Tutorial: MicMac installation

This document is related to the installation of the software micmac. The following sections give information about the **software dependencies** of micmac, the installation from the **binaries** and the installation and compiling from the **sources**. Additional info are provided about the change of the environment variable "PATH" and about the use of **mercurial**, the software versioning utilized for micmac. More details are provided on the forum devoted to micmac software: http://forum-micmac.forumprod.com/

Prerequisites

Some external tools need to be present on your system for Micmac to run properly:
- make (www.gnu.org/software/make) for parallel processes management.
- convert, d'ImageMagick (www.imagemagick.org) for image format conversion.
- exiftool (www.sno.phy.queensu.ca/~phil/exiftool) et exiv2 (www.exiv2.org) to read/write image meta-data.
- proj4 (http://trac.osgeo.org/proj/) for coordinate system conversion.

You have to install these tools before installing micmac. Since Windows does not have an easy-to-use package manager, a version of make, convert, exiftool and exiv2 are delivered with the source and Windows binaries archives. They are placed in the “binaire-aux” directory.
Here is the command line for the installation of dependencies under linux OS:
sudo apt-get install make imagemagick exiv2
more info about exiftool:
http://www.sno.phy.queensu.ca/~phil/exiftool/install.html

You can check before-hand that Micmac is able to find those programs by calling the following command from the micmac directory:

bin/mm3d CheckDependencies

'NOT FOUND' near one of the tools indicates either the specified executable is not on your disk or it cannot be found in the directories of the PATH environment variable. There is also a special directory for tool finding which is named 'binaire-aux', in Micmac directory. When an

external program is required, this directory is always scanned whatever the value of PATH.

Additional notes for Windows

You will need Visual C++ 2010 runtime redistribuables to run pre-compiled binaries of micmac (http://www.microsoft.com/fr-fr/download/details.aspx?id=5555). Both pre-compiled and compiled from source executables will require :
-Visual C++ 2005 runtime redistribuables (http://www.microsoft.com/fr-fr/download/details.aspx?id=3387)
-DOTNET (.Net) Framework 2.0 (http://www.microsoft.com/fr-fr/download/details.aspx?id=1639)

One of WINDIR or SystemRoot environment variable must be set to Windows' installation directory (“C:\Windows” in much cases). This **prevents Micmac from calling a convert.exe that is not ImageMagick's convert tool** but a network utility used by the system.

Compiling from the sources' archive

In addition of previously named tools, people willing to compile binaries from the source code will need to install the cmake program (www.cmake.org). Linux and MacOS X users may also want to get X11 header files in order to generate graphical functionalities like SaisieBasc, SaisieMasque, etc.

The package of X11 headers is general called 'libx11-dev' under Linux distributions. X11-based tools are not available in the Windows version. Here is the command line for installing the X11 package:

sudo apt-get install libx11-dev

Compiling process for Linux / MacOS X

One may compile the sources downloaded from the internet site. The following workflow explains step by step this installation. One may also prefer the versionning software mercurial, which enable to update easily the micmac software. As micmac is evolving quite rapidly, We advise you to use mercurial for its installation and update (see section " Using Mercurial as revision control system").

- extract source files from the tarball :
tar xvf micmac_source_revXXX.tar.gz
- access the 'micmac' directory :
cd micmac
- create a directory for the build's intermediate files, then access it :
mkdir build
cd build
- generate makefiles using cmake
cmake ../
- process compilation :
make install -j"cores number" (ex: 'make install -j4')

Using Mercurial as revision control system

-install mercurial

sudo apt-get install mercurial

-Upload the source file and store them in the directory micmac:

hg clone https://geoportail.forge.ign.fr/hg/culture3d **micmac**

login culture3d
pswd culture3d

-Compile source code

-Access the micmac directory
cd micmac
- create a directory for the build's intermediate files, then access it:
mkdir build
cd build
- run makefile generation:
cmake ../
- run compilation:
make install -jK where K="number of processors" (ex: make install -j8)

Update with mercurial

for updating , go on installation directory (cd micmac) and run:

hg pull

hg update

Then, compile the sources codes (i.e "make install" under build, see "- run compilation" above)

For displaying which mercurial version is installed, one may use either the command "mm3d CheckDependencies" or the following command launched from the directory micmac:

```
 hg log -r tip --template "{rev}\n"
```

Compiling process for Visual C++ (Windows)

The first steps are the same as for a Linux/MacOS build except for the 'make' call (process compilation).
Instead of makefiles, Cmake generates a Visual C++ solution, named 'micmac.sln'. Open it and compile the 'INSTALL' project (see figure 1). Be sure to be in "Release" configuration (see figure 1), for Micmac is much faster built this way than in 'Debug' mode. Again, do not compile the entire solution but just the 'INSTALL' project, otherwise compiled binaries won't be copied in the 'bin' directory and this will prevent Micmac from working.

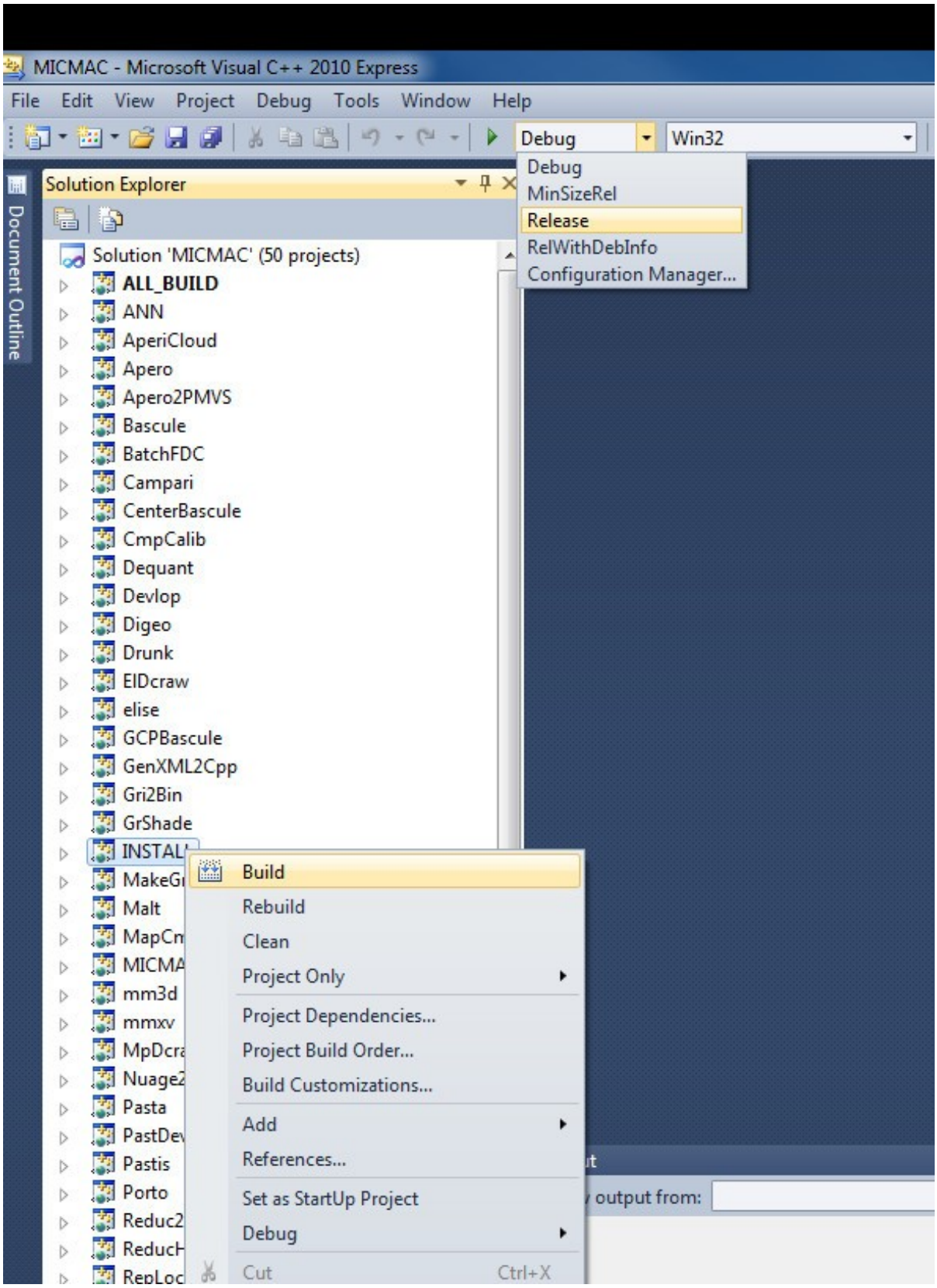

Figure 1: compile micmac on windows platform.

How to test if micmac installation has successed

The website logiciels.ign.fr (section image->Micmac->Téléchargement) also provides a test dataset called Boudha_dataset.zip.
This file contains images and configuration files needed to compute the 'Boudha' example from Micmac's documentation. By calling the script this way:

./boudha_test.sh MY_MICMAC_BIN/

assuming your working directory is the 'Boudha' directory contained in the file, you can process all the tool-chain until dense point matching. MY_MICMAC_BIN ' is a path to the 'bin' directory of your installation.

ex.: ./boudha_test.sh ../micmac/bin/

This example assumes 'Boudha' directory (containing data) and 'micmac' (installation directory) have the same parent directory. Notice the ending '/', it's mandatory for the script to work.
After some computation time, you may find three 'ply' files in the 'MEC-6-Im' directory with the three parts of the dense points cloud of the statue's head. Open the PLY files with a viewer like meshlab to check everything proceeded correctly.

additional notes

You can append the full path of the "bin" directory to "PATH" environment variable to call Micmac commands from anywhere. However, it is not necessary to add the "binaire-aux" directory to the PATH variable.

Here is a link toward more details on how to modify the PATH environment variable on a linux OS:
http://unix.stackexchange.com/questions/26047/how-to-correctly-add-a-path-to-path
Here for a Windows OS:
http://www.computerhope.com/issues/ch000549.htm

Compiling with Qt tools

To get the Qt tools (visual interfaces vMalt, vTapas etc., SaisieMasqQT, SaisieAppuisInitQT, SaisieAppuisPredicQT, SaisieBascQT, SaisieCylQT), you need to install Qt from http://www.qt.io/download/

For example, you may download: qt-opensource-linux-x64-5.4.1.run

Then installing Qt, for Linux:

sudo chmod +x qt-opensource-linux-x64-5.4.1.run
sudo ./qt-opensource-linux-x64-5.4.1.run

Ubuntu 12.04 LTS is only shipping v 2.8.7 of *cmake* with apt-get, which is not compatible with Qt5. You should download CMake 2.8.9 (or later) sources, and compile from sources.

For Ubuntu 14.04 LTS, the cmake package is 2.8.12, and everything is fine.

Edit .bashrc : gedit .bashrc and add path to Qt library:
--export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/user/PathToQt5/gcc_64/lib/

Run cmake with option WITH_QT5=ON :

cd build
cmake ../ WITH_QT5=ON

- run compilation:
make install -jK where K="number of processors"