름
  - width, height : 변경할 창의 폭과 높이
- cv2.destroyWindow(title) : 창 닫기
  - title : 닫을 대상 창 이름
- cv2.destroyAllWindows() : 열린 모든 창 닫기

Window Management

❖ Window Management

```
import cv2file_path = '../img/girl.jpg‘

img = cv2.imread(file_path)                          # 이미지를 기본 값으로 읽기

img_gray = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE) # 이미지를 그레이스케일로 읽기

cv2.namedWindow('origin')                            # origin 이름으로 창 생성
cv2.namedWindow('gray', cv2.WINDOW_NORMAL)           # gray 이름으로 창 생성

cv2.imshow('origin', img)                            # origin 창에 이미지 표시
cv2.imshow('gray', img_gray)                         # gray 창에 이미지 표시

cv2.moveWindow('origin', 0, 0)                       # 창 위치 변경
cv2.moveWindow('gray', 100, 100)                     # 창 위치 변경

cv2.waitKey(0)                                       # 아무키나 누르면

cv2.resizeWindow('origin', 200, 200)                 # 창 크기 변경 (변경 안됨)

cv2.resizeWindow('gray', 100, 100)                   # 창 크기 변경 (변경 됨))

cv2.waitKey(0)                                       # 아무키나 누르면
cv2.destroyWindow("gray")                            # gray 창 닫기

cv2.waitKey(0)                                       # 아무키나 누르면

cv2.destroyAllWindows()                              # 모든 창 닫기
```

Window Management

- **Window Management**

세부목차

1. Image and Video I/O
2. Drawing
3. Window Management
4. **Event Handling**

Event Handling

❖ Keyboard Event

- val = cv2.waitKey([delay]) : 키보드 입력 대기, 창이 활성화 된 상태만 동작
  - delay=0 : 대기할 시간, ms
    - delay <= 0 : 무한대기
  - val : code of pressed key
    - -1 : time out
- 키 코드 문자 비교, 파이썬 내장 함수와 함께 사용
  - ord(c) : c 문자에 매핑된 코드 반환
    - ex : ord('a') #97
  - chr(code) : 코드에 매핑된 문자 반환
    - ex : chr(97) # 'a'
  - cv2.waitKey() == ord('a')
- ASCII(8bit) code overflow
  - 몇몇 64bit 환경에서 32비트 정수 반환, 8bit 이상 비트 초기화 필요
  - key = cv2.waitKey(0) & 0xFF

Event Handling

❖ Keyboard Event

```
import cv2
img_file = "../img/girl.jpg"
img = cv2.imread(img_file) title = 'IMG'              # 창 이름
x, y = 100, 100              # 최초 좌표
while True:
      cv2.imshow(title, img)
      cv2.moveWindow(title, x, y)
      key = cv2.waitKey(0) & 0xFF # 키보드 입력을 무한 대기, 8비트 마스크처리
     print(key, chr(key))       # 키보드 입력 값,  문자 값 출력
     if key == ord('h'):        # 'h' 키 이면 좌로 이동
         x -= 10
     elif key == ord('j'):      # 'j' 키 이면 아래로 이동
          y += 10
     elif key == ord('k'):      # 'k' 키 이면 위로 이동
          y -= 10
     elif key == ord('l'):      # 'l' 키 이면 오른쪽으로 이동
          x += 10
```

Event Handling

❖ Keyboard Event

```
        elif key == ord('q') or key == 27: # 'q' 이거나 'esc' 이면 종료
            break
cv2.destroyAllWindows()
```

Event Handling

❖ Mouse Event

- cv2.setMouseCallback(win_name, onMouse [, param]) : onMouse 함수를 등록
  - win_name : 이벤트를 등록할 윈도우 이름
  - onMouse : 이벤트에 동작 할 것을 대비해서 미리 선언해 놓은 콜백 함수 객체
  - param : 필요에 따라 onMouse 함수에 전달할 인자
- MouseCallback(event, x, y, flags, param) : 콜백 함수 선언부
  - event : 마우스 이벤트 종류, cv2.EVENT_ 로 시작하는 상수, 모두 12가지
  - x, y : 마우스 좌표
  - flags : 마우스 동작과 함께 일어난 추가 상태, cv2.EVENT_FLAG_로 시작하는 상수, 모두 6가지
  - param : cv2.setMouseCallback() 함수에서 전달한 인자

```
def onMouse(event, x, y, flags, param):
  #여기에 마우스 이벤트에 맞게 해야할 작업을 작성합니다.
  pass

cv2.setMouseCallback('title', onMouse)
```

Drawing and Mouse-Event

❖ Mouse Event 와 Flags 종류

- cv2.EVENT_*

```
import cv2

events =[i for i in dir(cv2) if 'EVENT_'in i]
print( len(events) ,"events" )
print( events )
```

```
18 events
['EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON', 'E
VENT_FLAG_MBUTTON', 'EVENT_FLAG_RBUTTON', 'EVENT_FLAG_SHIFTKEY', 'E
VENT_LBUTTONDBLCLK', 'EVENT_LBUTTONDOWN', 'EVENT_LBUTTONUP', 'EVENT
_MBUTTONDBLCLK', 'EVENT_MBUTTONDOWN', 'EVENT_MBUTTONUP', 'EVENT_MOU
SEHWHEEL', 'EVENT_MOUSEMOVE', 'EVENT_MOUSEWHEEL', 'EVENT_RBUTTONDB
LCLK', 'EVENT_RBUTTONDOWN', 'EVENT_RBUTTONUP']
```

Drawing and Mouse-Event

❖ Mouse Event 종류

- cv2.EVENT_MOSEMOVE : 마우스 움직임
- cv2.EVENT_LBUTTONDOWN : 왼쪽 버튼 누름
- cv2.EVENT_RBUTTONDOWN : 오른쪽 버튼 누름
- cv2.EVENT_MBUTTONDOWN : 가운데 버튼 누름
- cv2.EVENT_LBUTTONUP : 왼쪽 버튼 뗌
- cv2.EVENT_RBUTTONUP : 오른쪽 버튼 뗌
- cv2.EVENT_MBUTTONUP : 가운데 버튼 뗌
- cv2.EVENT_LBUTTONDBLCLK : 왼쪽 버튼 더블 클릭
- cv2.EVENT_RBUTTONDBLCLK :오른쪽 버튼 더블 클릭
- cv2.EVENT_MBUTTONDBLCLK :가운데 버튼 더블 클릭
- cv2.EVENT_MOUSEWHEEL : 휠 스크롤
- cv2.EVENT_MOUSEHWHEEL : 휠 가로 스크롤

Drawing and Mouse-Event

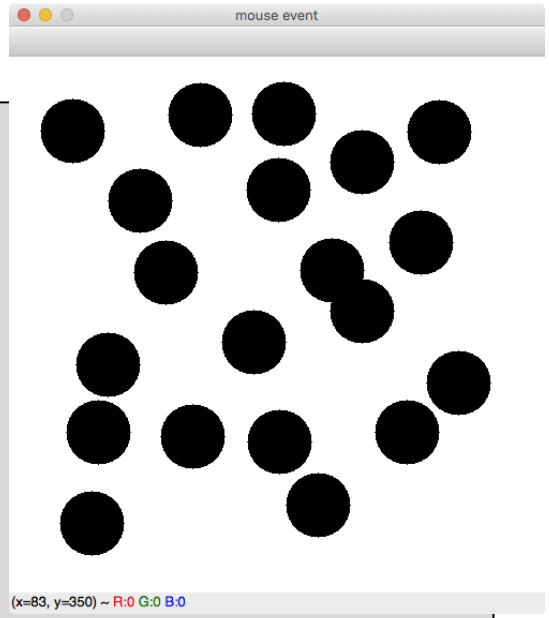
❖ Mouse Event Flag 종류

- cv2.EVENT_FLAG_LBUTTON (1) : 왼쪽 버튼 누름
- cv2.EVENT_FLAG_RBUTTON (2): 오른쪽 버튼 누름
- cv2.EVENT_FLAG_MBUTTON (4): 가운데 버튼 누름
- cv2.EVENT_FLAG_CTRLKEY (8): Ctrl 키 누름
- cv2.EVENT_FLAG_SHIFTKEY (16): Shift 키 누름
- cv2.EVENT_FLAG_ALTKEY (32): Alt 키 누름

Event Handling

❖ Mouse Event

```
import cv2

title ='mouse event'
img=cv2.imread('./img/blank_500.jpg')
cv2.imshow(title, img)

def onMouse(event, x, y, flags, param):
    print(event, x, y, )
    if event == cv2.EVENT_LBUTTONDOWN: cv2.circl
        e(img, (x,y), 30, (0,0,0), -1) cv2.ims
        how(title, img)

cv2.setMouseCallback(title, onMouse)

while True:
    if cv2.waitKey(0) &0xFF == 27: #esc로 종료
        break
cv2.destroyAllWindows()
```



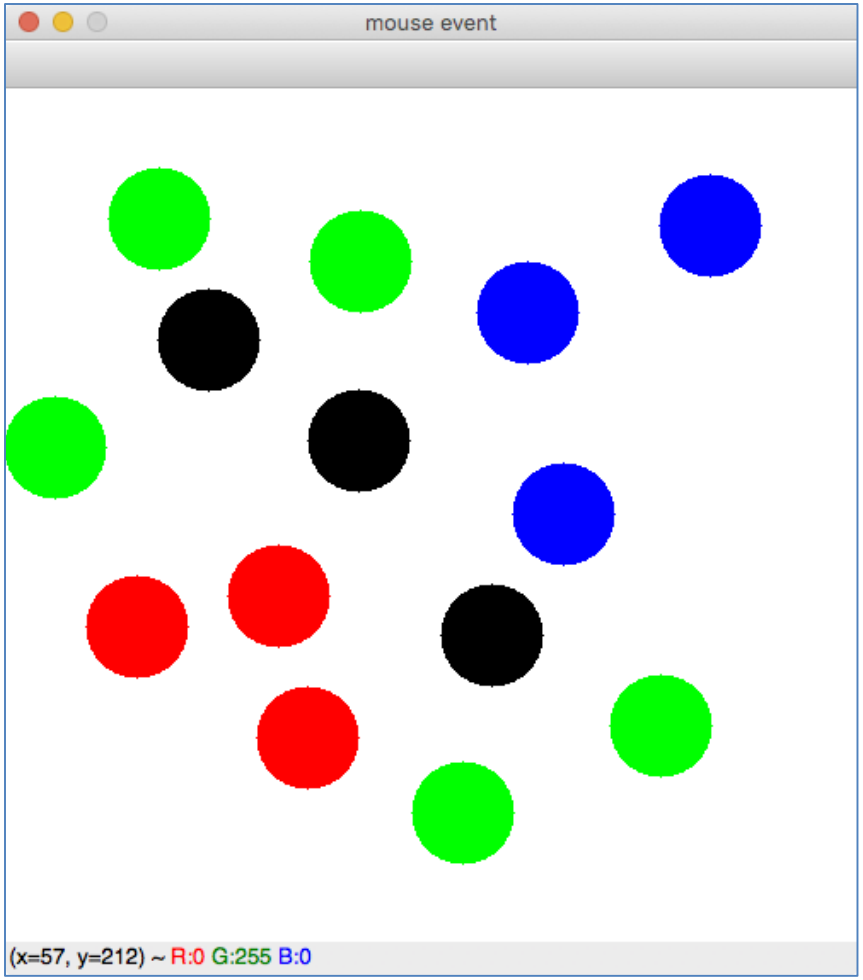
Event Handling

❖ Mouse Event Flag Exa

```
import cv2
title ='mouse event'
img=cv2.imread('./img/blank_500.jpg')
cv2.imshow(title, img)
colors ={'black':(0,0,0),'red':(0,0,255),'blue':(255,0,0), 'green': (0,255,0) }

def onMouse(event, x, y, flags, param):
    print(event, x, y, flags)
    color =colors['black']
    if event == cv2.EVENT_LBUTTONDOWN:
        if flags &cv2.EVENT_FLAG_CTRLKEY and flags &color =colors['green']
            &cv2.EVENT_FLAG_SHIFTKEY:
        elif flags &cv2.EVENT_FLAG_SHIFTKEY :  color =colors['blue']
        elif flags &cv2.EVENT_FLAG_CTRLKEY:
            color =colors['red'] cv2.circle(img, (x,y), 30, color, -1)
        cv2.imshow(title, img)

cv2.setMouseCallback(title, onMouse)
while True:
    if cv2.waitKey(0) &0xFF == 27:
        Break
cv2.destroyAllWindows()
```

Event Handling

❖ Mouse Event Flag Example



Event Handling

❖ Trackbar

- cv2.createTrackbar(name, win_name, value, count, onChange) : 트랙바 생성
  - name : 트랙바 이름
  - win_name : 트랙바를 표시할 창 이름
  - value : 트랙바가 처음 나타날때 갖을 초기 값, 0 ~ count 사이의 값
  - count : 트랙바 눈금의 갯수, 트랙바가 표시할 수 있는 최대 값
  - onChange : TrackbarCallback, 트랙바 이벤트 핸들러 함수
- TrackbarCallback(value) : 트랙바 이벤트 콜백 함수
  - value : 트랙바가 움직인 새 위치 값
- pos = cv2.getTrackbarPos(name, win_name)
  - name : 찾고자 하는 트랙바 이름
  - win_name : 트랙바가 있는 창의 이름
  - pos : 트랙바 위치 값

Event Handling

❖ Trackbar

- 트랙바로 색 조정

```
import cv2 ,  numpy as np

win_name = 'Trackbar'
img = cv2.imread('./img/blank_500.jpg')
cv2.imshow(win_name,img)

def  onChange(x):
     r  =cv2.getTrackbarPos('R',win_name)
     g =cv2.getTrackbarPos('G',win_name)
     b =cv2.getTrackbarPos('B',win_name)
     print(x,  r,  g,  b)
     img[:] =[b,g,r]
     cv2.imshow(win_name,  img)

# 트랙바 생성   --- ⑤
cv2.createTrackbar('R', win_name, 255, 255, onChange)
cv2.createTrackbar('G', win_name, 255, 255, onChange)
cv2.createTrackbar('B', win_name, 255, 255, onChange)
while True:
      if cv2.waitKey(0) & 0xFF == 27:
            Break
cv2.destroyAllWindows()
```

Event Handling

❖ Trackbar

- 트랙바로 색 조정