# OpenCV Tutorial C++



## How to Detect Mouse Clicks and Moves

OpenCV supports for detecting mouse events. Mouse events include mouse clicks and movements over an attached OpenCV window.

### OpenCV Example Code

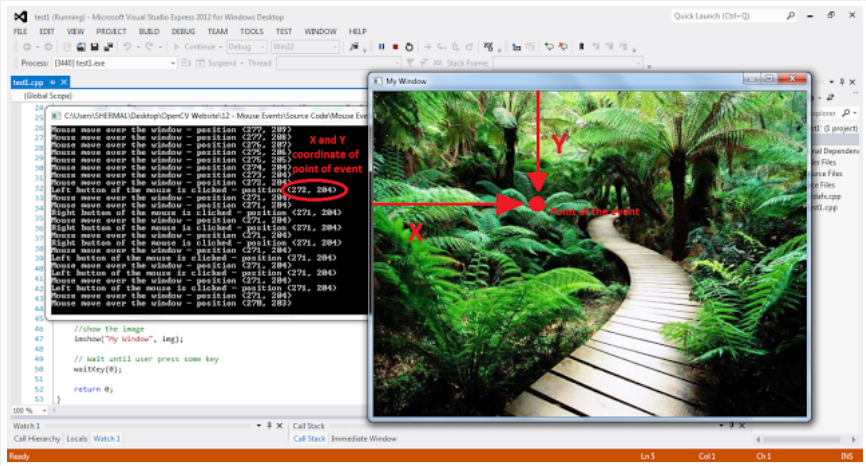
It is very simple to do that. All you have do is to define a callback function in the OpenCV C++ code attaching to the OpenCV window. That callback function will be called every time, mouse events occur. That callback function will also give the coordinates of the mouse events. (e.g - (x, y) coordinate of a mouse click).

Here is the simple OpenCV code to detect left, right and middle mouse clicks and mouse movements with its coordinates

```
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace std;
using namespace cv;

void CallBackFunc(int event, int x, int y, int flags, void* userdata)
{
    if  ( event == EVENT_LBUTTONDOWN )
    {
        cout << "Left button of the mouse is clicked - position (" << x << ", " << y << ")" << endl;
    }
    else if  ( event == EVENT_RBUTTONDOWN )
    {
        cout << "Right button of the mouse is clicked - position (" << x << ", " << y << ")" << endl;
    }
    else if  ( event == EVENT_MBUTTONDOWN )
    {
        cout << "Middle button of the mouse is clicked - position (" << x << ", " << y << ")" << endl;
    }
    else if ( event == EVENT_MOUSEMOVE )
    {
        cout << "Mouse move over the window - position (" << x << ", " << y << ")" << endl;

    }
}

int main(int argc, char** argv)
{
    // Read image from file
    Mat img = imread("MyPic.JPG");

    //if fail to read the image
    if ( img.empty() )
    {
        cout << "Error loading the image" << endl;
        return -1;
    }

    //Create a window
    namedWindow("My Window", 1);

    //set the callback function for any mouse event
    setMouseCallback("My Window", CallBackFunc, NULL);

    //show the image
    imshow("My Window", img);

    // Wait until user press some key
    waitKey(0);
```

```
    return 0;

}
```

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
You can download this OpenCV visual C++ project from here




Detect Mouse Clicks and Moves Over the OpenCV Window

## Summary

In the above OpenCV sample code, "CallbackFunc" function will be called on any mouse event (Moving a mouse over the attached OpenCV window is also a mouse event). By using suitable if - else blocks, I printed only left, right and middle mouse clicks and mouse movements over the window.

Here are new OpenCV functions, found in the above example code. If you are not familiar with the other OpenCV functions as well, please go through the other lessons in this tutorial.

- **void setMouseCallback(const string& winname, MouseCallback onMouse, void* userdata = 0)**

This function sets a callback function to be called every time any mouse events occurs in the specified window. Here is the detailed explanation of the each parameters of the above OpenCV function.

- **winname** - Name of the OpenCV window. All mouse events related to this window will be registered
- **onMouse** - Name of the callback function. Whenever mouse events related to the above window occur, this callback function will be called. This function should have the signature like the following
  - **void FunctionName(int event, int x, int y, int flags, void* userdata)**
    - **event** - Type of the mouse event. These are the entire list of mouse events
      - EVENT_MOUSEMOVE
      - EVENT_LBUTTONDOWN
      - EVENT_RBUTTONDOWN
      - EVENT_MBUTTONDOWN
      - EVENT_LBUTTONUP
      - EVENT_RBUTTONUP
      - EVENT_MBUTTONUP
      - EVENT_LBUTTONDBLCLK
      - EVENT_RBUTTONDBLCLK
      - EVENT_MBUTTONDBLCLK
    - **x** - x coordinate of the mouse event
    - **y** - y coordinate of the mouse event
    - **flags** - Specific condition whenever a mouse event occurs. See the next OpenCV example code for the usage of this parameter. Here is the entire list of enum values which will be possesed by

"flags"

- EVENT_FLAG_LBUTTON
- EVENT_FLAG_RBUTTON
- EVENT_FLAG_MBUTTON
- EVENT_FLAG_CTRLKEY
- EVENT_FLAG_SHIFTKEY
- EVENT_FLAG_ALTKEY

- **userdata** - Any pointer passes to the "**setMouseCallback**" function as the 3rd parameter (see below)

- **userdata** - This pointer will be passed to the callback function

### OpenCV Example to Detect Mouse Clicks While Pressing a Key

I am going to explain you how to detect a mouse event while pressing a key of the keyboard.

The following OpenCV example code will detect left mouse clicks while pressing the "CTRL" key , right mouse clicks while pressing the "SHIFT" key and movements of the mouse over the OpenCV window while pressing the "ALT" key.

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace std;
using namespace cv;

void CallBackFunc(int event, int x, int y, int flags, void* userdata)
{
    if ( flags == (EVENT_FLAG_CTRLKEY + EVENT_FLAG_LBUTTON) )
    {
        cout << "Left mouse button is clicked while pressing CTRL key - position (" << x << ", " << y << ")" << endl;
    }
    else if ( flags == (EVENT_FLAG_RBUTTON + EVENT_FLAG_SHIFTKEY) )
    {
        cout << "Right mouse button is clicked while pressing SHIFT key - position (" << x << ", " << y << ")" << endl;
    }
    else if ( event == EVENT_MOUSEMOVE && flags == EVENT_FLAG_ALTKEY)
    {
        cout << "Mouse is moved over the window while pressing ALT key - position (" << x << ", " << y << ")" << endl;
    }
}

int main(int argc, char** argv)
{
    // Read image from file
    Mat img = imread("MyPic.JPG");

    //if fail to read the image
    if ( img.empty() )
    {
        cout << "Error loading the image" << endl;
        return -1;
    }

    //Create a window
    namedWindow("My Window", 1);

    //set the callback function for any mouse event
    setMouseCallback("My Window", CallBackFunc, NULL);

    //show the image
    imshow("My Window", img);

    // Wait until user press some key
    waitKey(0);

    return 0;
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////
```

You can download this visual C++ OpenCV project from here.

No new OpenCV functions in the above example. If you have some doubt about any OpenCV functions in the above example, please refer to previous lessons.

## Next Tutorial : Rotate Image & Video