

OpenCV 6차시

Event driven

○ 마우스 이벤트 처리

마우스 동작에 의해 발생하는 이벤트들은 다음과 같다.

마우스 움직임, 마우스버튼 클릭, 마우스 휠 움직임

예를 들어, 각 이벤트를 나타내는 정의는 다음과 같다.

EVENT_MOUSEMOVE (마우스 움직임), EVENT_LBUTTONDOWN (왼쪽 버튼 누름)

이벤트 처리를 위해서 ‘callback’ 함수를 이용한다.

Callback함수를 만들어서 시스템에 등록해 두면, 이벤트가 발생할 때 마다 그 함수가 호출된다.

예를 들어, 마우스 이벤트를 위한 callback함수를 시스템에 등록하기 위해서는 함수 `setMouseCallback()`을 이용한다.

이 때, callback함수는 다음과 같은 형태여야 한다.

```
void callback_name (int event, int x, int y, int flags, void *userdata)
```

. callback_name: 함수 이름으로 임의로 지정가능

. event: 어떤 이벤트인지 정보를 담고 있다. 예) EVENT_LBUTTONDOWN

. x, y: 이벤트가 발생한 지점의 (x, y) 좌표

. flags: 이벤트 발생시점에서 다른 키의 조합

. userdata: 기타 부가정보

※ 마우스 이벤트는 ‘프로그램에 의해 만들어진 window’ 안에서만 동작한다.

다음 프로그램은 마우스 좌, 우측 버튼을 누를 때마다 어느 버튼이 눌렸는지를 콘솔 창에 출력한다.

. callback 함수로 ‘onMouse’를 정의했다.

. 화면에 height=200, width=300인 window를 흰색으로 표시한다.

```

1  #include <opencv2/opencv.hpp>
2
3  using namespace cv;
4  using namespace std;
5
6  void onMouse(int event, int x, int y, int flags, void* userdata); // callback
7
8  int main(void)
9  {
10
11     Mat image(200, 300, CV_8U); // ( 200, 가 300, 8 unsigned=> )
12     image.setTo(255); // 255
13
14     imshow("MouseEvent1", image);
15     setMouseCallback("MouseEvent1", onMouse, 0); // call back
16     waitKey(0);
17
18 }
19
20 void onMouse(int event, int x, int y, int flags, void *param)
21 {
22     switch (event)
23     {
24     case EVENT_LBUTTONDOWN:
25         cout << "Left Button Click" << endl;
26         break;
27     case EVENT_RBUTTONDOWN:
28         cout << "Right Button Click" << endl;
29         break;
30     default:
31         break;
32     }
33 }

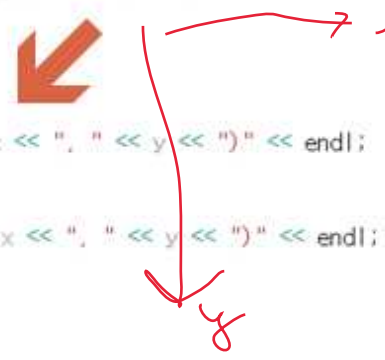
```

○ 마우스 이벤트가 발생한 위치의 (x,y) 좌표를 출력해 보자.

Callback 함수 onMouse에 인수로 주어진 x, y를 이용하여 좌표를 출력해 보자.

※ x좌표는 가로방향, y좌표는 세로방향임에 주의. 또한 y좌표는 아래로 갈수록 증가

```
21 void onMouse(int event, int x, int y, int flags, void *param)
22 {
23     switch (event)
24     {
25     case EVENT_LBUTTONDOWN:
26         cout << "Left Button Click at (" << x << ", " << y << ")" << endl;
27         break;
28     case EVENT_RBUTTONDOWN:
29         cout << "Right Button Click at (" << x << ", " << y << ")" << endl;
30         break;
31     default:
32         break;
33     }
34 }
```



○ 마우스 왼쪽 버튼을 누를 때 마다 해당 위치에 반지름 10인 원을 빨간 색으로 그려 보자.

circle() : 이미지 위에 원을 그리는 함수

circle (Mat& img, Point p, int radius, Scalar color, int thickness)

. img: 원을 그려 넣을 이미지

. Point: 원의 중심좌표 클래스 Point(x,y)를 이용하여 생성

. radius: 원의 반지름

. Scalar color: 원 색깔

. thickness: 선의 굵기, -1이면 안을 채운 원이 그려진다.

전역변수 Mat image

Callback함수 onMouse에서 접근하기 위해서 전역변수 선언

컬러이미지 Mat image

원을 빨간색으로 그리기 위해서, image를 CV_8UC3로 선언

C3: channel이 3개라는 뜻으로 **B, G, R 컬러를** 의미 // 8bit unsigned가

8U: 각 channel의 값이 8bit unsigned 임을 의미

imshow 재호출

원을 그린 이후에 화면을 갱신하기 위해서 함수 imshow()를 재호출

```

1
2  #include <opencv2/opencv.hpp>
3
4  using namespace cv;
5  using namespace std;
6
7  void onMouse(int event, int x, int y, int flags, void* userdata);
8
9  Mat image; //
10
11  int main(void)
12  {
13
14      image = Mat(200, 300, CV_8UC3);
15      image.setTo(Scalar(255,255,255)); // bgr      255
16
17      imshow("MouseEvent1", image);
18      setMouseCallback("MouseEvent1", onMouse, 0);
19      waitKey(0);
20  }
21
22
23  void onMouse(int event, int x, int y, int flags, void *param)
24  {
25      switch (event)
26      {
27      case EVENT_LBUTTONDOWN:
28          cout << "Left Button Click at (" << x << ", " << y << ")" << endl;
29          circle(image, Point(x, y), 10, Scalar(0, 0, 255), 3);
30          imshow("MouseEvent1", image); //
31          break;
32      case EVENT_RBUTTONDOWN:
33          cout << "Right Button Click at (" << x << ", " << y << ")" << endl;
34          break;
35      default:
36          break;
37      }
38  }

```

- 전역변수를 사용하지 않고 마우스 이벤트 처리

전역변수를 이용하는 것은 구조 측면에서 바람직하지 않다. 위에서 작성한 프로그램을 전역변수 Mat image를 사용하지 않는 방향으로 바꾸어 보자.

함수 setMouseCallback()의 세 번째 인수를 이용

세 번째 인수는 사용자가 지정하는 포인터이다. 이 포인터를 이용하여 Mat image의 주소를 callback함수에 넘길 수 있다.

Callback함수에서 Mat image 접근

함수 onMouse()의 마지막 인수 void* param을 이용하여 Mat image를 접근

param은 Mat image의 주소이므로, 이를 적절히 casting하여 사용가능

```

1  #include <opencv2/opencv.hpp>
2
3  using namespace cv;
4  using namespace std;
5
6  void onMouse(int event, int x, int y, int flags, void* userdata);
7
8  int main(void)
9  {
10
11     Mat image = Mat(200, 300, CV_8UC3);
12     image.setTo(Scalar(255, 255, 255));
13
14     imshow("MouseEvent1", image);
15     setMouseCallback("MouseEvent1", onMouse, &image);
16     waitKey(0);
17 }
18
19 void onMouse(int event, int x, int y, int flags, void* param) // void
20 {
21     Mat *_img = (Mat *)param; // Main 가 Mat
22
23     switch (event)
24     {
25     case EVENT_LBUTTONDOWN:
26         cout << "Left Button Click at (" << x << ", " << y << ")" << endl;
27         circle(*_img, Point(x, y), 10, Scalar(0, 0, 255), 3);
28         imshow("MouseEvent1", *_img);
29         break;
30     case EVENT_RBUTTONDOWN:
31         cout << "Right Button Click at (" << x << ", " << y << ")" << endl;
32         break;
33     default:
34         break;
35     }
36 }
37

```

o 마우스 클릭한 곳의 픽셀값을 출력

컬러이미지를 읽어서 화면에 표시하고, 마우스 왼쪽 버튼을 클릭한 곳의 픽셀값을 출력한다.

컬러이미지의 한 픽셀은 3 byte

(Blue, Green, Red)의 값이 하나의 픽셀을 구성. 각 색깔이 가질 수 있는 값은 0 ~ 255.

클래스 Mat의 at () 함수 이용 // x,y

좌표 상의 픽셀값을 읽는 함수로서, 주의할 점은 화면상의 (x,y) 좌표를 이용하는 것이 아니라, (row, column)을 이용한다. 따라서 화면상의 (x,y) 좌표는 함수 at(y, x)로 읽어야 한다.

π Vec3b == array , 3b = 3 byte

Vec3b를 이용한 BGR 값 추출

픽셀의 BGR 값은 Vec3b 클래스를 이용하여 추출하고, 각 컬러는 배열 인덱스 [0],[1],[2]를 이용하여 추출 가능하다.

```
31 void onMouse(int event, int x, int y, int flags, void *param)
32 {
33     Mat *_img = (Mat *)param;
34     Vec3b bgr = 0;
35
36     switch (event)
37     {
38     case EVENT_LBUTTONDOWN:
39         cout << "Left Button Click at (" << x << ", " << y << ")" << endl;
40         bgr = _img->at<Vec3b>(y, x);
41         cout << "BGR value is " << bgr << endl;
42         cout << "BLUE value is " << (int)bgr[0] << endl;
43         cout << "GREEN value is " << (int)bgr[1] << endl;
44         cout << "RED value is " << (int)bgr[2] << endl;
45         break;
46     case EVENT_RBUTTONDOWN:
47         cout << "Right Button Click at (" << x << ", " << y << ")" << endl;
48         break;
49     default:
50         break;
51     }
52 }
```

특정영역을 확대해서 보기

마우스로 클릭한 부분을 확대해서 새로운 창에서 보여주는 것을 만들어보자. 우선 마우스로 클릭한 지점을 좌상단으로 해서, 가로와 세로 30씩인 사각형 영역을 추출한다. 이 영역을 10배 확대하여 새로운 창에 표시한다.

Mat (Rect)를 이용한 사각형 영역 추출

Mat image의 (x,y)좌표에서 가로,세로 30씩인 사각형 영역을 추출하기 위해서는

Mat roi = `image(Rect(x, y, 30, 30));`

ROI는 'region of interest' , '관심영역' 의 약자이다.

```
32 void onMouse(int event, int x, int y, int flags, void *param)
33 {
34     Mat _img = *(Mat *)param;
35     Mat roi;
36
37     switch (event)
38     {
39     case EVENT_LBUTTONDOWN:
40         cout << "Left Button Click at (" << x << ", " << y << ")" << endl;
41         if (x + 30 < _img.cols && y + 30 < _img.rows) //
42         {
43             roi = _img(Rect(x, y, 30, 30));
44             resize(roi, roi, Size(), 10.0, 10.0);
45             imshow("ROI", roi);
46         }
47         break;
48     case EVENT_RBUTTONDOWN:
49         cout << "Right Button Click at (" << x << ", " << y << ")" << endl;
50         break;
51     default:
52         break;
53     }
54 }
```

실습:

컬러이미지를 읽어서 화면에 표시한 후, 마우스로 사각영역을 왼쪽 마우스 누르고 drag 한 후에 버튼을 놓아서 선택한다.

선택된 영역에 대해서 canny edge detection한 결과를 표시하시오.

오른쪽 마우스 버튼으로 선택된 영역에 대해서는 Otsu 방식으로 binarization한 결과를 표시하시오.

※ 재사용가능한 클래스를 설계하여 구현하는 것이 핵심.