윈도우 창 제어

함수명과 반환형 및 인수 구조

```
void namedWindow(const string& winname, int flags = WINDOW_AUTOSIZE)
void imshow(const string& winname, InputArray mat)
void destroyWindow(const string& winname)
void destroyAllWindows()
void moveWindow(const string& winname, int x, int y)
void resizeWindow(const string& winname, int width, int height)
```

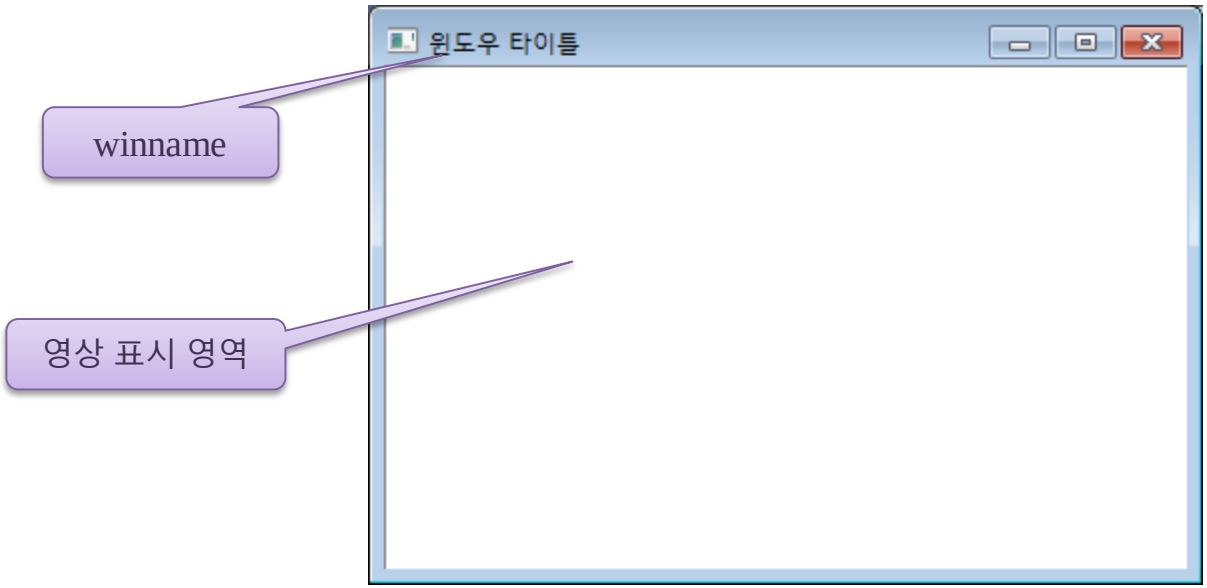
함수 및 인수	설명
`void namedWindow()`	윈도우의 이름(winname)을 설정하고, 해당 이름으로 윈도우를 생성한다.
• `string& winname`	윈도우의 이름, 윈도우의 타이틀 문자열
• `int flags`	윈도우의 크기 조정

옵션	값	설명
WINDOW_NORMAL	0	윈도우 크기의 재조정 가능
WINDOW_AUTOSIZE	1	표지될 행렬의 크기에 맞춰 자동 설정
WINDOW_OPENGL	8	OpenGL을 지원하는 윈도우 생성

`void imshow()`	winname 이름의 윈도우에 mat 행렬을 영상으로 표시한다. 해당이름의 윈도우가 없으면, winname 이름으로 창을 생성하고, 영상을 표시한다.
• `InputArray mat`	윈도우에 표시되는 영상(행렬이 화소값을 밝기로 표시)
`void destroyWindow()`	인수로 지정된 타이틀의 윈도우를 파괴한다.
`void destroyAllWindows()`	HighGUI로 생성된 모든 윈도우를 파괴한다.
`void moveWindow()`	winname 이름의 윈도우를 지정된 위치로(x,y)로 이동시킨다. 이동되는 윈도우의 기준 위치는 좌측 상단이다.
• `int x, int y`	모니터의 이동하려는 위치의 x, y 좌표
`void resizeWindow()`	윈도우의 크기를 재설정한다.
• `int width, int height`	재설정된 윈도우의 가로, 세로 크기

윈도우 창 제어

- 윈도우



윈도우 창 제어

★실습

예제 4.1.1 윈도우(영상 출력창) 이동 |- window_move.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04  int main()
05  {
06      Mat image1(300, 400, CV_8U, Scalar(255));        // 흰색바탕 영상 생성
07      Mat image2(300, 400, CV_8U, Scalar(100));        // 회색바탕 영상 생성
08      string  title1 = "white창 제어";                 // 윈도우 이름
09      string  title2 = "gray 창 제어";

11      namedWindow(title1, WINDOW_AUTOSIZE);            // 윈도우 이름 지정
12      namedWindow(title2, WINDOW_NORMAL);
13      moveWindow(title1, 100, 200);                    // 윈도우 이동
14      moveWindow(title2, 300, 200);

16      imshow(title1, image1);                          // 행렬 원소를 영상으로 표시
17      imshow(title2, image2);
18      waitKey();                                       // 키이벤트 대기
19      destroyAllWindows();                             // 열린 모든 윈도우 제거
20      return 0;
21  }
```

cv::imshow() 함수에 입력되는 행렬의 크기에 맞춰 생성 – 창크기 변경 불가

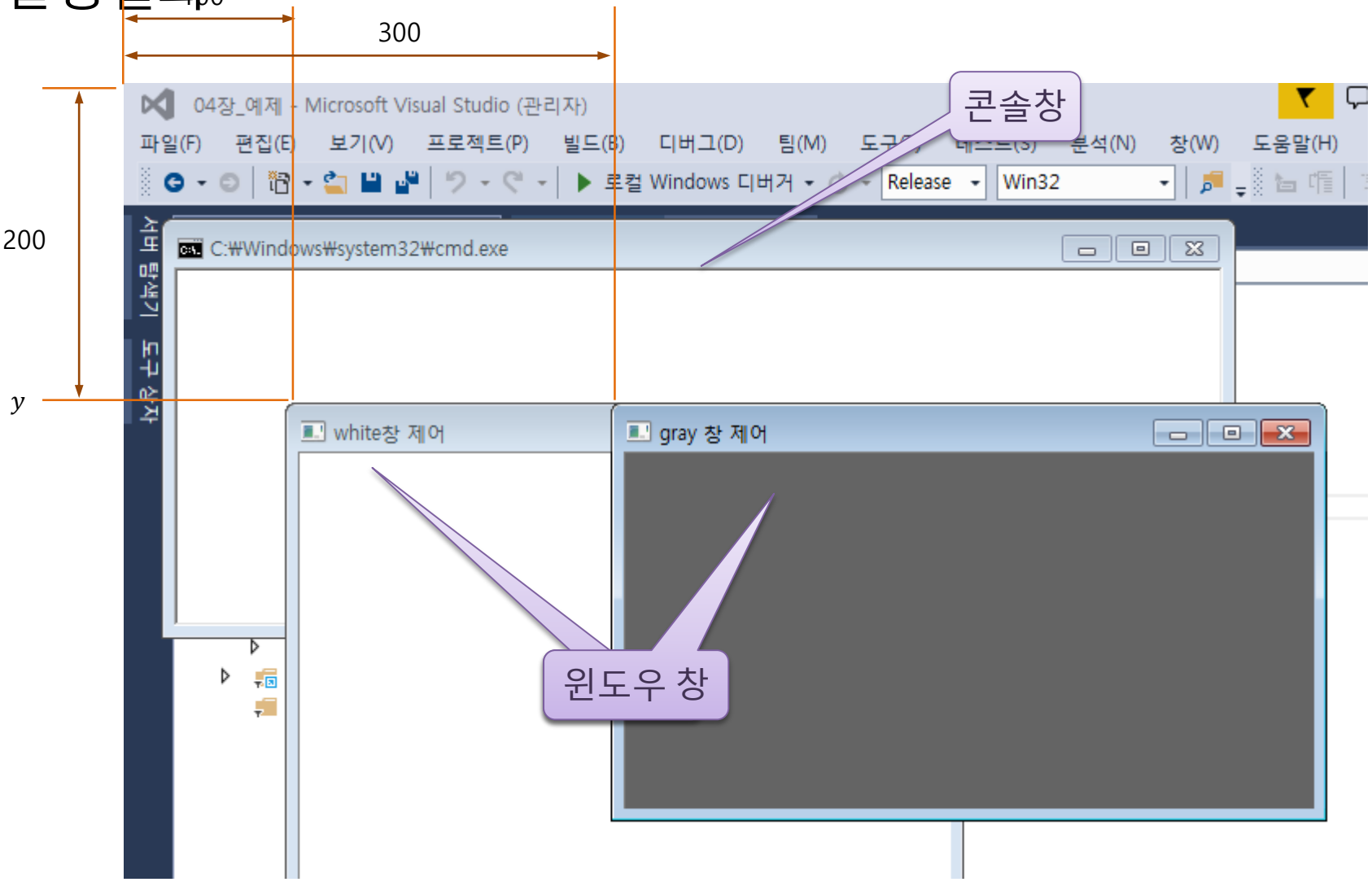
모니터의 가로 100, 세로 200 위치로 윈도우 이동

창 크기가 임의로 생성. 사용자가 마우스로 창의 크기 변경 가능

모니터의 200, 300 위치로 윈도우 이동

윈도우 창 제어

- 실행결과


100
300
200
y
04장_예제 - Microsoft Visual Studio (관리자)
파일(F)
편집(E)
보기(V)
프로젝트(P)
빌드(B)
디버그(D)
팀(M)
분석(N)
창(W)
도움말(H)
로컬 Windows 디버거
Release
Win32
서버 탐색기
도구 상자
C:₩Windows₩system32₩cmd.exe
콘솔창
white창 제어
gray 창 제어
윈도우 창

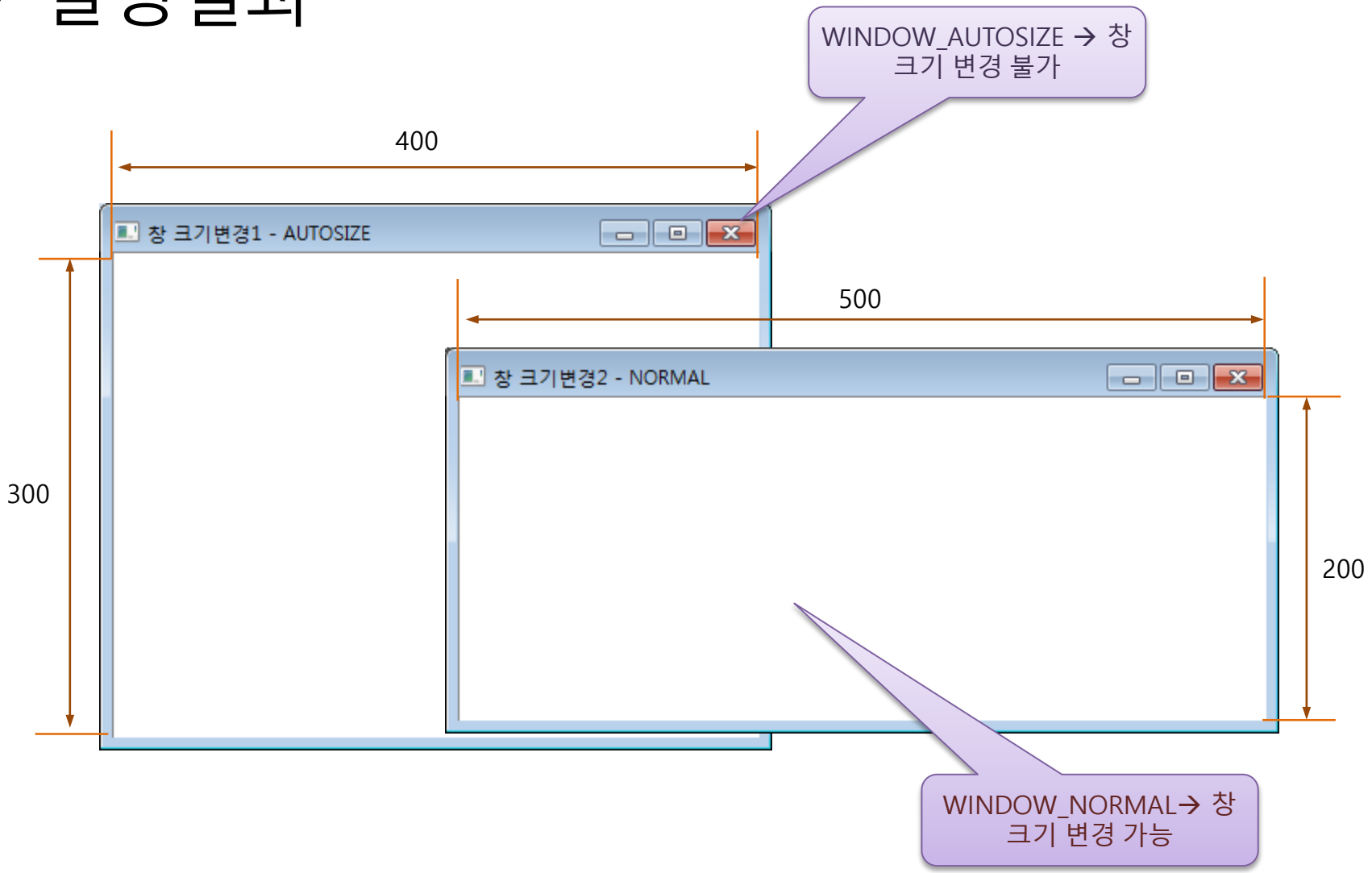
윈도우 창 제어

예제 4.1.2 윈도우 크기 변경 – window_resize.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04  int main()
05  {
06      Mat image(300, 400, CV_8U, Scalar(255));          // 흰색 바탕 영상 생성
07      string  title1 = "창 크기변경1 - AUTOSIZE";       // 윈도우 이름
08      string  title2 = "창 크기변경2 - NORMAL";
09
10      namedWindow(title1, WINDOW_AUTOSIZE);             // 윈도우 크기 변경 불가
11      namedWindow(title2, WINDOW_NORMAL);               // 윈도우 크기 변경 가능
12      resizeWindow(title1, 500, 200);                   // 창크기 변경
13      resizeWindow(title2, 500, 200);
14
15      imshow(title1, image);
16      imshow(title2, image);
17      waitKey();                                        // 키이벤트 대기
18      return 0;
19  }
```

윈도우 창 제어

- 실행결과


WINDOW_AUTOSIZE → 창 크기 변경 불가
400
창 크기변경1 - AUTOSIZE
300
500
창 크기변경2 - NORMAL
200
WINDOW_NORMAL→ 창 크기 변경 가능

키보드 이벤트 제어

- cv::waitKey()

함수명과 반환형 및 인수 구조	
`int waitKey(int delay = 0)`	
함수 및 인수	**설명**
`int waitKey()`	delay(ms) 시간만큼 키 입력 대기하고, 키 이벤트 발생하면 해당 키 값을 반환한다.
• `int delay`	지연시간, ms단위
- `delay ≤ 0`	키 이벤트 발생까지 무한 대기
- `delay > 0`	지연시간 동안 키 입력 대기, 지연시간 내에 키 이벤트 없으면 –1 반환

키보드 이벤트 제어

★실습

예제 4.2.1 키이벤트 사용 – event_key.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04  int main()
05  {
06      Mat image(200, 300, CV_8U, Scalar(255));
07      namedWindow("키보드 이벤트", WINDOW_AUTOSIZE);
08      imshow("키보드 이벤트", image);
09
10      while (1)                                   // 무한 반복
11      {
12          int key = waitKey(100);                 // 100ms 동안 키이벤
13          if (key == 27)  break;
14
15          switch (key)
16          {
17              case 'a':   cout << "a키 입력" << endl; break;
18              case 'b':   cout << "b키 입력" << endl; break;
                case 0x41:  cout << "A키 입력" << endl; break;
                case 66:    cout << "B키 입력" << endl; break;
21
22              case 0x250000:  cout << "왼쪽 화살표 키 입력" << endl; break
23              case 0x260000:  cout << "윗쪽 화살표 키 입력" << endl; break
24              case 0x270000:  cout << "오른쪽 화살표 키 입력" << endl; bre
25              case 0x280000:  cout << "아래쪽 화살표 키 입력" << endl; bre
26          }
27      }
28      return 0;
29  }
```

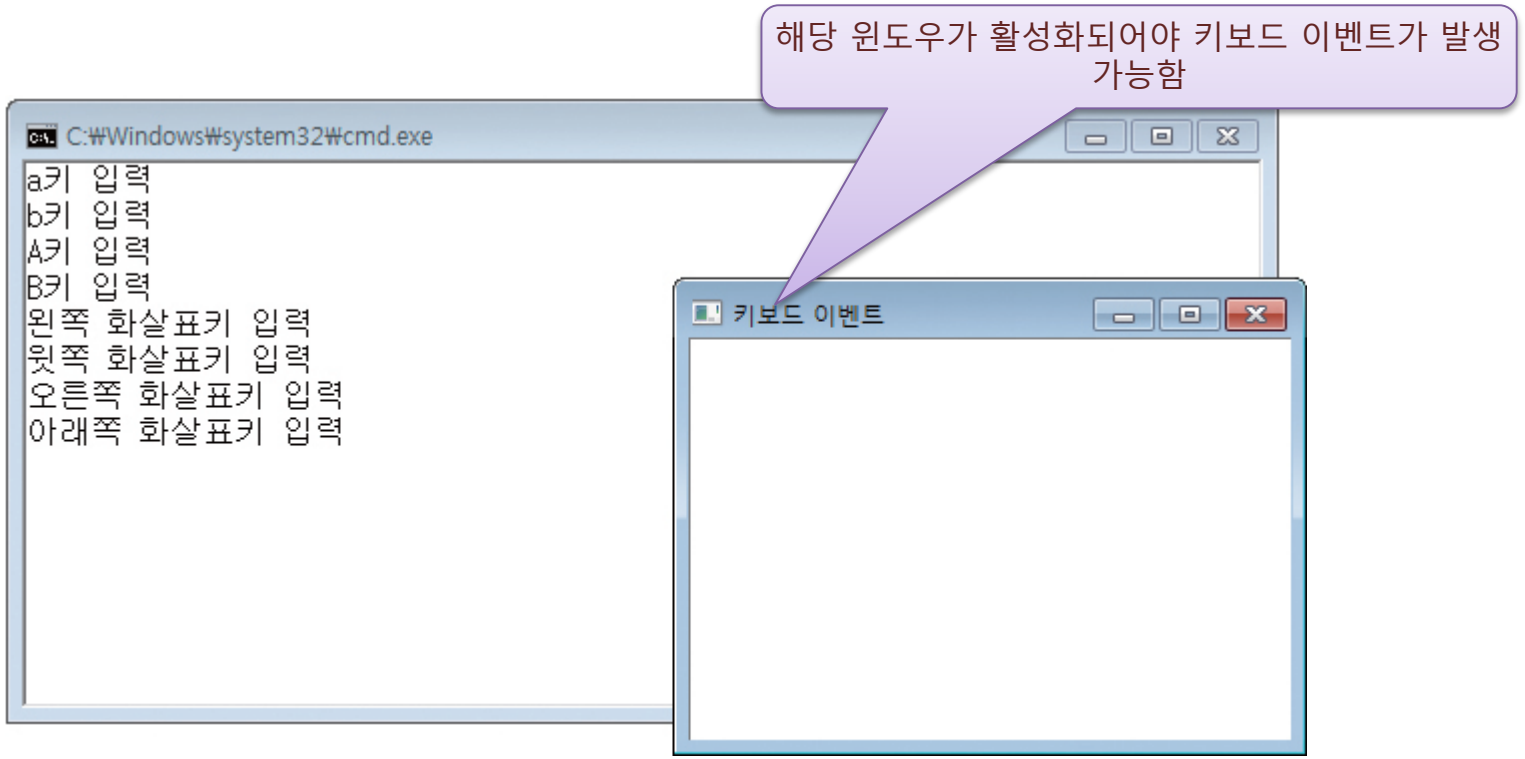
100ms 대기하며 키입력 기다림
키입력 없으면 -1 반환

무한반복 종료조건 (Esc 키)

키 값에 따른 출력

키보드 이벤트 제어

- 실행결과



키보드 이벤트 제어 - 예제

소스 분석

★실습

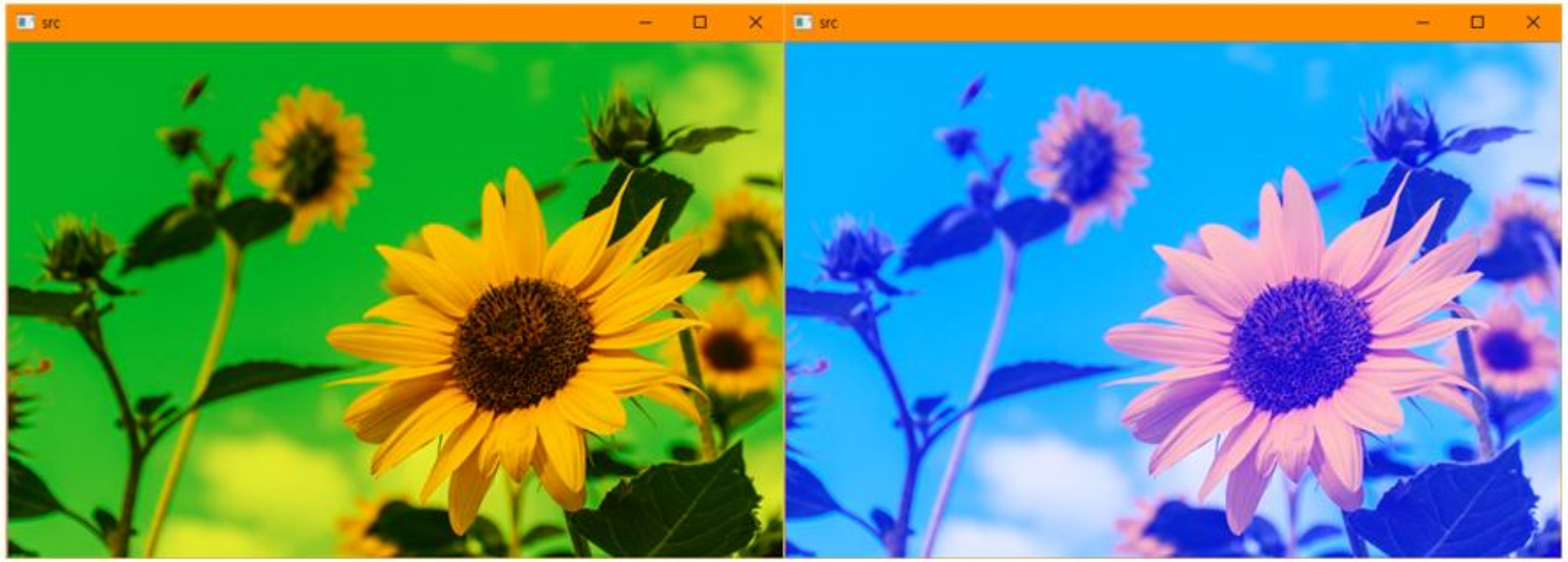
```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
		Mat img;
		img = imread("d:/dog.jpg", IMREAD_COLOR);
		if (img.empty()) { cout << "영상을 읽을 수 없음" << endl; }


		imshow("img", img);
		int x = 300;
		int y = 300;
```

소스 분석

```
        while (1) {

                int key = waitKey(100);
                if (key == 'q') break;
                else if (key == 'a')
                        x -= 10;
                else if (key == 'w')
                        y -= 10;
                else if (key == 'd')
                        x += 10;
                else if (key == 's')
                        y += 10;

                circle(img, Point(x, y), 200, Scalar(0, 255, 0), 5);
                imshow("img", img);
        }
        return 0;
}
```

100ms 동안 key 를 누리기를 기다림

키보드 영상 제어하기



★실습

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;

int main()
{
	Mat src = imread("d:/photo1.jpg", IMREAD_COLOR);
	if (src.empty()) { cout << "영상을 읽을 수 없음" << endl; }

	imshow("src", src);

	while (1) {
		int key = waitKeyEx();		// 사용자로부터 키를 기다린다.
		cout << key << " ";
		if (key == 'q') break;		// 사용자가 'q'를 누르면 종료한다.
		else if (key == 2424832) {	// 왼쪽화살표 키
			src -= 50;// 영상이 어두워진다.
		}
		else if (key == 2555904) {	// 오른쪽화살표 키
			src += 50;		// 영상이 밝아진다.
		}
		imshow("src", src);	// 영상이 변경되었으므로 다시 표시한다.
	}
	return 0;
}
```

마우스 이벤트 제어

- OpenCV에서 제공하는 콜백 (Callback) 함수를 통해서 이벤트를 발생 및 제어
  - 콜백 함수 – 개발자가 함수를 호출하는 것이 아니라, 어떤 이벤트가 발생하거나 특정 시점에 도달했을 때 시스템에서 개발자가 등록한 함수를 호출하는 방식
  - 마우스 이벤트 등록 : cv::setMouseCallback()

마우스 이벤트 제어

함수명과 반환형 및 인수 구조	
`void setMouseCallback(const string& winname, MouseCallback onMouse, void* userdata = 0)` `typedef  void (*MouseCallback)(int event, int x, int y, int flags, void* userdata)`	
함수 및 인수	**설명**
`void setMouseCallback()`	사용자가 정의한 마우스 콜백함수를 시스템에 등록하는 함수이다.
• `string& winname`	이벤트 발생을 체크할 윈도우 이름
• `MouseCallback onMouse`	마우스 이벤트를 처리하는 콜백 함수 이름(함수 포인터)
• `void* userdata`	이벤트 처리 함수로 전달할 추가적인 사용자 정의 인수
`void (*MouseCallback)()`	발생한 마우스 이벤트에 대해서 처리 및 제어를 구현하는 콜백 함수이다. setMouseCallback() 함수의 두 번째 인수(함수포인터)의 구현 부이기 때문에 함수명이 인수명과 같아야 함. typedef 통해서 함수포인터로 정의되어 있어 인수의 구조(인수의 데이터 타입, 인수의 순서 등)를 유지해야 함.
• `int event`	발생한 마우스 이벤트의 종류
• `int x, int y`	이벤트 발생 시 마우스 포인터의 x, y 좌표
• `int  flags`	마우스 버튼과 동시에 특수 키([Shift], [Alt], [Ctrl])가 눌려졌는지 여부 확인

옵션	값	설명
EVENT_FLAG_LBUTTON	1	왼쪽 버튼 누름
EVENT_FLAG_RBUTTON	2	오른쪽 버튼 누름
EVENT_FLAG_MBUTTON	3	중간 버튼 누름
EVENT_FLAG_CTRLKEY	8	[Ctrl] 키 누름
EVENT_FLAG_SHIFTKEY	16	[Shift] 키 누름
EVENT_FLAG_ALTKEY	32	[Alt] 키 누름

• `void* userdata`	콜백 함수로 전달되는 추가적인 사용자 정의 인수

마우스 이벤트 제어

표 4.2.1 마우스 이벤트 옵션

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

마우스 이벤트 제어

★실습

예제 4.2.2 마우스 이벤트 사용 – event_mouse.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04
05  void onMouse(int, int, int, int, void *);              // 마우스 콜백 함수 선언
06
07  int main()
08  {
09      Mat image(200, 300, CV_8U);
10      image.setTo(255);                                  // image 행렬 초기화 - 흰색 바탕
11      imshow("마우스 이벤트1", image);
12      imshow("마우스 이벤트2", image);
13
14      setMouseCallback("마우스 이벤트1", onMouse, 0);
15      waitKey(0);
16      return 0;
17  }
18
```

창이름

콜백 함수이름

마우스 이벤트 콜백함수 시스템에 등록

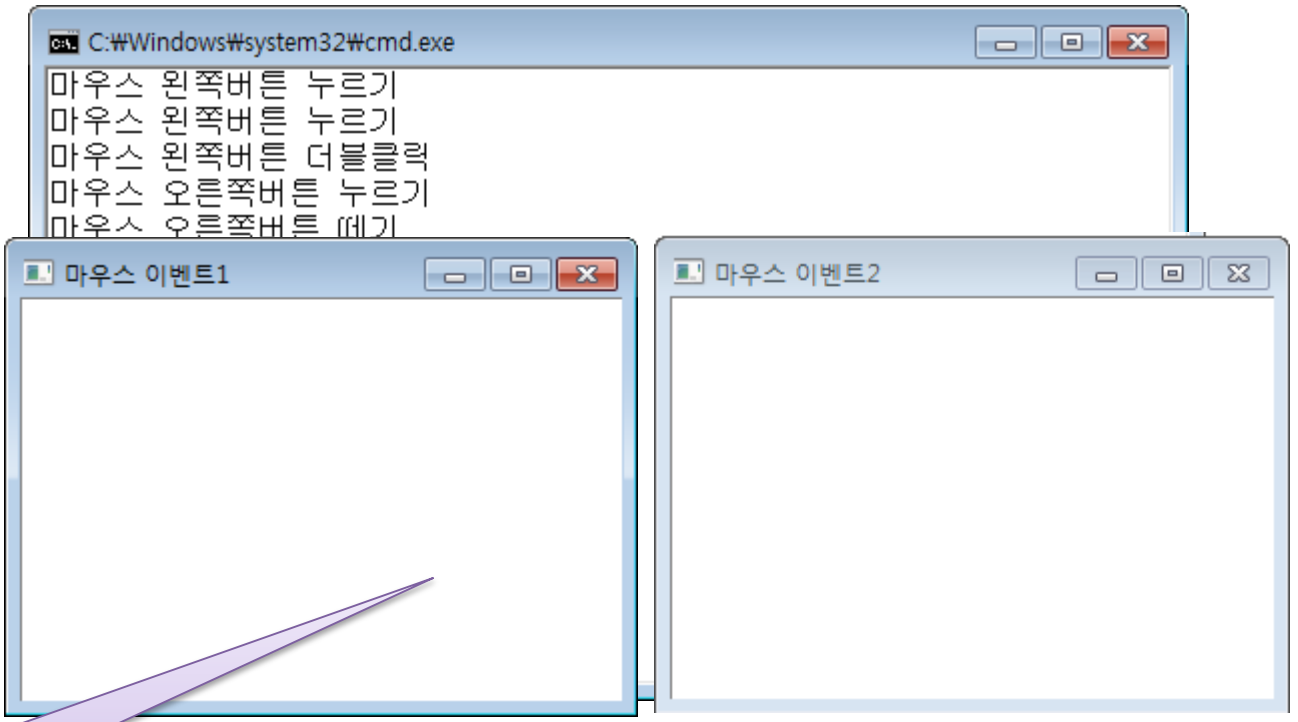
마우스 이벤트 제어

```
19  void onMouse(int event, int x, int y, int flags, void * param)
20  {
21      switch (event)                          // event값에 따라 버튼 종류 구분
22      {
23      case EVENT_LBUTTONDOWN:
24          cout << "마우스 왼쪽버튼 누르기" << endl;
25          break;
26      case EVENT_RBUTTONDOWN:
27          cout << "마우스 오른쪽 버튼 누르기" << endl;
28          break;
29      case EVENT_RBUTTONUP:
30          cout << "마우스 오른쪽 버튼 떼기" << endl;
31          break;
32      case EVENT_LBUTTONDBLCLK:
33          cout << "마우스 왼쪽버튼 더블클릭" << endl;
34          break;
35      }
36  }
```

마우스 이벤트 종류 : 왼쪽버튼 누르기

마우스 이벤트 제어

- 실행결과



콜백함수를 등록된 윈도우 창이 활성이라야 이벤트 감지 가능

마우스 이벤트 제어 - 예제

소스 분석

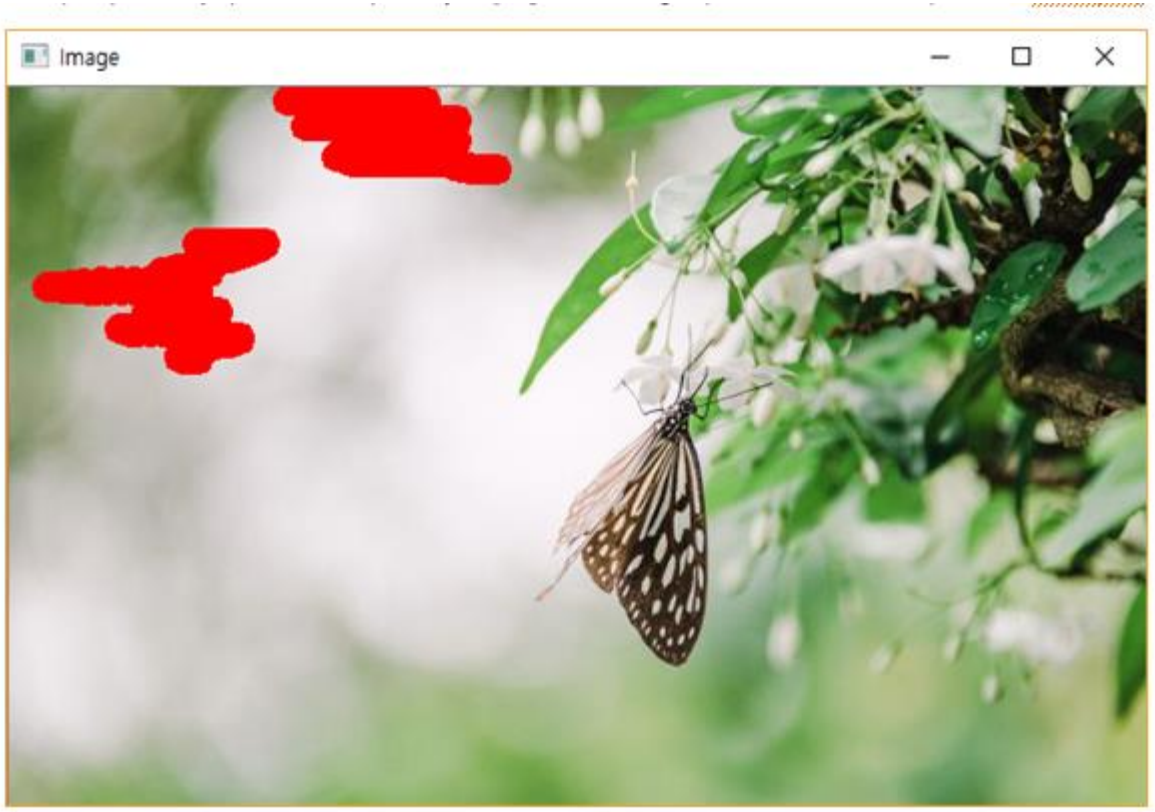
★실습

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;

// 마우스 이벤트가 발생하면 호출되는 콜백 함수이다.
void onMouse(int event, int x, int y, int flags, void* param)
{
		if (event == EVENT_LBUTTONDOWN) {
				Mat& img = *(Mat*)(param);
				circle(img, Point(x, y), 200, Scalar(0, 255, 0), 10);
				putText(img, "I found a dog!", Point(x, y + 200),
						FONT_HERSHEY_PLAIN, 2.0, 255, 2);
				imshow("src", img); // 영상이 변경되면 다시 표시한다.
		}
		else if (event == EVENT_RBUTTONDOWN) {}
		else if (event == EVENT_MBUTTONDOWN) {}
		else if (event == EVENT_MOUSEMOVE) {
		}
}
```

소스 분석

```
int main()
{
	Mat src = imread("d:/dog.jpg", IMREAD_COLOR);
	if (src.empty()) { cout << "영상을 읽을 수 없음" << endl; }
	imshow("src", src);

	setMouseCallback("src", onMouse, &src);
	waitKey(0);
	return 0;
}
```

마우스 이벤트 제어 - 예제



소스 분석

★실습

```
#include <opencv2/opencv.hpp>
#include<iostream>
using namespace std;
using namespace cv;

Mat img;
int drawing = false;

void drawCircle(int event, int x, int y, int, void* param) {
		if (event == CV_EVENT_LBUTTONDOWN)
			drawing = true;
		else if (event == CV_EVENT_MOUSEMOVE) {
			if (drawing == true)
				circle(img, Point(x, y), 3, Scalar(0, 0, 255), 10);
		}
		else if (event == CV_EVENT_LBUTTONUP)
			drawing = false;
		imshow("Image", img);
}
```

소스 분석

```
int main()
{
	img = imread("d:/bug.jpg");
	if (img.empty()) { cout << "영상을 읽을 수 없음" << endl;  return -1; }

	imshow("Image", img);

	setMouseCallback("Image", drawCircle);
	waitKey(0);
	imwrite("d:/bug1.jpg", img);
	return 0;
}
```

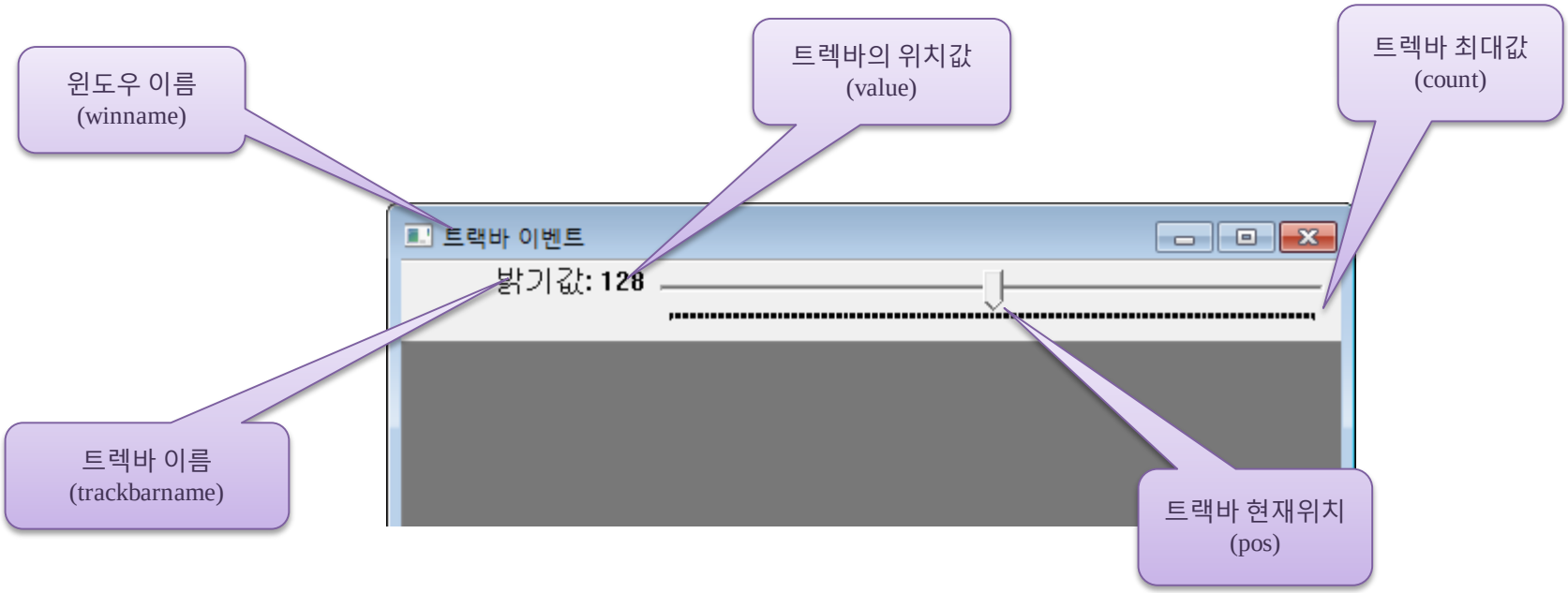
트랙바 이벤트 제어

- 트랙바
  - 일정한 범위 내에서 특정한 값을 선택하고자 할 때 사용하는 일종의 스크롤 바

함수명과 반환형 및 인수 구조	
int createTrackbar(const String& trackbarname, const String& winname, int* value, int count, TrackbarCallback onChange = 0, void* userdata = 0); typedef void (*TrackbarCallback)(int pos, void* userdata) int getTrackbarPos(const string& trackbarname, const string& winname) void setTrackbarPos(const string& trackbarname, const string& winname, int pos)	
함수 및 인수	**설명**
int createTrackbar()	트랙바를 생성하고, 그것을 지정된 윈도우 창에 추가하는 함수이다.
• String& trackbarname • String& winname • int * value • int count • TrackbarCallback onChange • void* userdata	생성된 트랙바 이름 트랙바의 부모 윈도우 이름(트랙바 이벤트 발생을 체크하는 윈도우) 트랙바 슬라이더의 위치를 반영하는 정수 값 슬라이더의 최대 위치, 최소 위치는 항상 0 트랙바 위치가 변경될 때마다 호출되는 콜백 함수의 포인터 콜백 함수로 전달할 추가적인 사용자 정의 인수
(*TrackbarCallback)()	트랙바의 위치가 변경될 때마다 호출되는 콜백 함수 • cv::createTrackbar()의 두 번째 인수(함수포인터)의 구현 부분이기 때문에 그 인수와 이름이 같아야 함. • typedef 지시어로 함수포인터로 정의되어 인수의 구조(인수의 데이터 타입, 인수의 순서 등)를 유지해야 함.
• int pos • void* userdata	트랙바 슬라이더 위치 콜백 함수로 전달되는 추가적인 사용자 정의 인수
int getTrackbarPos()	지정된 트랙바의 슬라이더 위치를 반환받는다.
void setTrackbarPos()	지정된 트랙바의 슬라이더 위치를 설정한다.

트랙바 이벤트 제어

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



트랙바 이벤트 제어

★실습

예제 4.2.3 트렉바 이벤트 사용 – event_trackbar.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04
05  string  title = "트랙바 이벤트";                    // 전역변수 선언
06  Mat  image;
07
08  void onChange(int  value, void*  userdata)
09  {
10      int  add_value = value - 130;
11      cout << "추가 화소값 " << add_value << endl;
12
13      Mat tmp = image + add_value;
14      imshow(title, tmp);
15  }
16
17  int main()
18  {
19      int  value = 128;
20      image = Mat(300, 400, CV_8UC1, Scalar(120));
21
22      namedWindow(title, WINDOW_AUTOSIZE);                    // 윈도우 생성
23      createTrackbar("밝기값", title, &value, 255, onChange);    // 트렉바 등록
24
25      imshow(title, image);
26      waitKey(0);
27      return 0;
28  }
```

-126~129 범위

트랙바 변경값

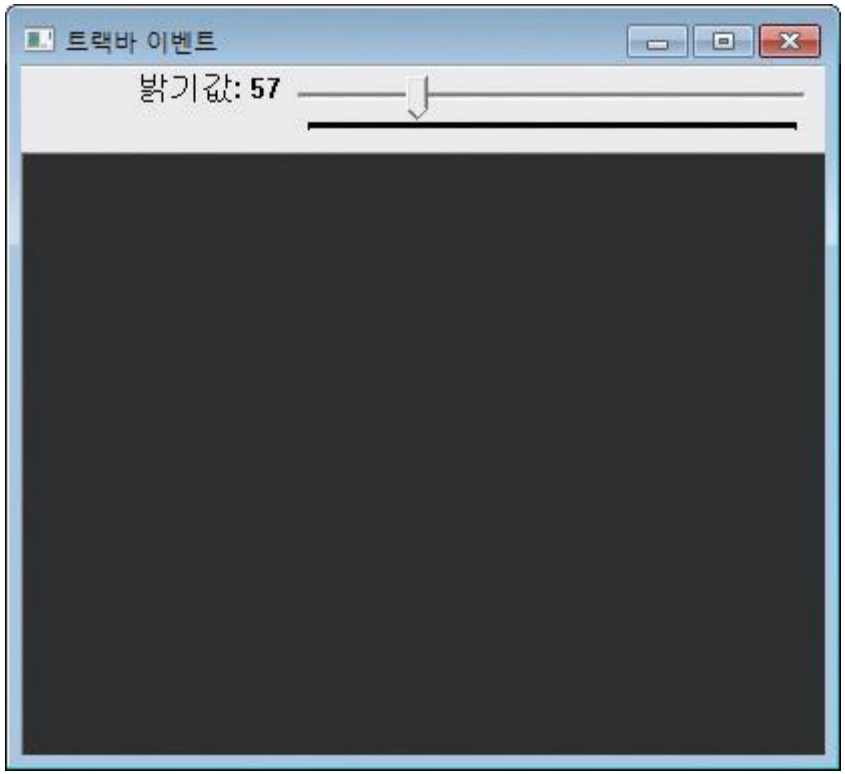
콜백함수 이름

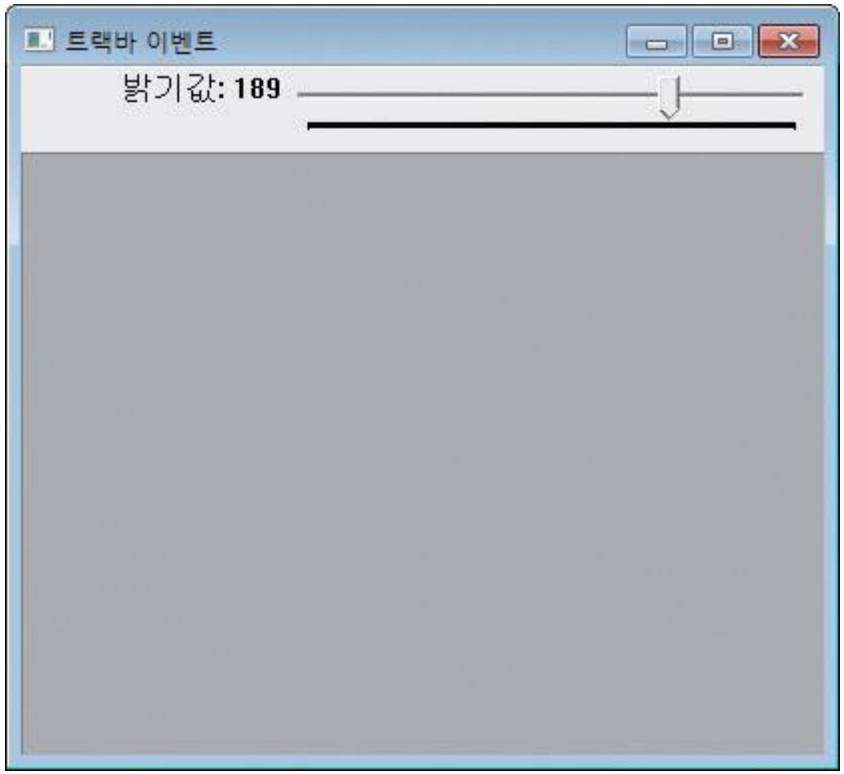
트랙바 등록할 창이름

트랙바 최대값

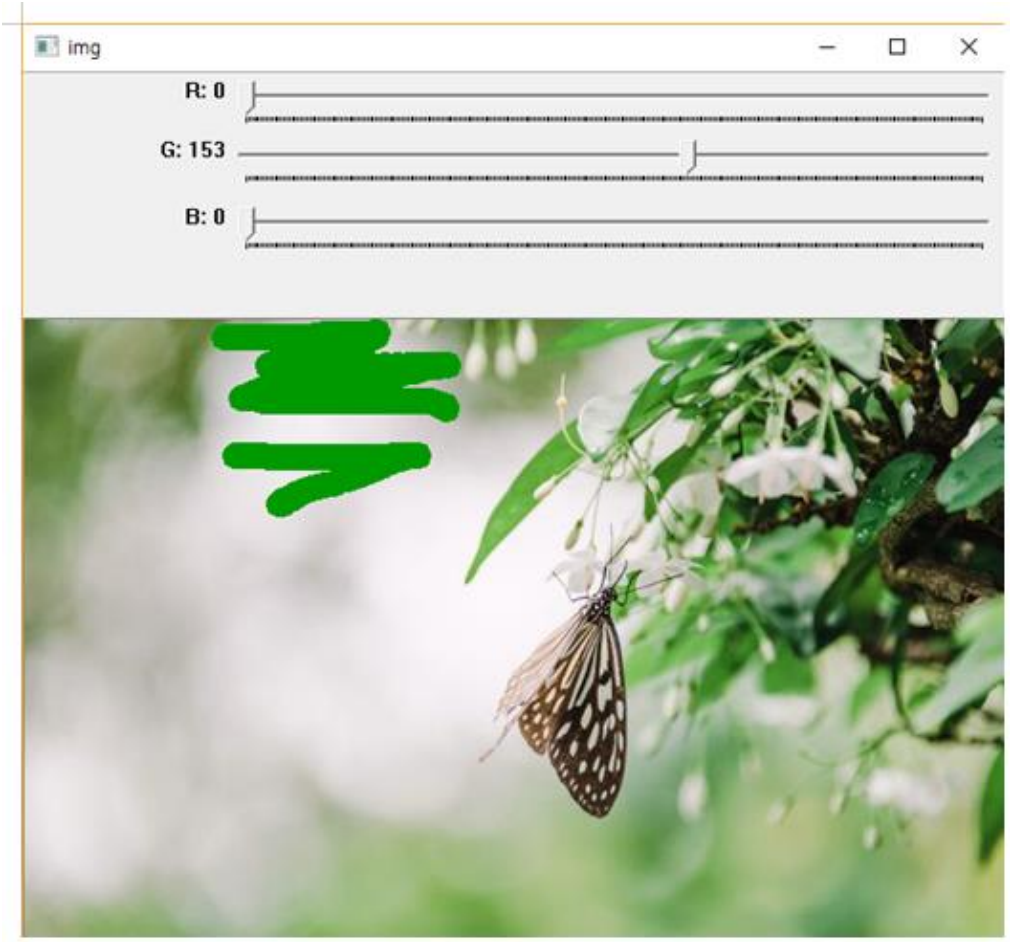
트랙바 이벤트 제어

- 실행결과





트랙바 이벤트 제어 - 예제



소스 분석

★실습

```
#include <opencv2/opencv.hpp>
#include<iostream>
using namespace std;
using namespace cv;

Mat img;
int red, green, blue;
int drawing = false;

void on_trackbar(int, void*) {      }

void drawCircle(int event, int x, int y, int, void* param) {
            if (event == CV_EVENT_LBUTTONDOWN)
                        drawing = true;
            else if (event == CV_EVENT_MOUSEMOVE) {
                        if (drawing == true)
                                    circle(img, Point(x, y), 3, Scalar(blue, green, red), 10);
            }
            else if (event == CV_EVENT_LBUTTONUP)
                        drawing = false;
            imshow("img", img);
}
```

소스 분석

```
int main()
{
		img = imread("d:/bug.jpg");
		if (img.empty()) { cout << "영상을 읽을 수 없음" << endl;  return -1; }
		namedWindow("img", 1);
		imshow("img", img);
		setMouseCallback("img", drawCircle);
		createTrackbar("R", "img", &red, 255, on_trackbar);
		createTrackbar("G", "img", &green, 255, on_trackbar);
		createTrackbar("B", "img", &blue, 255, on_trackbar);
		waitKey(0);
		return 0;
}
```

HW

1. 800 x 600 크기의 윈도우를 만들고 마우스 오른쪽 버튼을 누르면 100 x 100 크기의 사각형을 그리고, 왼쪽버튼을 누르면 반지름 100 인 원을 그리는 프로그램을 작성하시오.

2. 위 문제에 다음을 추가하여 프로그램을 작성하시오
   (1) 트랙바를 추가해서 선의 굵기를 1~10 픽셀로 조절한다.
   (2) 트랙바를 추가해서 원의 반지름을 10~200 픽셀로 조절한다.