```
#include <iostream>
#include <sstream>
#include <Kinect.h>
#include <opencv2¥opencv.hpp>
#include <atlbase.h>
// 次のように使います
// ERROR_CHECK( ::GetDefaultKinectSensor( &kinect ) );
// 書籍での解説のためにマクロにしています。実際には展開した形で使うことを検討してください。
#define ERROR_CHECK( ret )  ¥
    if ( (ret) != S_OK ) {    ¥
        std::stringstream ss; ¥
        ss << "failed " #ret " " << std::hex << ret << std::endl;   ¥
        throw std::runtime_error( ss.str().c_str() );   ¥
        }
class KinectApp
{
private:
    // Kinect SDK
    CComPtr<IKinectSensor> kinect = nullptr;
    CComPtr<IDepthFrameReader> depthFrameReader = nullptr;
    // 表示部分
    int depthWidth;
    int depthHeight;
    std::vector<UINT16> depthBuffer;
    int depthPointX;
    int depthPointY;
    const char* DepthWindowName = "Depth Image";
public:
    ~KinectApp()
    {
        // Kinectの動作を終了する
        if (kinect != nullptr){
            kinect->Close();
        }
    }
    // 初期化
    void initialize()
    {
        // デフォルトのKinectを取得する
        ERROR_CHECK(::GetDefaultKinectSensor(&kinect));
        ERROR_CHECK(kinect->Open());
        // Depthリーダーを取得する
        CComPtr<IDepthFrameSource> depthFrameSource;
        ERROR_CHECK(kinect->get_DepthFrameSource(&depthFrameSource));
        ERROR_CHECK(depthFrameSource->OpenReader(&depthFrameReader));
        // Depth画像のサイズを取得する
        CComPtr<IFrameDescription> depthFrameDescription;
        ERROR_CHECK(depthFrameSource->get_FrameDescription(
            &depthFrameDescription));
        ERROR_CHECK(depthFrameDescription->get_Width(&depthWidth));
        ERROR_CHECK(depthFrameDescription->get_Height(&depthHeight));
        depthPointX = depthWidth / 2;
        depthPointY = depthHeight / 2;
        // Depthの最大値、最小値を取得する
        UINT16 minDepthReliableDistance;
        UINT16 maxDepthReliableDistance;
        ERROR_CHECK(depthFrameSource->get_DepthMinReliableDistance(
            &minDepthReliableDistance));
        ERROR_CHECK(depthFrameSource->get_DepthMaxReliableDistance(
            &maxDepthReliableDistance));
        std::cout << "Depth最小値       : " << minDepthReliableDistance << std::endl;
        std::cout << "Depth最大値       : " << maxDepthReliableDistance << std::endl;
        // バッファーを作成する
        depthBuffer.resize(depthWidth * depthHeight);
        // マウスクリックのイベントを登録する
        cv::namedWindow(DepthWindowName);
        cv::setMouseCallback(DepthWindowName, &KinectApp::mouseCallback, this);
    }
    // マウスイベントのコールバック
    static void mouseCallback(int event, int x, int y, int flags, void* userdata)
    {
        // 引数に渡したthisポインタを経由してメンバ関数に渡す
        auto pThis = (KinectApp*)userdata;
```

マウスCallback用

デストラクタ

Depthデータ持有の部分（自分で触らない）

```
        pThis->mouseCallback(event, x, y, flags);
    }
    // マウスイベントのコールバック(実処理)
    void mouseCallback(int event, int x, int y, int flags)
    {
        if (event == CV_EVENT_LBUTTONDOWN) {
            depthPointX = x;
            depthPointY = y;
        }
    }
    void run()
    {
        while (1) {
            update();
            draw();
            auto key = cv::waitKey(10);
            if (key == 'q'){
                break;
            }
        }
    }
private:
    // データの更新処理
    void update()
    {
        updateDepthFrame();
    }
    void updateDepthFrame()
    {
        // Depthフレームを取得する
        CComPtr<IDepthFrame> depthFrame;
        auto ret = depthFrameReader->AcquireLatestFrame(&depthFrame);
        if (ret != S_OK){
            return;
        }
        // データを取得する
        ERROR_CHECK(depthFrame->CopyFrameDataToArray(
            depthBuffer.size(), &depthBuffer[0]));
    }
    void draw()
    {
        drawDepthFrame();
    }
    void drawDepthFrame()
    {
        // Depthデータを表示する
        cv::Mat depthImage(depthHeight, depthWidth, CV_8UC1);
        // Depthデータを0-255のグレーデータにする
        for (int i = 0; i < depthImage.total(); ++i){
            depthImage.data[i] = depthBuffer[i] % 255;
        }

        // Depthデータのインデックスを取得して、その場所の距離を表示する
        int index = (depthPointY * depthWidth) + depthPointX;
        std::stringstream ss;
        ss << depthBuffer[index] << "mm";
        cv::circle(depthImage, cv::Point(depthPointX, depthPointY), 10,
            cv::Scalar(0, 0, 255), 2);
        cv::putText(depthImage, ss.str(), cv::Point(depthPointX, depthPointY),
            0, 1, cv::Scalar(0, 255, 255));
        cv::imshow(DepthWindowName, depthImage);
    }
};
void main()
{
    try {
        KinectApp app;
        app.initialize();
        app.run();
    }
    catch (std::exception& ex){
        std::cout << ex.what() << std::endl;
    }
```

マウスをクリックしたときの座標がこれに格納される

上下左右

8bit 1チャンネル

赤

B=0
G=255
R=255
⇒黄

灰色になるのは depthImageが1チャンネル(モノクロ)データのため

```
}
```