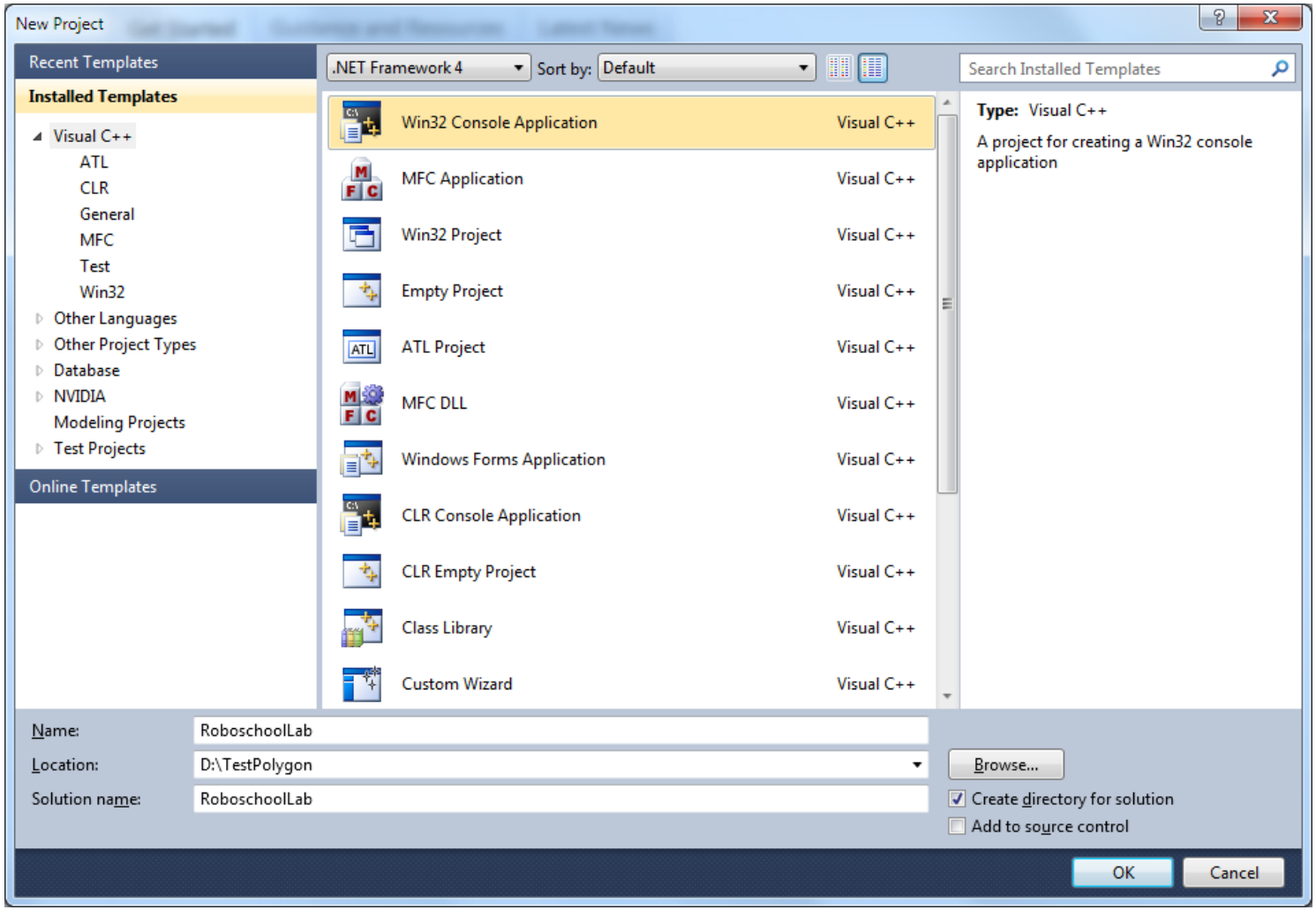
Computer vision. Лабораторная работа

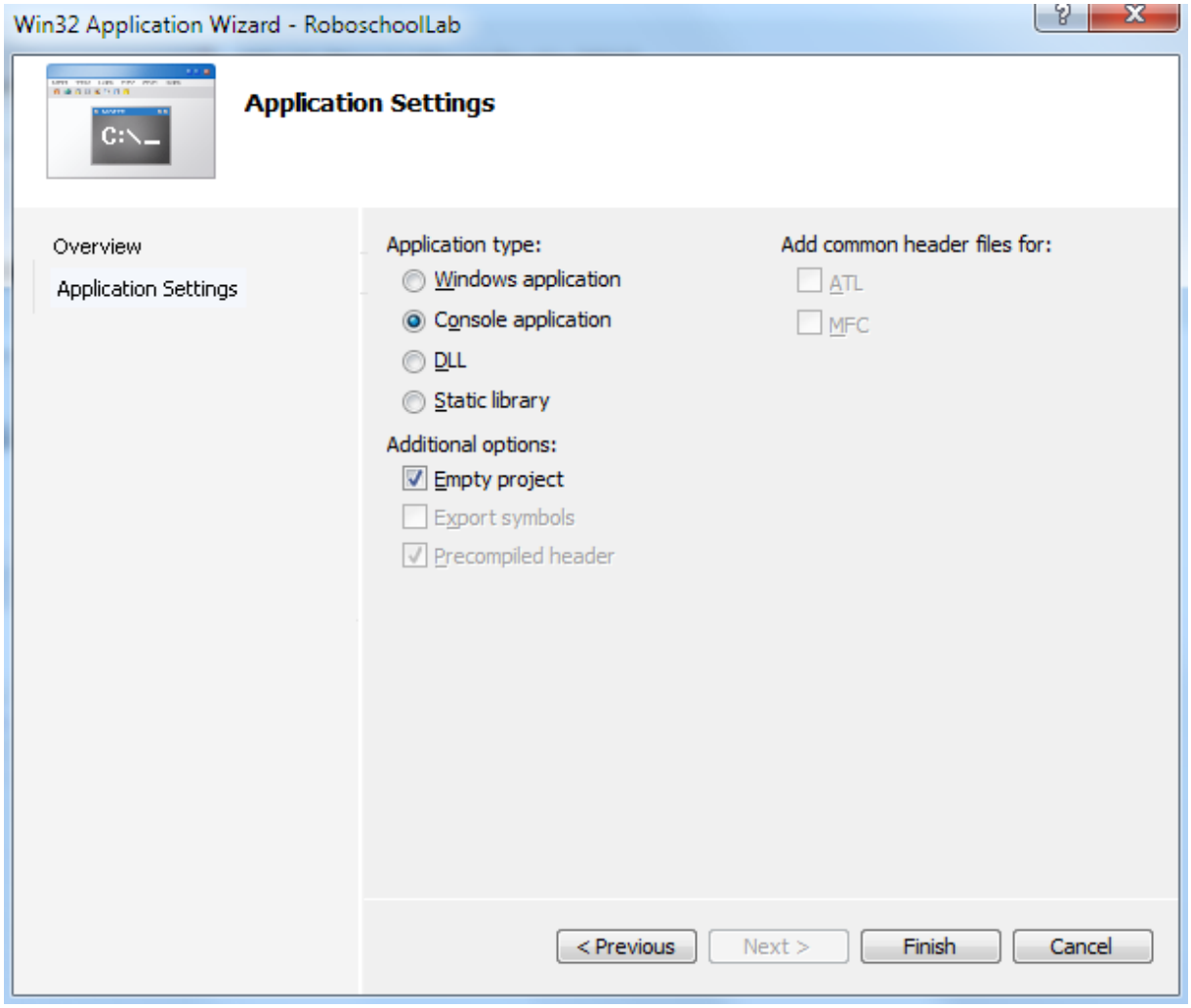
Алексеев Алексей aleksey.alekseev@singularis-lab.com
Александр Катаев alexander.kataev@singularis-lab.com

Создание нового проекта



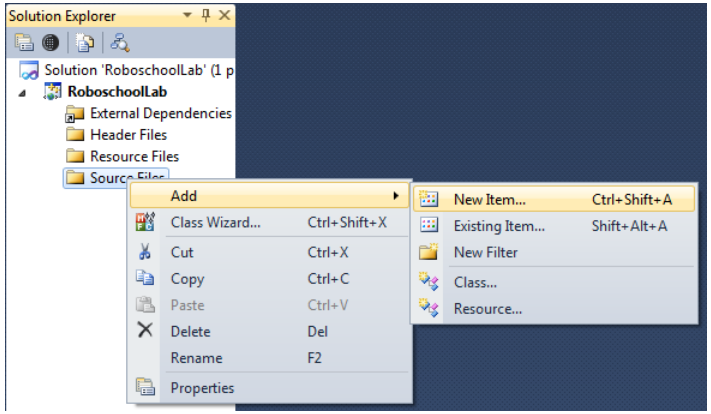
Создание нового проекта

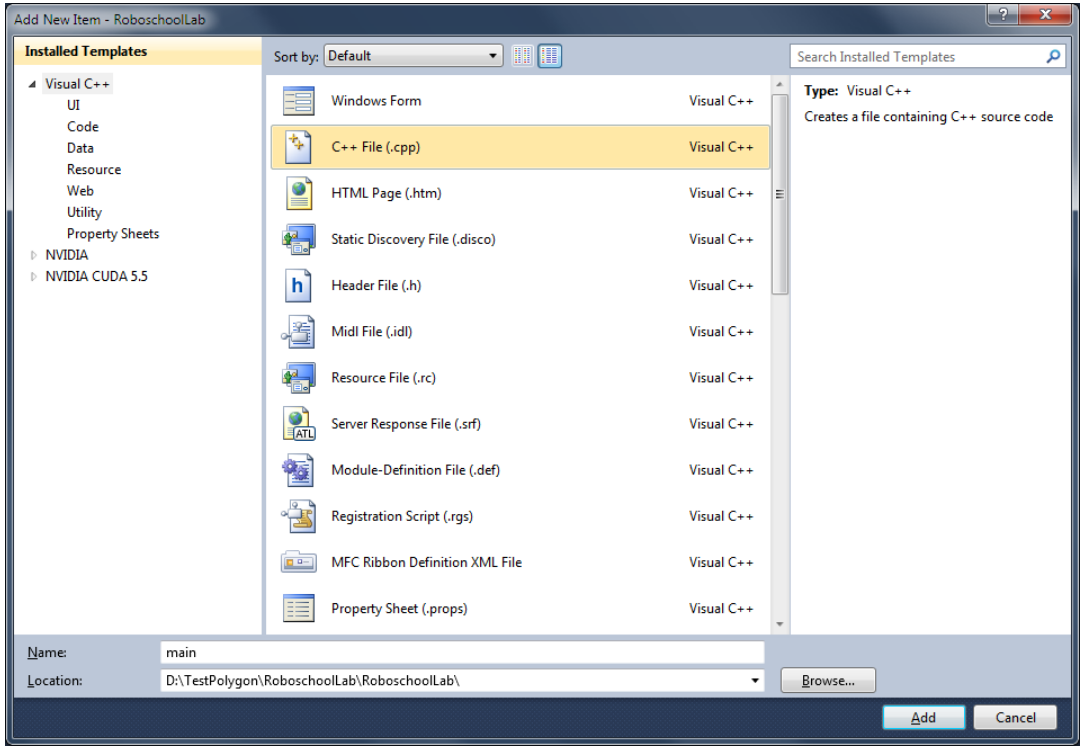
Установите галочку “Empty project”



Создание нового проекта

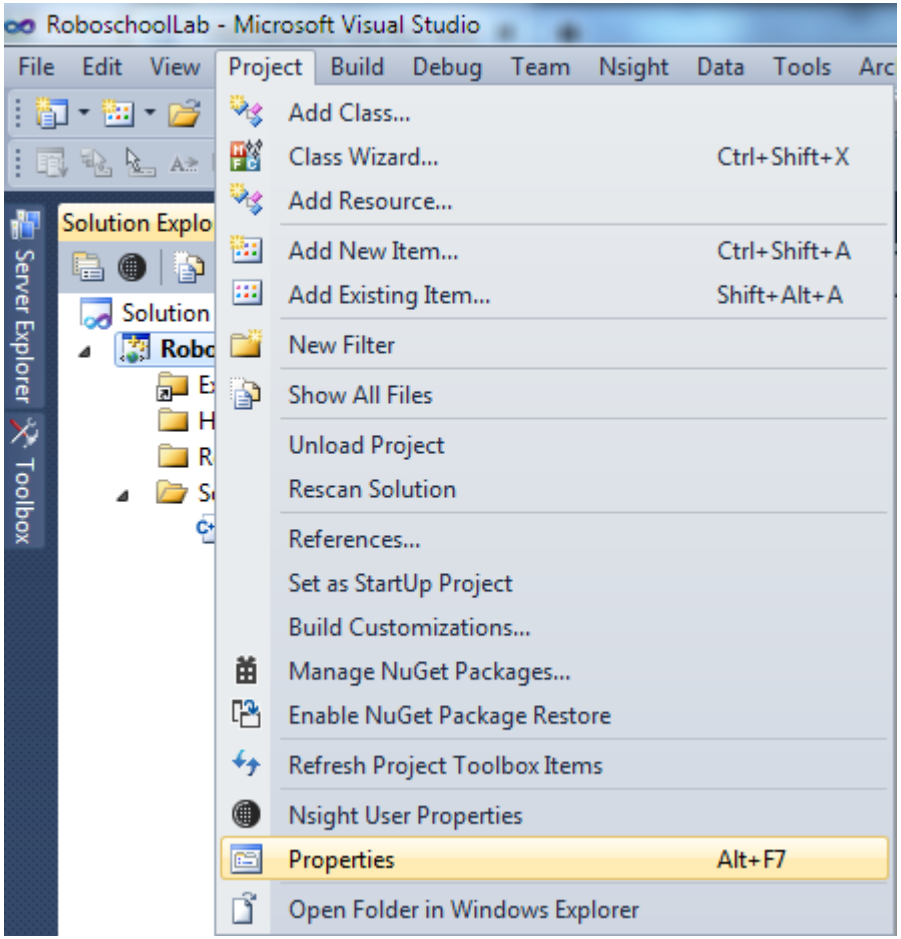
Добавьте пустой .cpp файл и укажите его имя:





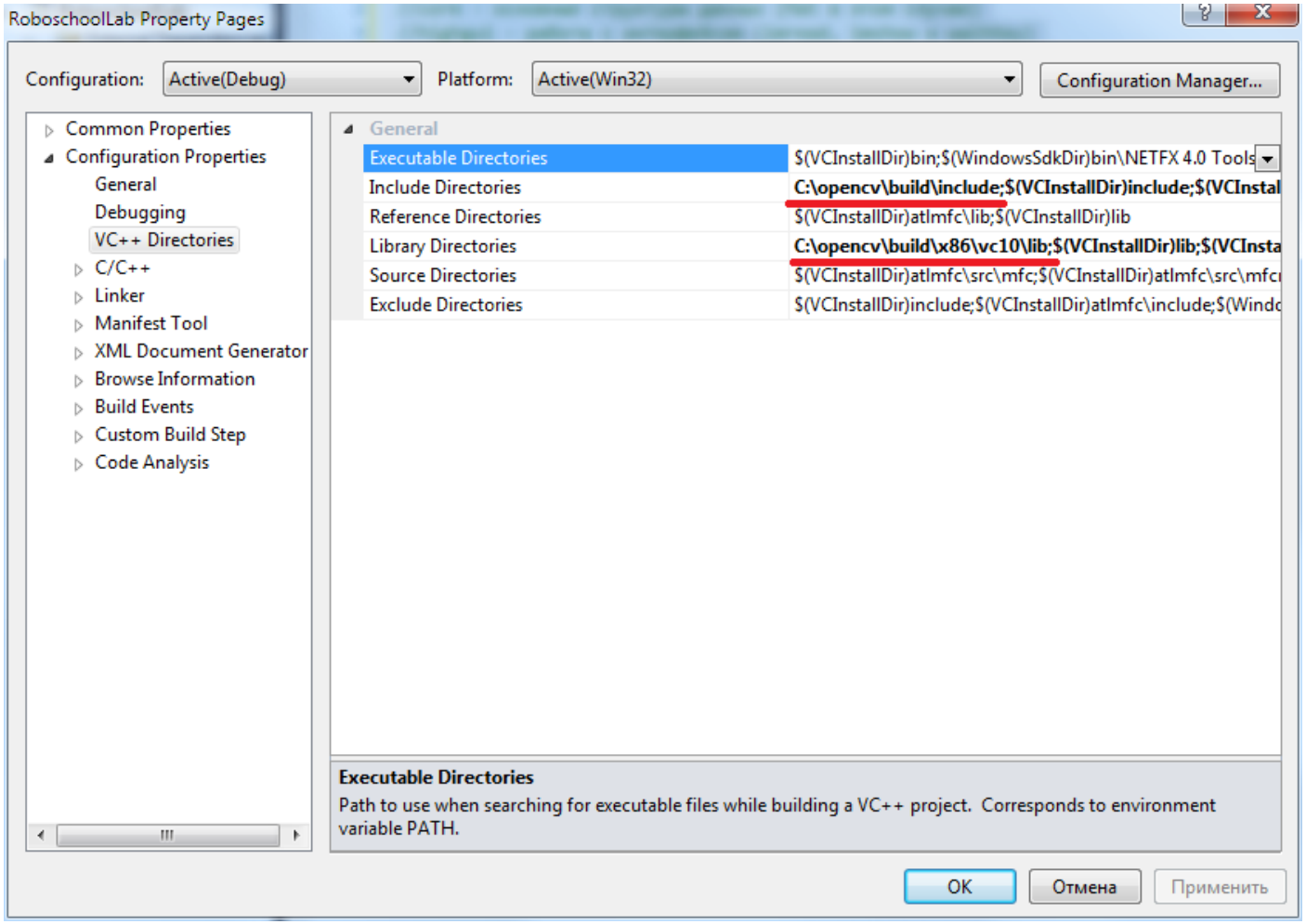
Настройка проекта

Зайдите в свойства проекта:



Настройка проекта

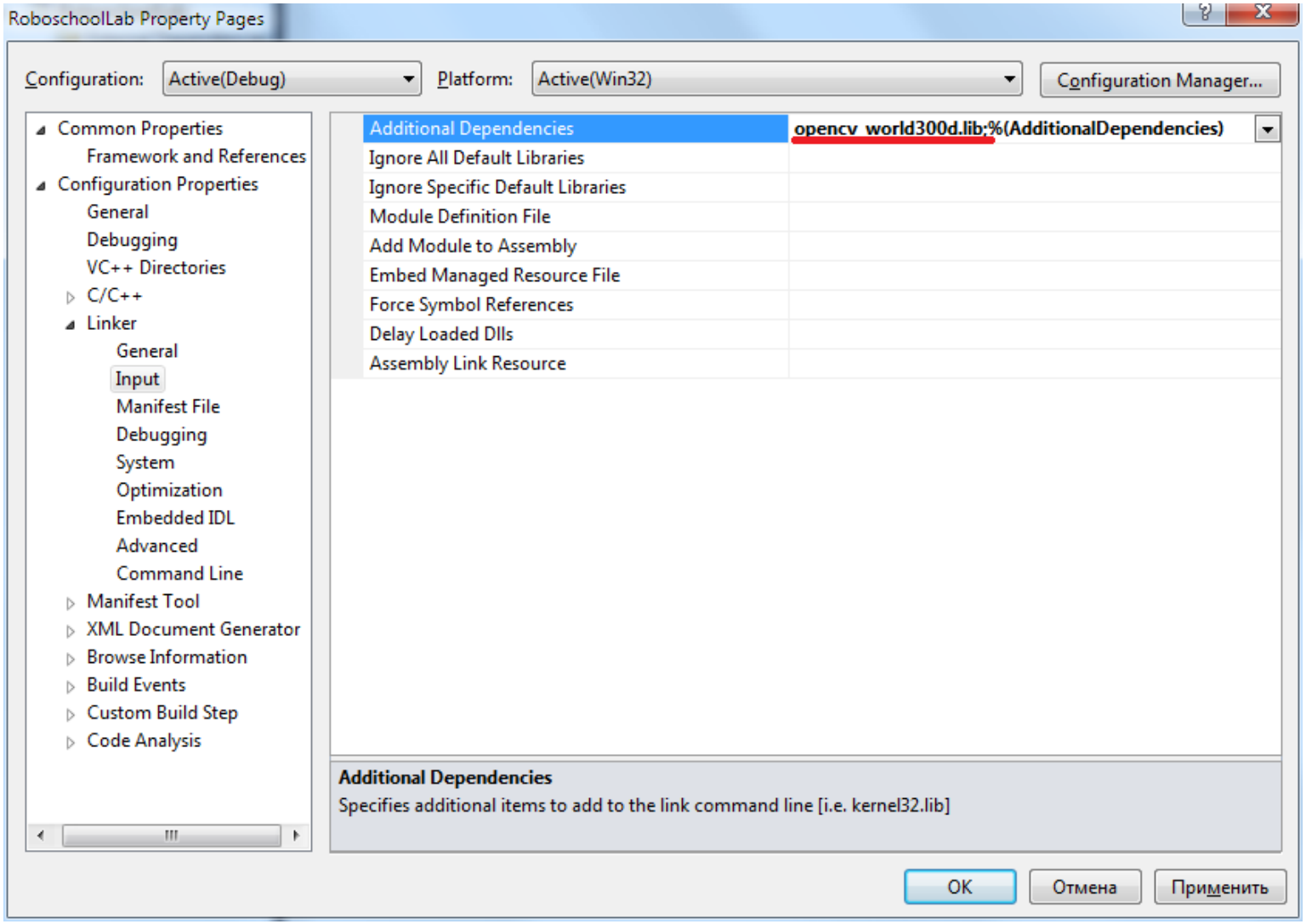
Configuration Properties -> VC++ Directories:



+ Необходимо добавить путь c:\opencv\build\x86\vc10\bin в системную переменную Path

Настройка проекта

Configuration Properties -> Linker -> Input:



Открытие изображения

```
//заголовочные файлы opencv
//core - основные структуры данных (Mat в этом случае)
//highgui - работа с интерфейсом (imread, imshow и waitKey)
#include <opencv2\core.hpp>
#include <opencv2\highgui.hpp>

using namespace cv;

int main(int argc, char *argv)
{
    //загружаем изображение, параметр - путь до изображения
    Mat image = imread("Lenna.png");
    //выводим изображение на экран,
    //первый параметр - имя окна,
    //второй - изображение
    imshow("image", image);
    // ожидаем нажатия клавиши
    waitKey();
    return 0;
}
```

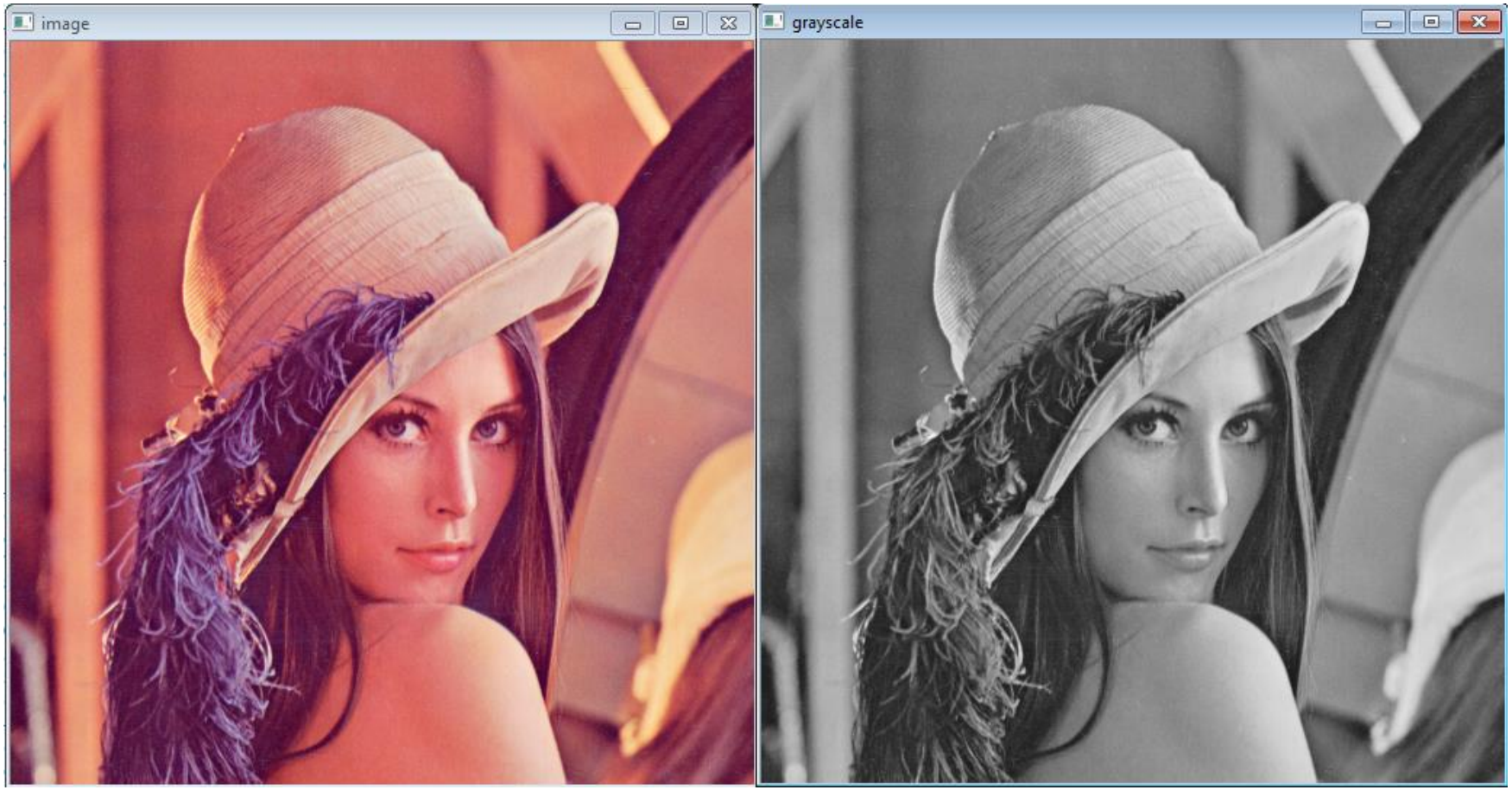
Перевод изображения в градации серого

```
#include <opencv2\core.hpp>
#include <opencv2\highgui.hpp>
//необходим для основных функций обработки изображений (cvtColor)
#include <opencv2\imgproc.hpp>

using namespace cv;

int main(int argc, char *argv)
{
    Mat image = imread("Lenna.png");
    Mat grayscale;
    //Конвертирует изображение из одного
    //цветового пространства в другое
    //первый параметр - исходное изображение
    //второй параметр - полученное изображение
    //третий параметр - код конвертации (другой пример - COLOR_BGR2HSV)
    cvtColor(image, grayscale, COLOR_BGR2GRAY);
    imshow("image", image);
    imshow("grayscale", grayscale);
    waitKey();
    return 0;
}
```

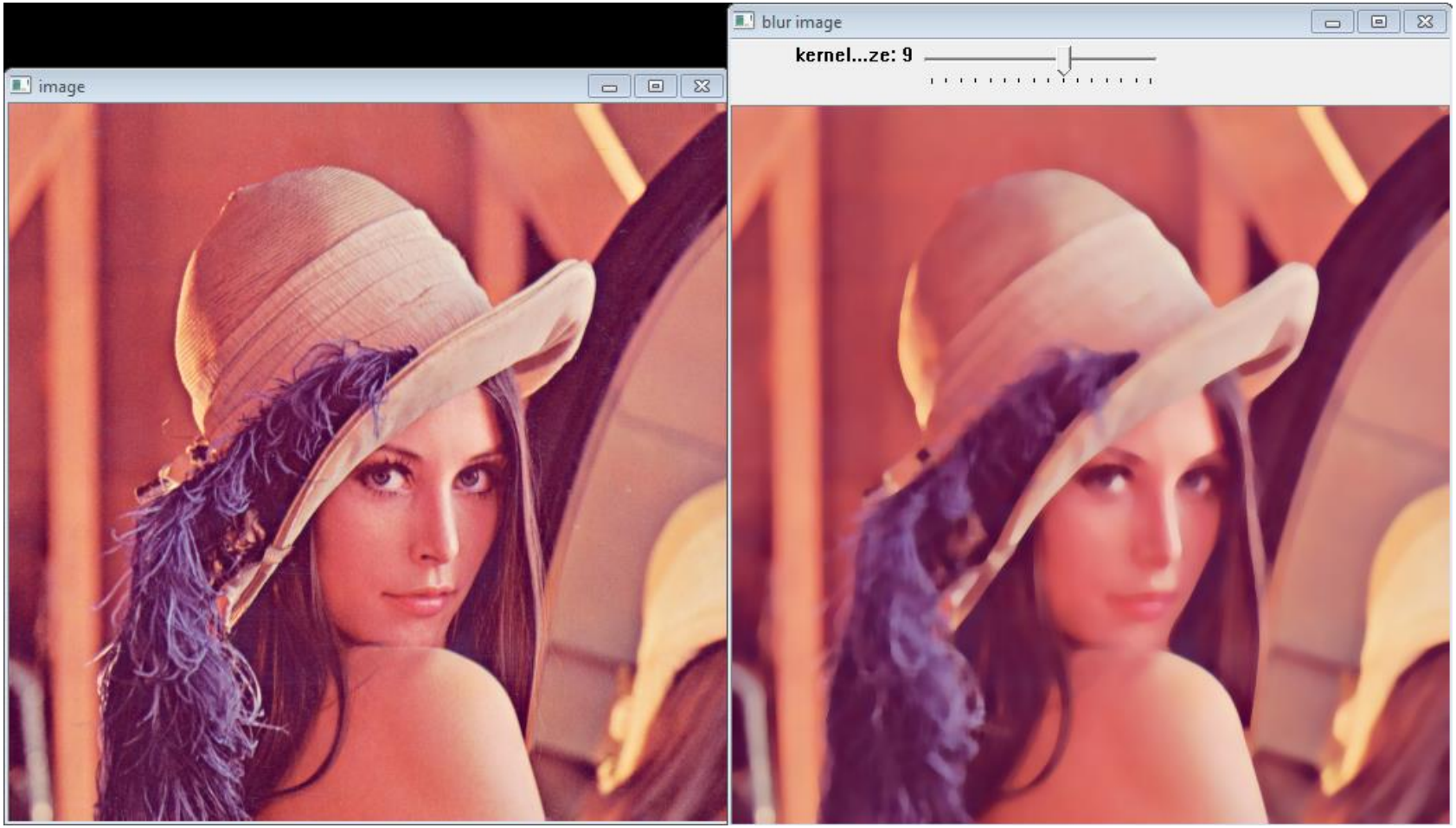
Перевод изображения в градации серого



Размытие изображения

```
#include <opencv2\core.hpp>
#include <opencv2\highgui.hpp>
#include <opencv2\imgproc.hpp>
#include <string.h>
using namespace cv;
int main(int argc, char *argv)
{
    Mat image = imread("Lenna.png");
    Mat blurImage;
    int kernel_size = 3; //размер ядра
    //создаем окно, это необходимо для того, что создать trackbar у окна
    std::string WinName = "blur image";
    namedWindow(WinName);
    //trackbar для изменения размера ядра размыва (чем больше ядро, тем сильней размытие)
    //параметры: название trackbar'a; название окна, к которому он применяется;
    //переменная - размер, ее значение меняется ползунком; максимальное значение
    createTrackbar("kernel size", WinName, &kernel_size, 15);
    char c = 0;
    //выход по нажатию пробела
    while (c != ' ')
    {
        //медианное размытие изображения с нечетным размером ядра
        medianBlur(image, blurImage, kernel_size/2*2 + 1);
        imshow("image", image);
        imshow(WinName, blurImage);
        //ждем 30 миллисекунд и запоминаем код нажатой клавиши
        c = waitKey(30);
    }
    return 0;
}
```

Размытие изображения



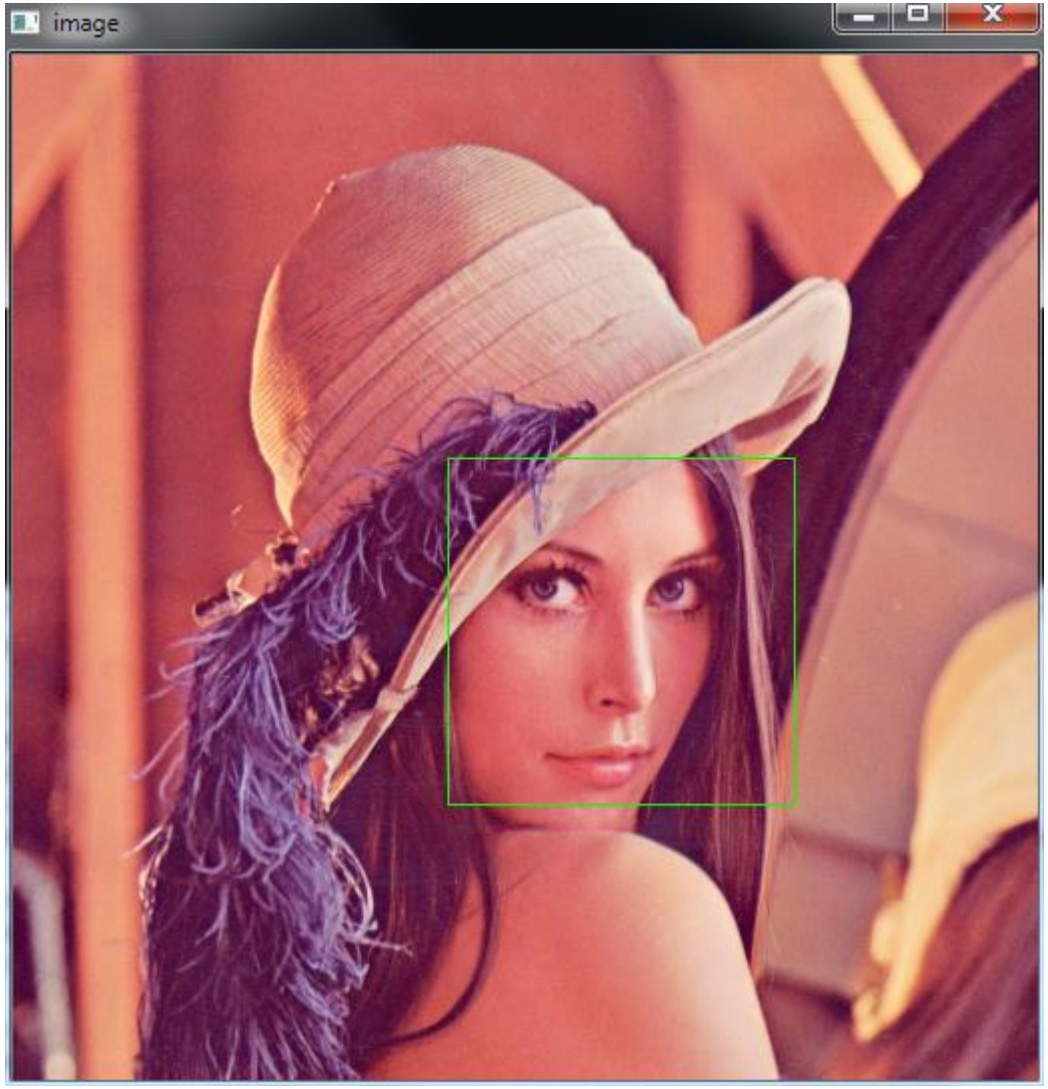
Детектирование лиц

```
#include <opencv2\core.hpp>
#include <opencv2\highgui.hpp>
#include <opencv2\imgproc.hpp>
#include <opencv2\objdetect.hpp> //для детектирования лиц (CascadeClassifier)
#include <vector>

using namespace cv;

int main(int argc, char *argv)
{
    Mat image = imread("Lenna.png");
    std::vector<Rect> faces;
    CascadeClassifier cascade;
    //загружаем каскад для детектирования лиц
    if(cascade.load("C:\\opencv\\sources\\data\\haarcascades\\haarcascade_frontalface_default.xml"))
    {
        //детектируем лица, найденные прямоугольники записываются в вектор faces
        cascade.detectMultiScale(image, faces);
        for (std::vector<Rect>::iterator it = faces.begin() ; it != faces.end(); ++it)
        {
            //рисуем прямоугольник вокруг каждого найденного лица, третий параметр - цвет, три значения - B,G,R
            rectangle(image, *it, Scalar(0,255,0));
        }
        imshow("image", image);
        waitKey();
    }
    return 0;
}
```

Детектирование лиц



Открытие видео

```
#include <opencv2\core.hpp>
#include <opencv2\highgui.hpp>
#include <opencv2\video.hpp>
#include <vector>

using namespace cv;

int main(int argc, char *argv)
{
    VideoCapture cap;
    //открытие видео с веб-камеры, параметр - номер
    //чтобы открыть файл, в качестве параметра нужно указать путь
    cap.open(0);
    if(cap.isOpened())
    {
        //пока можем получить следующий кадр
        while(cap.grab())
        {
            Mat frame;
            //декодируем кадр
            cap.retrieve(frame);
            imshow("video",frame);
            waitKey(30);
        }
    }
    return 0;
}
```

Выделение объекта на видео

```
Mat image;
bool selectObject = false;
int trackObject = 0;
Point origin;
Rect selection;
int main( int argc, const char** argv )
{

    VideoCapture cap;
    cap.open(0);
    namedWindow( "Selection", 0 );
    setMouseCallback( "Selection", onMouse, 0 );
    char c = 0;
    Mat frame;
    while(c != ' ')
    {
        cap >> frame;
        if( frame.empty() )
            break;
        frame.copyTo(image);
        if( selectObject && selection.width > 0 && selection.height > 0 )
        {
            Mat roi(image, selection);
            bitwise_not(roi, roi);
       }
        imshow( "Selection", image );
        c = (char)waitKey(10);
    }
    return 0;
}
```

Выделение объекта на видео

```
static void onMouse( int event, int x, int y, int, void* )
{
    if( selectObject )
    {
        selection.x = MIN(x, origin.x);
        selection.y = MIN(y, origin.y);
        selection.width = std::abs(x - origin.x);
        selection.height = std::abs(y - origin.y);
        selection &= Rect(0, 0, image.cols, image.rows);
    }

    switch( event )
    {
    case EVENT_LBUTTONDOWN:
        origin = Point(x,y);
        selection = Rect(x,y,0,0);
        selectObject = true;
        break;
    case EVENT_LBUTTONUP:
        selectObject = false;
        if( selection.width > 0 && selection.height > 0 )
        {
            trackObject = -1;
            std::cout<<"Our region is "<<selection<<"\n";
        }
        break;
    }
}
```