CodeBlocks as a portable development environment with OpenCV library

This is a quick guide on how to use the CodeBlocs-EP (Education Portable) integrated development environment (IDE) witht the OpenCV library compile with such IDE for portability.

The aim of this project was to have a lightweight and portable integrated development environment with which to develop OpenCV applications in C++. Portable in this context means that the user should be able to have this IDE in a USB memory and be able to run it from there under any (as many as possible) computer with a Windows operating system. The IDE can also be copied to the HDD and run from there.

In this notes I will explain how the user can get up and running in developing OpenCV applications with this IDE. C++ programming language or OpenCV itself are not in the scope of this notes. A degree of understanding by the user on these two topics are presume by me and only a minimal information on these will be discuss here.

First I will like to acknowledge the creator of CodeBlocks-EP which made the possibility of running the IDE as a portable application with whom this project would not have be possible and also to the people at OpenCV for developing the library and their great work maintaining it. Below are links where one can find more information in each of these topics.

CodeBlocks-EP webpage: http://codeblocks.codecutter.org/
OpenCV webpage: http://opencv.org/

These nots are organized as follow. First I will describe the way the IDE and where it can be found, the where the OpenCV library is located and how to add this to a project and finally I will demonstrate how to run a simple project to display a video from feed from a webcam in a console application.

To begin you will have to download the zip file from my webpage at: http://guivi.one/Mario/Portable_Programming.zip Once this is downloaded just unzip this file into any directory on your HDD or a USB memory. Once you have done this you will the following four folders: **CodeBlocks-EP**, **cmake-3.7.1-win64-x64**, **Libraries** and **Projects**. All of them are quite self explanatory. **Cmake-3.7.1-win64-x64** is out of the scope of these notes apart from saying that it was used to generate the OpenCV project for compilation, to learn more about this you can visit http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html.

On the Project directory you will find two example programs. You can try to run them and see what they do and how they are build. To open a CodeBlocks project you will need to open the ".cbp" file with the CodeBlocks-EP.

Under the folder **CodeBlocks-EP** you will find the executable which will run the CodeBlocks IDE. It is important to note that you will find many executables but the one you will want to use to run CodeBlocks-EP is CbLauncher.exe. Once you run this executable the following windows will appear after it finishes loading.

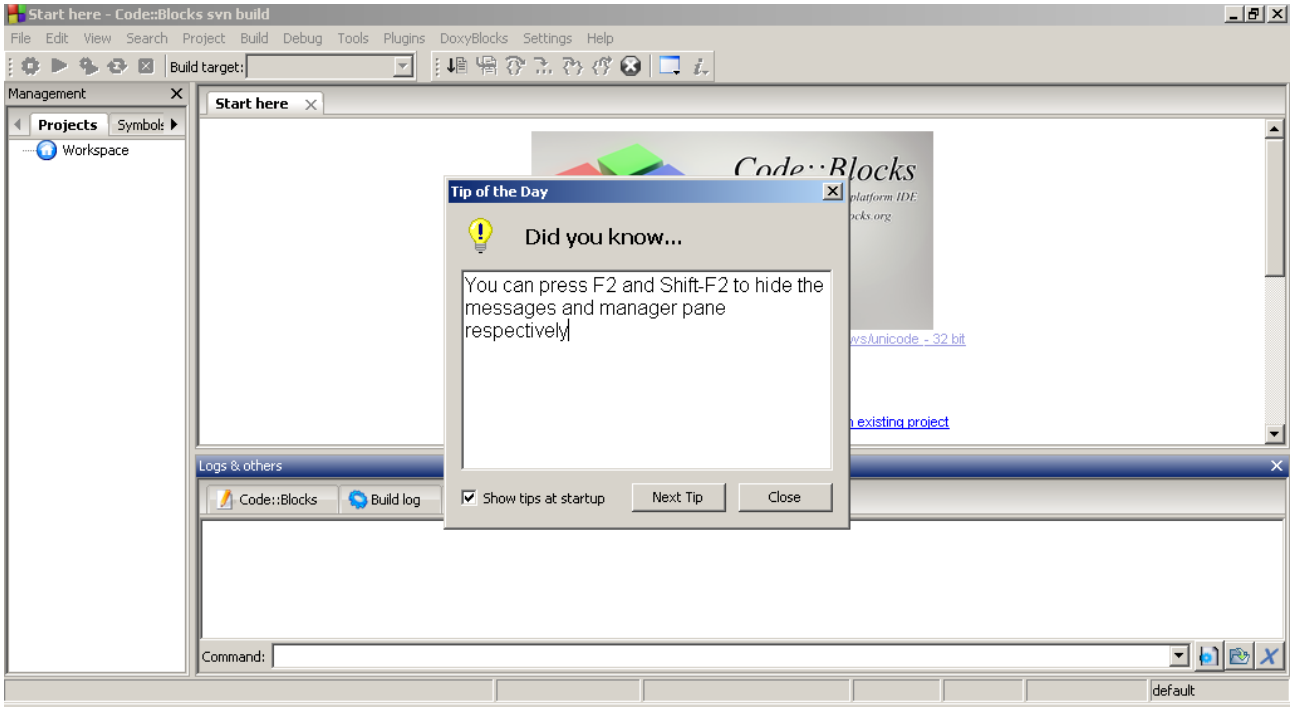

Illustration 1: CodeBlocks IDE

Before going any farther I will also like to mention that under the CodeBlocks-EP folder you can find a folder named MinGW. This is the compiler which CodeBLockes-EP uses, it is based on Linux GNU compile. In this folder one can find all the libraries and header files which come with this compiler by default. As well I will like to point out that this is the 32 bit compiler which should be able to generate executables which will run in 32 and 64 bit PC architectures. It is also the compiler I used to compile the OpenCV library.

Now lets look at the OpenCV library. It is located in a folder named opencv under the Libraries folder. In it you will find three folders and a few files. Of these folders two are of most importance, x86 and include. In the include folder you will find all the header files you will need the IDE to be able to search upon and on the x86 are the binaries and library files which you will need to link the compiler with (this will become clearer when running our first program). Under the folder Libraries/opencv/x86/mingw/ there are three folders: bin, lib, samples. The bin contains the dll files which will have to be deployed with the executable generated as opencv was compiled as a shared library. In the lib folder are the actual library files which for CodeBlocks have an extension “.a”, on the sample folder you will appropriately find sample applications generated when the OpenCV library was compiled.

Now lets look at making our first program. Once the CodeBlocks has opened and you see the window as it seems in the illustration above. The first thing I will recommend is to uncheck the

"show tips at startup" as this can become very irritating. Then go to File→New→Project this will pop the following dialog.

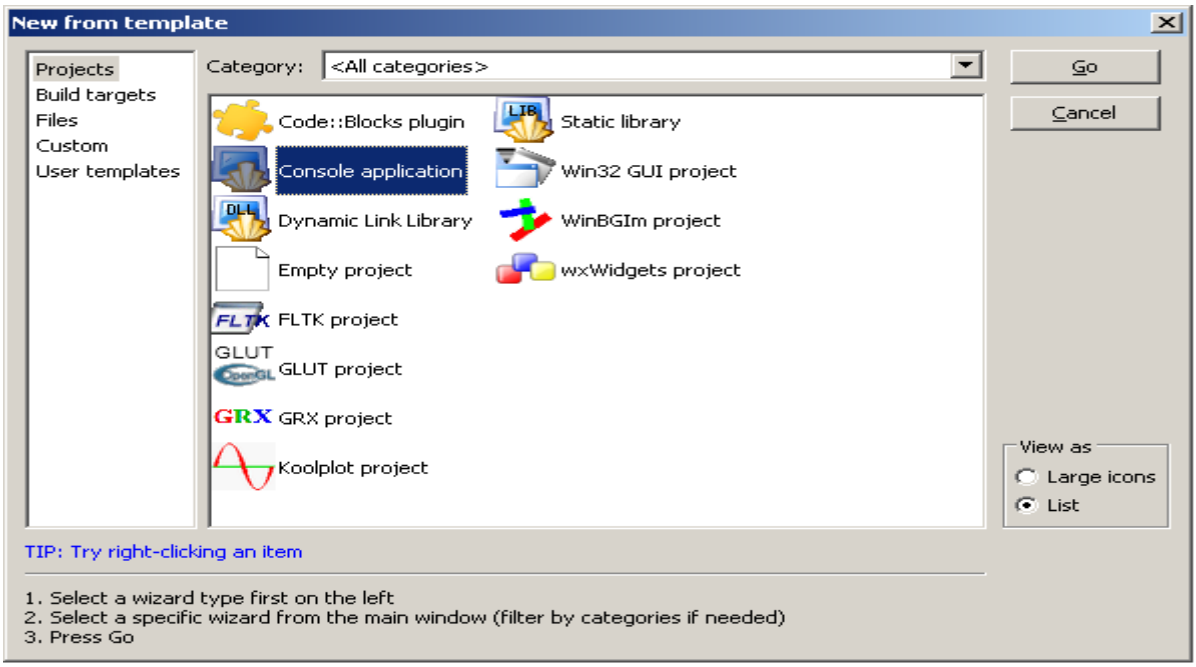

Illustration 2: New File

In this we should select a console application and press go. At this point the wizard process for console application will guide us through the process of creating the console application. The first dialog is just a general one so press Next. Then it will ask us if we want to create a C or a C++ project we will choose C++ for this application as we will be using the C++ classes on OpenCV, then press Next.

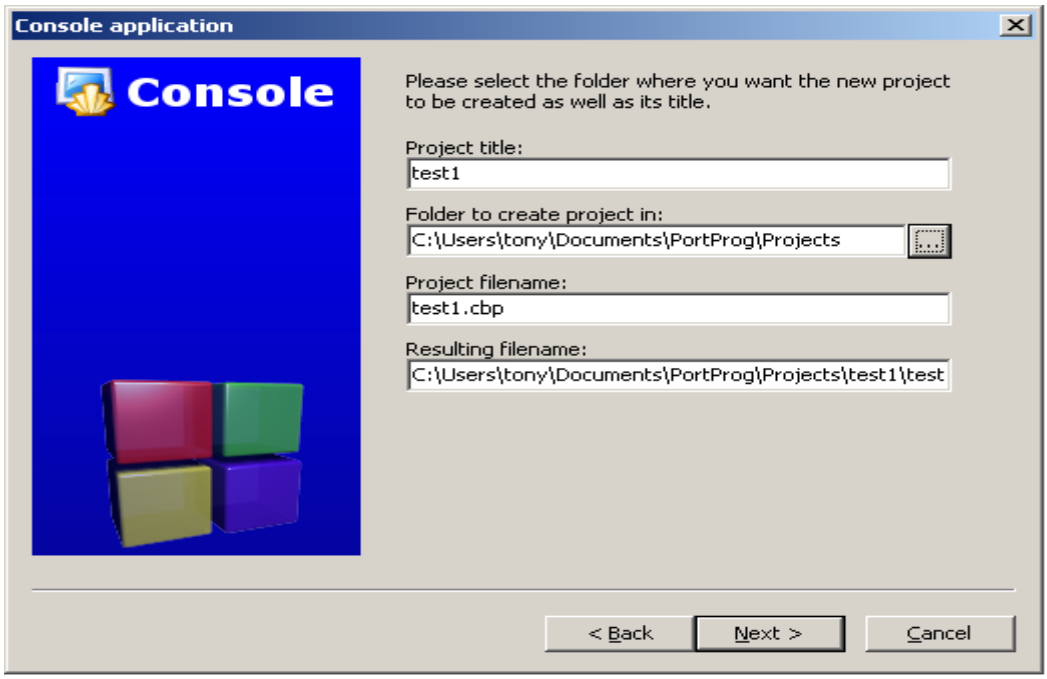

Illustration 3: Name wizard

At this point it will ask us to name our project and to place a location for it. The first time we run this it will have a bad directory on the "Folder to create project in" as it will have the last location

when I run it from a USB before I ziped the folder. I recommend you to use the Project folder that we unzip earlier in this field, once you set it it will save this and every project should be put there by default in the future. We will name this project test1, see that the wizard will create a folder in the Project folder with the project name. Then press Next.

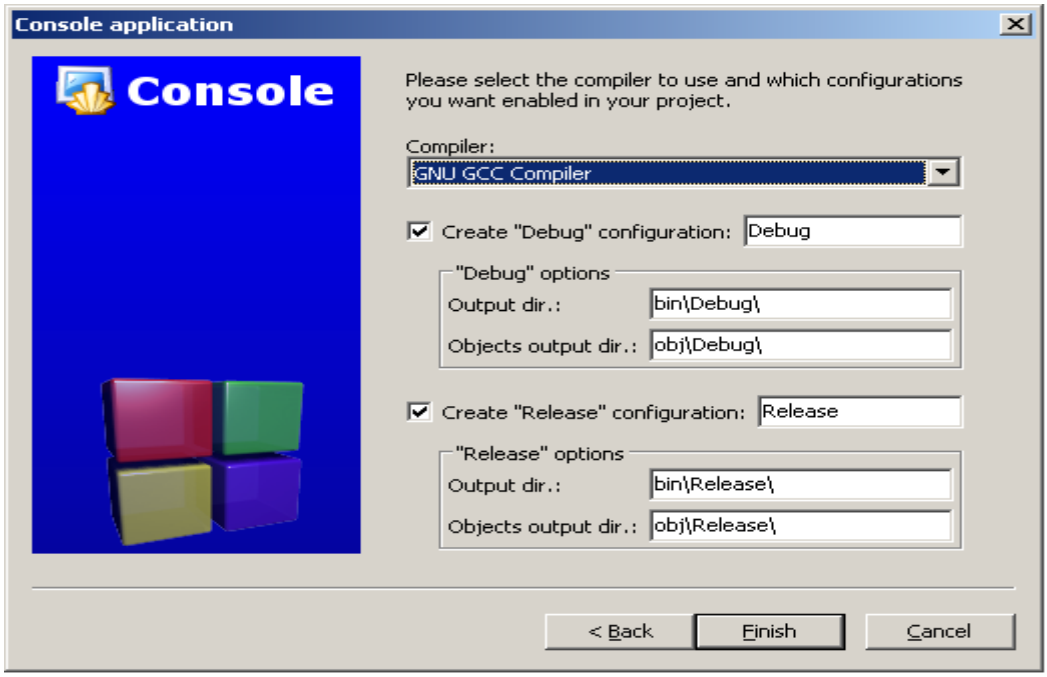

Illustration 4: Debug/Release wizard

The final step is to ask you about which compiler to use and the directories in which to put the Debug and Release versions of the executables and their related files. At this point simply press Finish.

CodeBlocks will create an empty project. To add a file we need to File → New → File.

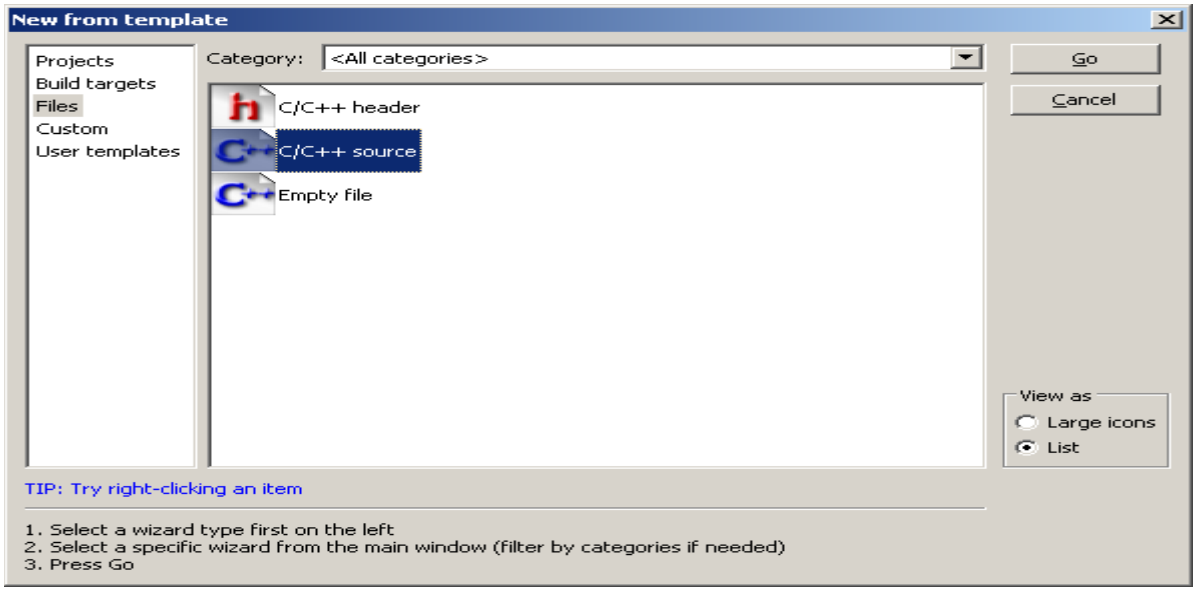

Illustration 5: New file

Select a C/C++ source and press go. Then a wizard process will run again which will help us add the file to the project. Simply press next for the first two dialogs and then on the last one make sure

to check the debug and release box and also the “add file to active project”. Then name the file (name needs the path). For this project we will name the file main.cpp

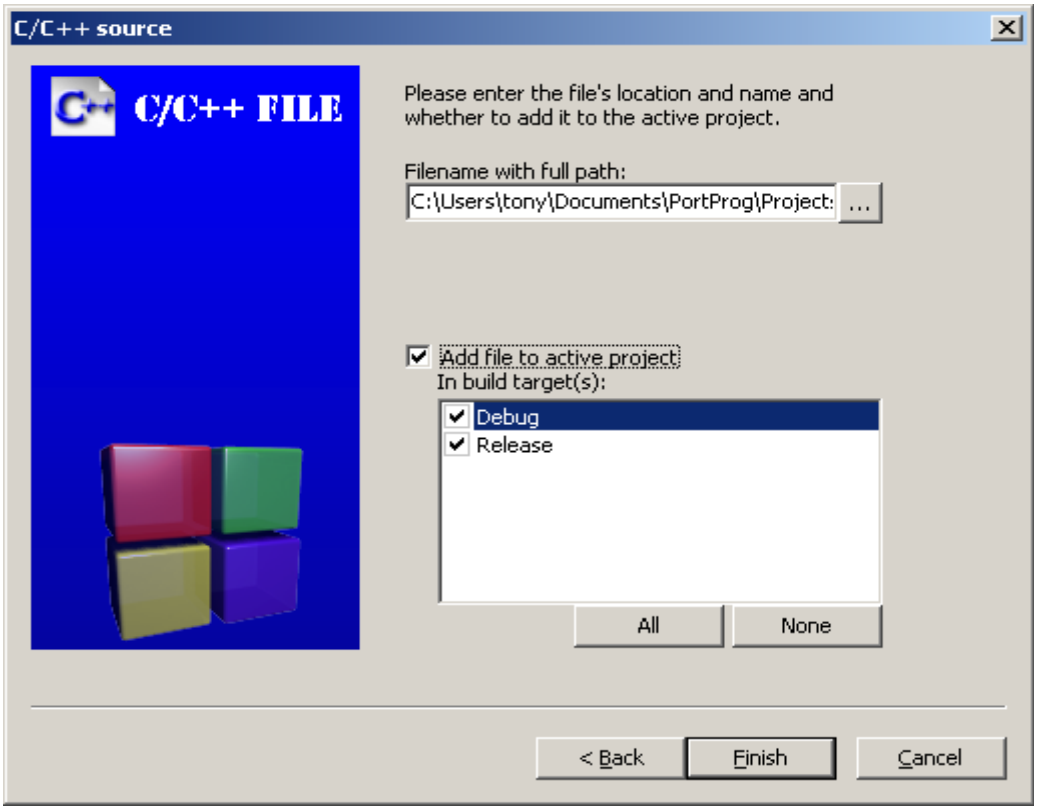

Illustration 6: New file wizard

At this point we can start writing our first OpenCV application in C++.

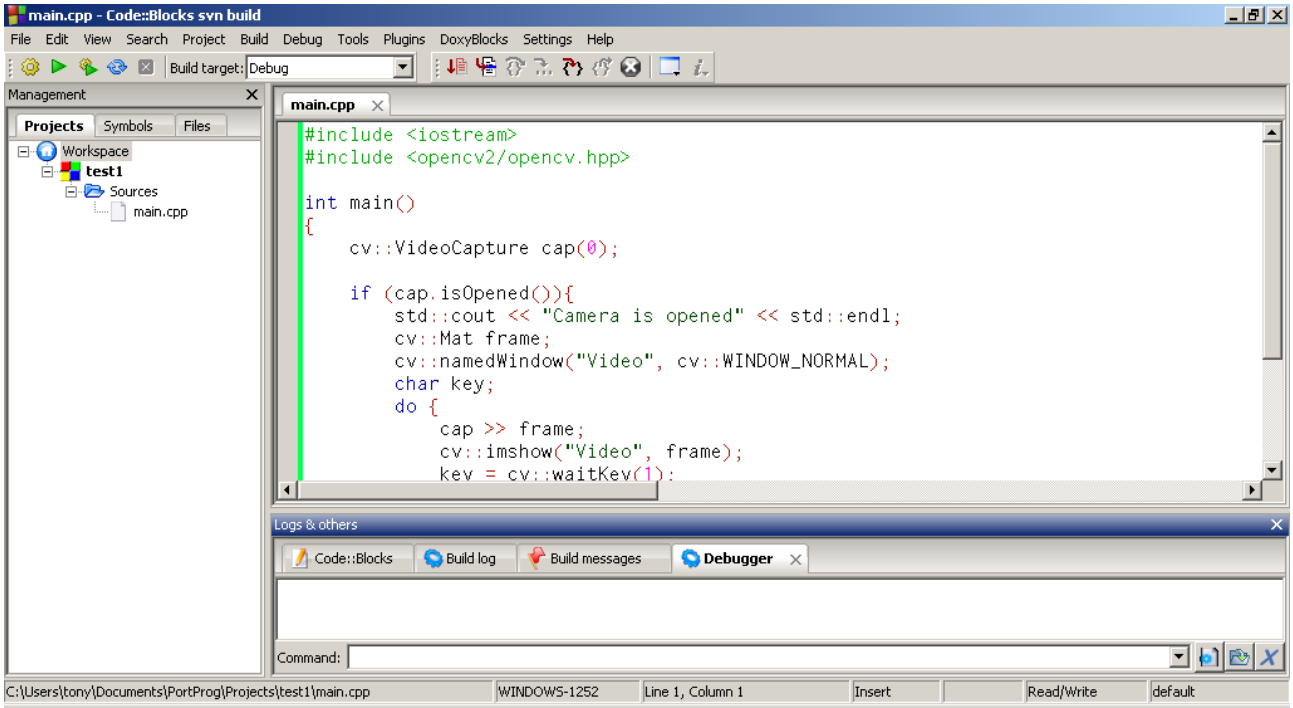

Illustration 7: main.cpp

The code for this first test program is as follow:

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

int main()
{
   cv::VideoCapture cap(0);

   if (cap.isOpened()){
      std::cout << "Camera is opened" << std::endl;
      cv::Mat frame;
      cv::namedWindow("Video", cv::WINDOW_NORMAL);
      char key;
      do {
         cap >> frame;
         cv::imshow("Video", frame);
         key = cv::waitKey(1);
      }while(key != 27);
   }

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

Lets look at the code in more detail. The cv::VideoCapture is a class on OpenCV which as its name indicate it helps us capture video from a webcam (or open from a file). In the case above we are initializing the variable with a 0 to indicate we want it to try to open the first camera it finds in the PC. Then we check if the camera was opened and if so we create a matrix which will hold the frames pulled from the capture class and then this frames are displayed in a window named “Video”. The program will do this until the key “Esc” is pressed by the user. “Esc” key has normally the ASCII value of 27. At this point if you compile by pressing the gear icon on the top left toolbar as shown in the image above everything should compile properly with no errors. But if after compiling you try to run the software (this is done by pressing the second from the left icon that looks like a play button on the tool bar menu) it will complain that it cannot find the dll files for OpenCV.

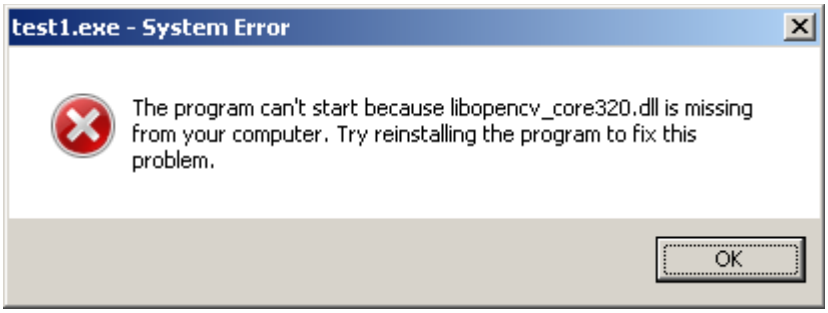

Illustration 8: Error

To fix this you simply have to copy the dlls found on the ..\Libraries\opencv\x86\mingw\bin to the project directory my case is on ..\Projects\test1. If you want to redistribute your executable you will have to redistribute these dlls in the same directory.

At this point you should be able to run the program with no problems.

What is been hidden from you the user.

A C/C++ programmer with experience will have notice that some how you have not needed to indicate to the compiler where the header and library files are. That is because I have already set this in the properties of the IDE. If we look at these by going to settings→compiler and debugger… A dialog will appear which looks as follow:

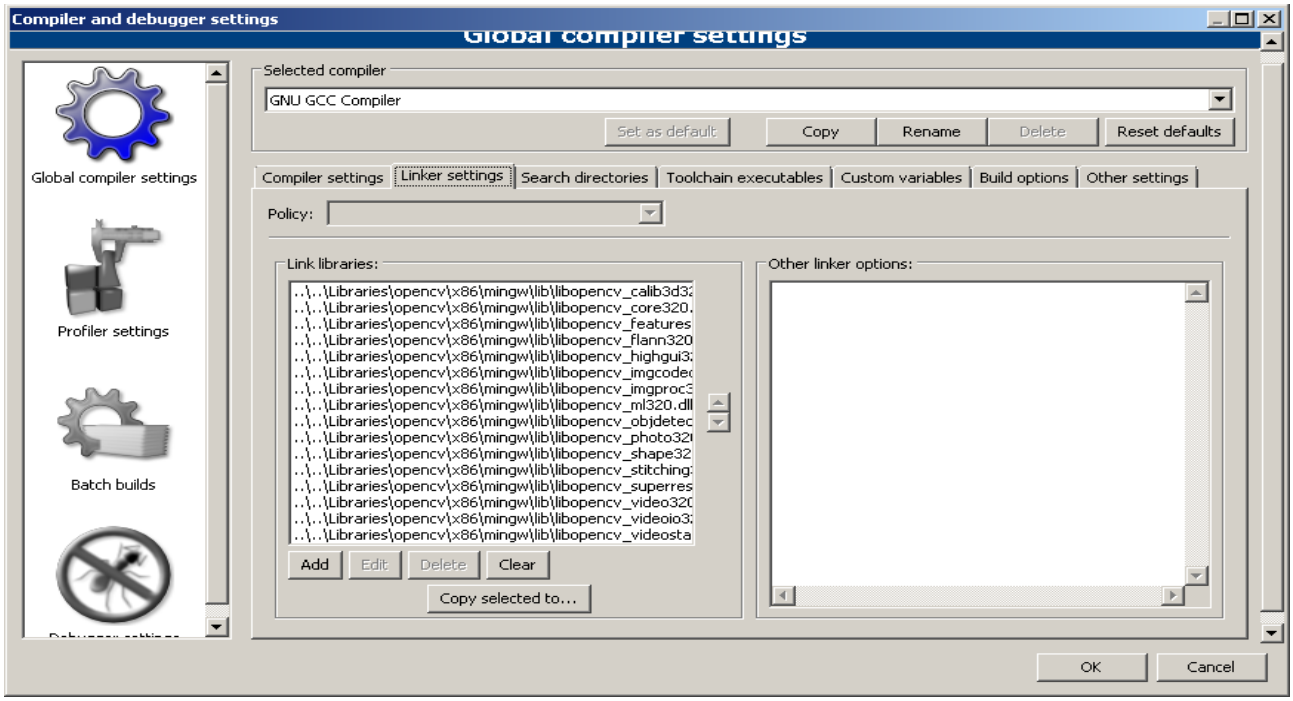

Illustration 9: Linker

Here we can see that I have already linked all the OpenCV libraries to any project that we create from now on in the IDE. Also if we look at the search directories section you will see that I have already added the include directory for OpenCV.

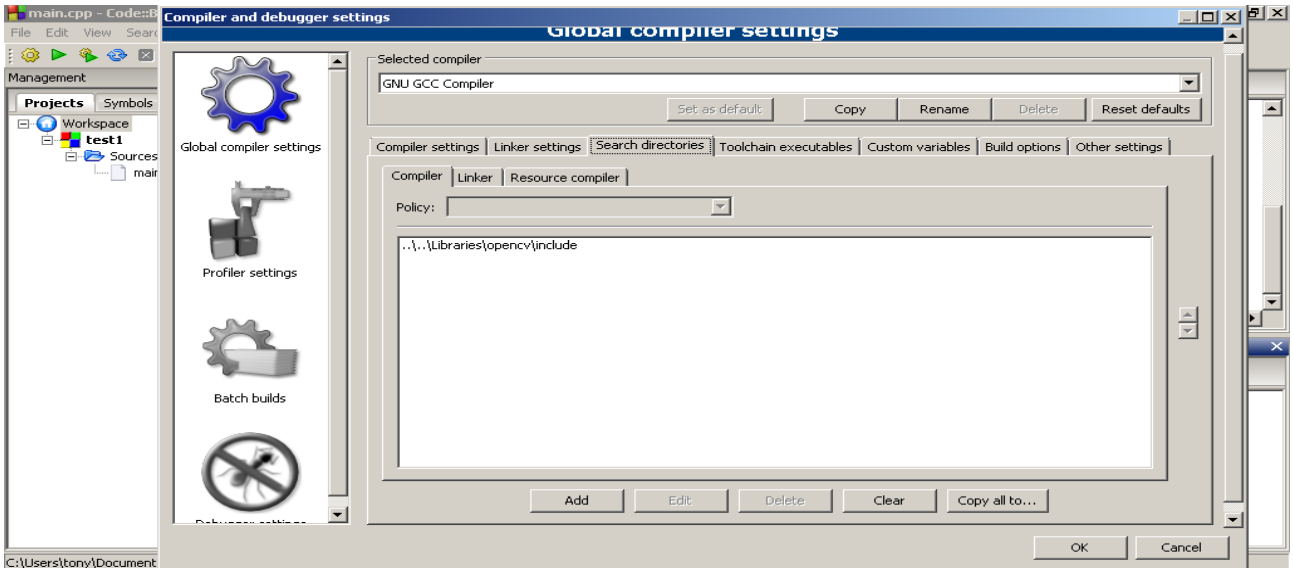

Illustration 10: Search paths

You can see that I added all this files and directory paths as relative paths rather than as absolute paths. This is to keep the IDE portable. I will recommend that if you add any other library to the IDE you add the paths to this as relative paths as to keep the portability of the IDE.