GigaVoxels Install

Requirements

GigaVoxels requires some dependencies. On Linux Ubuntu, all are not available with “apt-get install” command.

CUDA Library

I think Ubuntu has a CUDA package available with an “apt-get install” command. The actual version of CUDA is 6.5, available there:
https://developer.nvidia.com/cuda-downloads
But if you already have installed the 6.0 it works. I have not yet tested the 7 Release Candidate for developers.
NOTE: you have to install the “SAMPLES” cause we are using one file inside… On my PC its located in: /usr/local/cuda-6.0/samples

CMake

Check if you have CMake already installed with the following command:

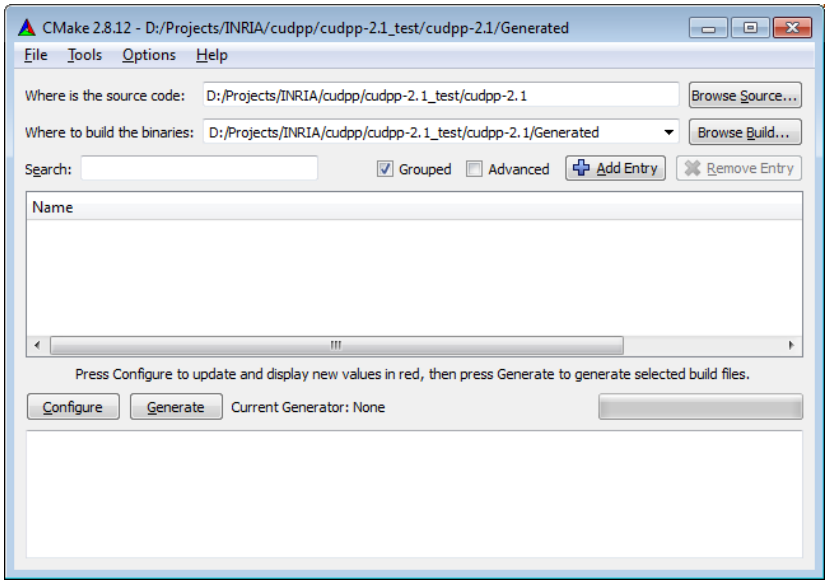
```
dpkg -l | grep cmake
```

If not, you can install it with:

```
sudo apt-get install -y cmake
```

In case of problem, the sources are available there:
http://www.cmake.org/download/

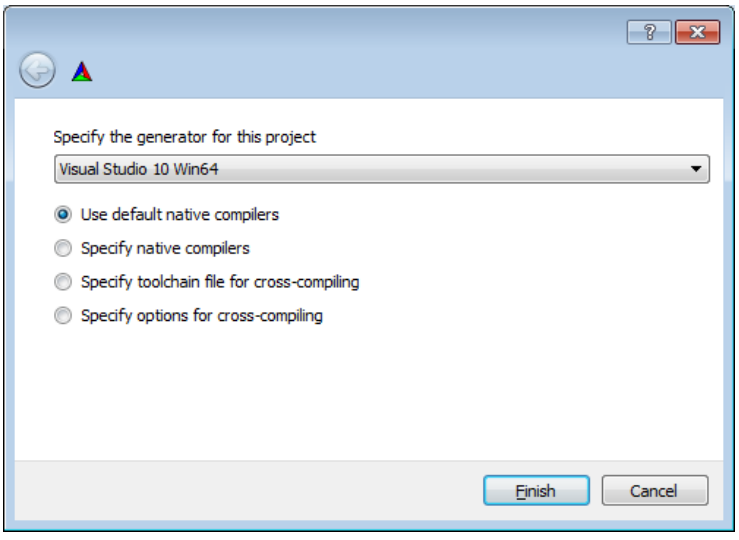
The actual version of CUDPP is 2.2. Sadly, there are no binaries available on Linux, so you have to compile it on your own. Sources are available there:
http://cudpp.github.io/
You need to launch the "cmake-gui" executable available in "CMake". It has a nice user interface.



You just need to fill two directories:

- where is the root directory of CUDPP
- where you want to create the Project Environment for compilation (i.e. makefiles)

Click on "configure" button (bottom left). CMake will ask you to choose the Generator used to produce your project environment (Visual Studio 2010, Unix Makefiles, etc…). On Linux, you should see : "Unix Makefiles" and choose the "Use default native compilers".



Click on Finish.

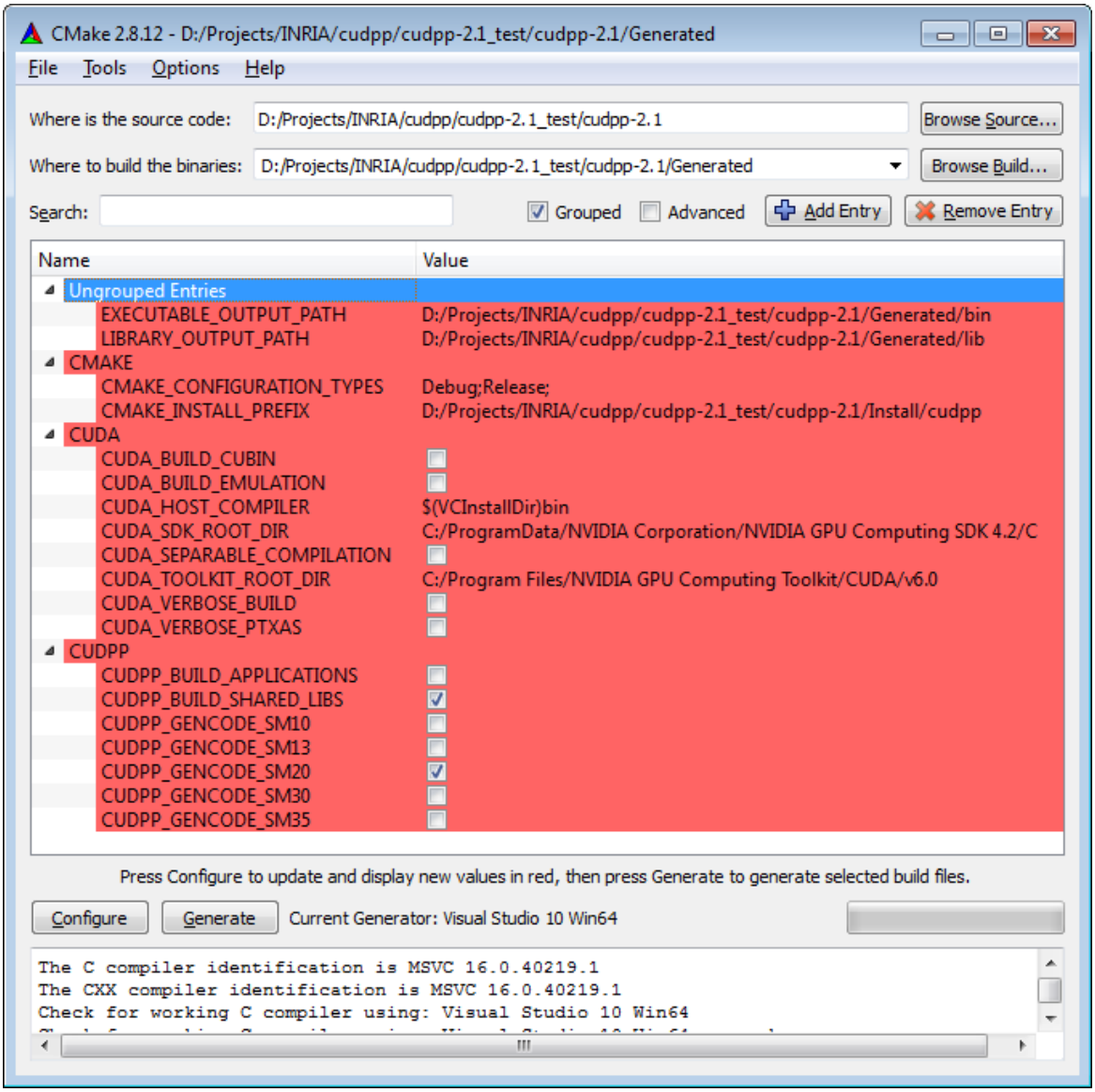
Now, CMake will propose you to modify some internal variables (in RED). Here are my example on Windows.

I have modified:

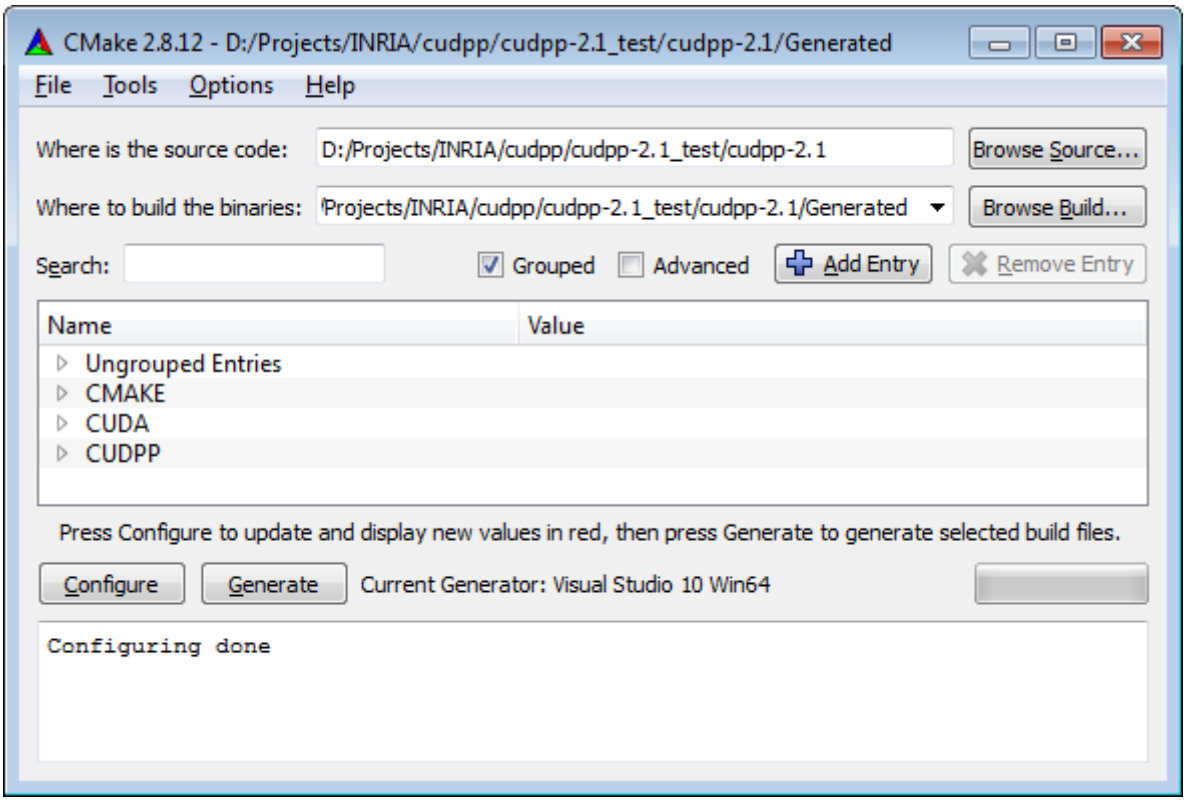
- CUDPP_GENCODE_SM20: you need to select the GPU “compute capability” (for your GTX660, I think it’s a Kepler 3.0, search on Internet)
- check CUDPP_BUILD_SHARED_LIBS
- CMAKE_CONFIGURATION_TYPES : I have choosed “Debug;Release”
- CMAKE_INSTALL_PREFIX: on Windows, I have modified the path used to copy and install the library (include, lib, etc…). On Linux, it could be /usr/local as you want.

Notes:

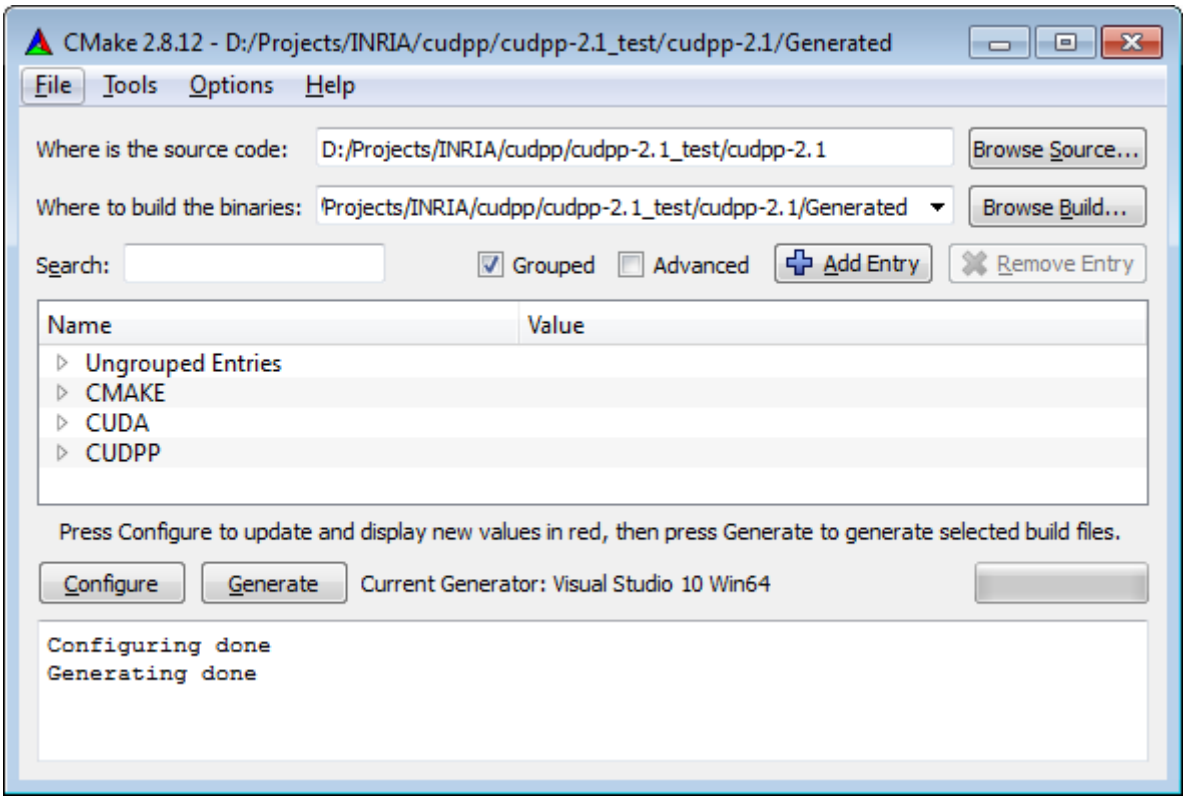
- CUDA_TOOLKIT_ROOT_DIR: should have been found and filled automatically by CMake and points on your CUDA installation directory
- CUDA_SDK_ROOT_DIR is not used…



Click you “Configure” button again. You should have nothing in RED:



OK, then click on “Generate” button (bottom left). You can see a “Generating Done” in the log window:



OK, it’s finish for the CMake configuration.

CUDPP Compilation

Now you have to compile CUDPP. Just go to the directory you have previously indicated in: “Where to build the binaries”, then launch a command like:

“make” or “make -jN” (with N, the number of processors used to compile in parallel)

When it’s finished, type:

“make install”

It will copy the generated library in the directory you have previously indicated in “CMAKE_INSTALL_PREFIX”

NOTE: I think you will need to add one file to the install. There is a bug in CUDPP script, so:

- copy the file “cuda_config.h” in your INSTALL/include directory

Other dependencies

Just in case of :

Installing p7zip for External archive decompression:

```
dpkg -l | grep p7zip-full
sudo apt-get install -y p7zip-full
```

Installing unzip for Common archive decompression:

```
dpkg -l | grep unzip
sudo apt-get install -y unzip
```

Dependencies:

Finally, to the same with all the following dependencies:

```
libqglviewer-dev
libmagick++-dev
freeglut3-dev
libqt4-dev
libtinyxml-dev
libqwt-dev
cimg-dev
libassimp-dev
libglm-dev
```