Preparing a MSWindows installer of DAMQT with Inno Setup

Rafael López*

July 11, 2017

1 Preparing software

The following software must be installed on a machine with MSWindows©:

1. Qt library including tools. In particular, including Qt creator and mingw32 features.
2. Mingw32© and MSYS©.
3. Inno Setup©.
4. Cmake©.
5. DAMQT package.

Figure 1 Shows a typical installation of Qt and MinGW.

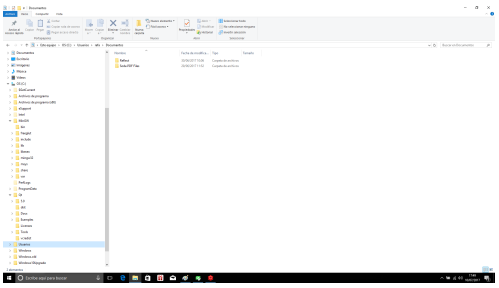

Figure 1: Software for MSWindows installer of DAMQT

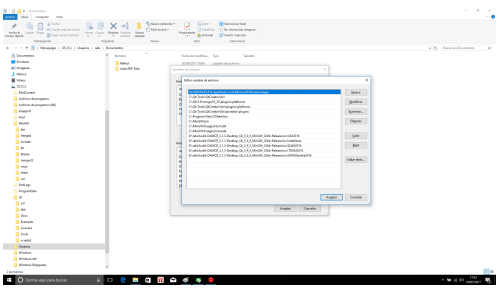

Figure 2: `PATH` settings for Qt creator

To get the Qt creator working properly, some directories must be accessible. This can be achieved, for instance, by adding them to the environment variable `PATH`. The `PATH` settings shown in fig 2 are sufficient to do the job.

DAMQT package can be installed by unzipping the corresponding tarball (which is delivered compressed with `gzip`). This can be done with any suitable tool or, alternatively, with Mingw32. In this case, open a MSYS console (which will ba available in the

*Universidad Autónoma de Madrid, Facultad de Ciencias. Departamento de Química Física Aplicada.

`MinGW/msys/1.0` folder), navigate to the place where you want to deploy the package and run the command:

```
tar vxzf DAMQTtarball.gz
```

where `DAMQTtarball` stands for the name of the tarball to be deployed including its full path. The package will be deployed in a folder named `DAMQT_2.1.3`.

2 Bulding the project with Qt creator

Once the software is installed, the corresponding executables can be generated using Qt creator. To do this, launch Qt creator and you will get the *Welcome* page, similar to that shown in fig 3.

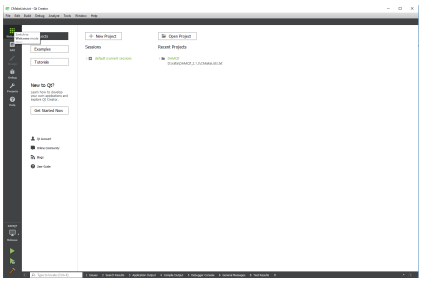

Figure 3: Qt creator *Welcome* page

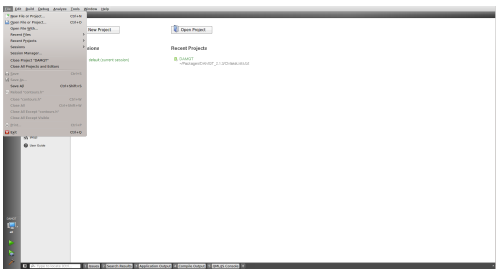

Figure 4: *File* menu

Within Qt creator, the DAMQT project can be built using `cmake`. For this purpose, click on the upper left option *File* in the toolbar and a menu will be displayed as shown in fig 4. Push the option *Open File or Project* and a navigator will appear (see fig 5). Navigate to the home directory of the application and select the *CMakeLists.txt*. Selecting a *CMakeLists.txt* file triggers `cmake` utility. Running `cmake` directly, it would complain of undefined environment variables. It also may eventually complain that `cmake` is not available. For these reasons, it is better to decline building the project for the moment and to make some settings before, using the *Projects* option in the left vertical menu. Pressing on it, a screen appears like that shown in fig 6.

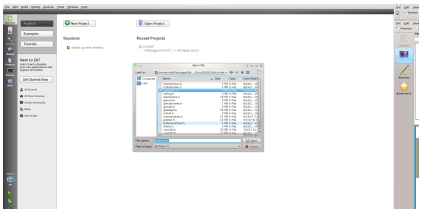

Figure 5:

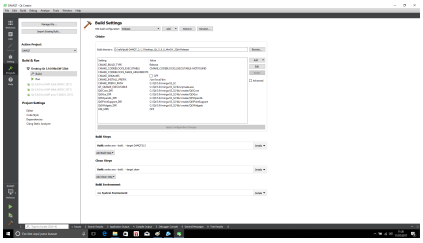

Figure 6:

In this screen, a suitable directory for building the project should be chosen in the box

labeled *Build directory*. A temptative name is proposed, but it can be changed to any other of your preference. Below that box, another one is opened containing the current settings of some environment variables. Checking the box labeled as *Advanced*, all the meaningful environment variables will appear. In figs 7 to 12, suitable settings of the variables are shown, according to the installation proposed.

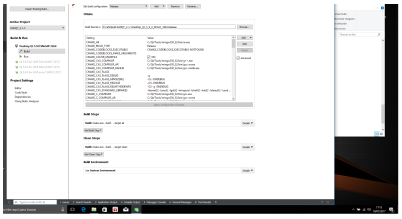

Figure 7:

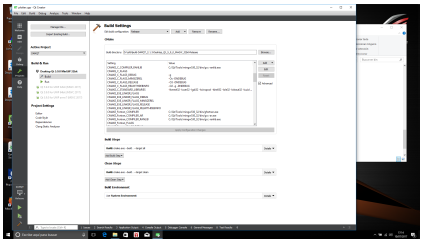

Figure 8:

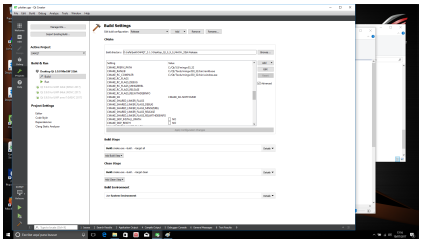

Figure 9:

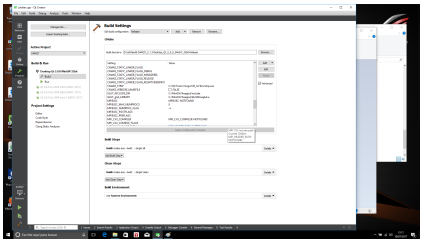

Figure 10:

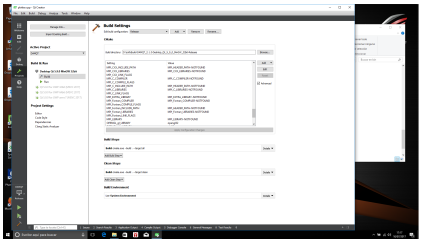

Figure 11:

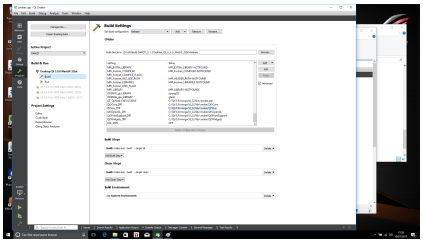

Figure 12:

Next, pressing in the *Details* box on the right of *Build steps* option, a list of possible targets will appear, as fig 13 shows. By default, the *all* option is checked. Uncheck it and check the box corresponding to DAMQT213. Furthermore, *Details* on *Build Environment* option will show the environment pertaining variables. Check that they are correctly set. In particular, check that the `PATH` variable contains the paths to the folders commented above.

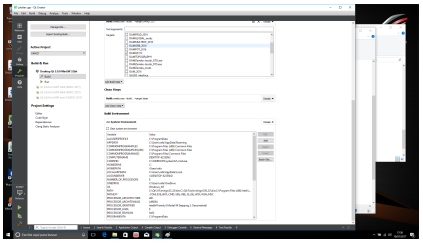

Figure 13:

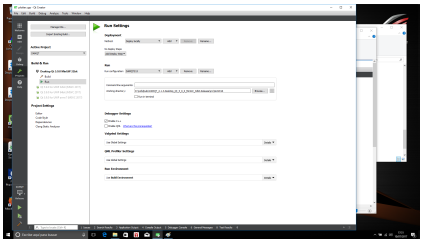

Figure 14:

Selecting the option *Run* appearing on te *Build and Run* entry on the left, a screen will be displayed (see fig 14) where the *Working directory* can be selected among other options.

Pressing in the upper left option *Manage kits*, an *Options* menu is displayed where important installation settings are available. As it can be seen in fig 15 there is a menu on the left in which the entry *Build and Run* is very important. Pressing it, the screen gives access to several tabs which deserve comment.

Tab labeled as *General* allows us general customization. Default settings are fine, but you can change them at will.

Kits tab (see fig 16) is important. It lists the different possibilities of project deployment. *Desktop Qt 5.9.0 MinGW 32 bits (default)* shoud be marked as currently chosen since we want to build the project with MinGW tools.

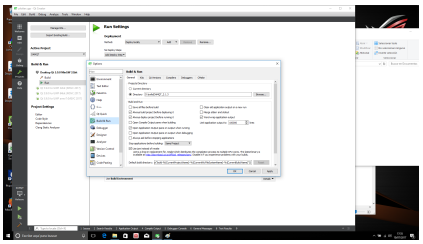

Figure 15:

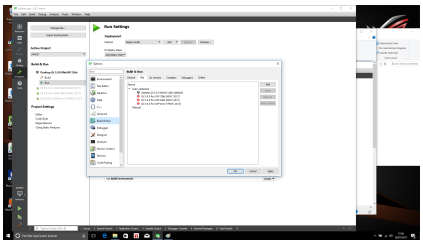

Figure 16:

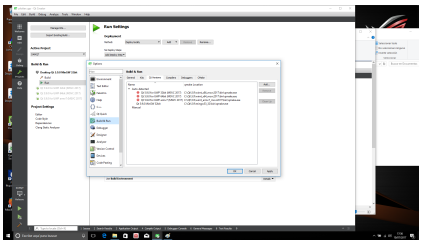

Figure 17:

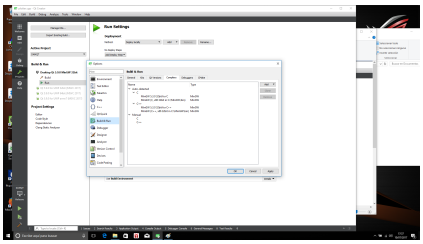

Figure 18:

Qt versions is also important. Again Qt 5.9.0 MinGW 32 bits must be chosen (see fig 17).

Compilers. The system should detect MinGW C and C++ compilers. Otherwise, they

can be added by hand (see fig 18).

Debuggers. The `gdb` debugger may be auto-detected (see fig 19). This is only important if a version of the package is to be built for debugging. In this case, the build command should be run with the option `-DCMAKE_BUILD_TYPE=Debug`.

CMake. This is utmost important (see fig 20). If the system does not detect `cmake`, it must be allocated manually.

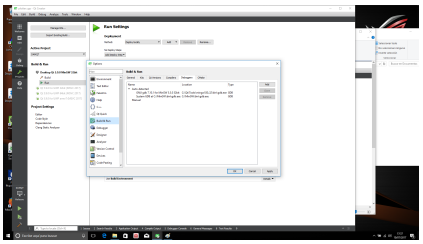

Figure 19:

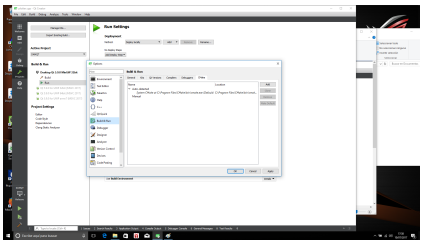

Figure 20:

3 Editing files, building and running with Qt creator

Project files can be edited by choosing the *Edit* option in the upper left menu of Qt creator. This editor is particularly useful for c++ files, since it carries out dynamical checking of the code. (See fig 21).

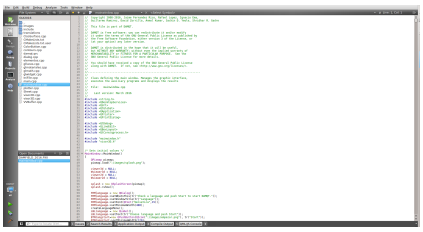

Figure 21:

The project can be run by pressing the green triangle in the lower left menu. If any file has been changed since last run, it asks to save changes before building and running.

The debug mode can be launched by pressing on the green triangle with the ladybird. It only works if the project has been built in debugging mode. Otherwise, the project will be run without debugging options.

Finally, pressing the hammer icon on the lower left menu, the project is built without running.

A green bar shows progress in project building. The bar turns into red if severe errors are detected. Several output screens can be displayed by choosing in the lower menu.

Once the project has been built, it is important to test that it works. This should be made before the project is ready to create the installer.

4 Creating MSWindows installer with Inno Setup

Once the project has been tested and found to work, a MSWindows installer can be prepared using Inno Setup. The DAMQT package comes with a `.iss` file in the *windows* directory. This file can be loaded into Inno Setup and modified according to the current installation.

It is not necessary to know all the subtleties of Inno Setup to make the installer. In fact, all the required settings are present in the `.iss` file. Before running Inno Setup, it is important to check the suitability of the definitions of environment variables quoted in the top of the `.iss` file (see fig 22).

This suitability can be seen by looking at the `[Files]` section, where these variables are used as paths to the different files to be loaded in the installer. Some files refer to the executable files in DAMQT, and other to the libraries to be loaded. Check that the quoted files exist in the folders pointed by the pertaining variables. Otherwise, seek in your installation for the files and redefine the variables to point to the folders where they are contained (see fig 23).

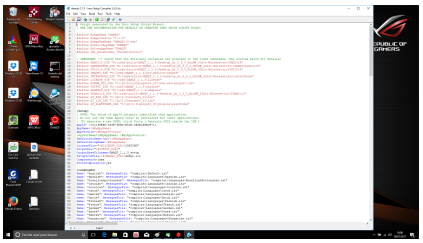

Figure 22:

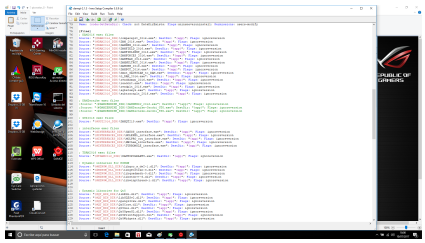

Figure 23:

In some cases, different libraries with the same name may appear in the installation. Choose those that are compatible with your building choices. As a rule of thumb, libraries corresponding to MinGW are preferable to others with equal names when available.

To generate the installer, build it with the corresponding option in Inno Setup Compiler and the run it. You may attempt to run it directly and it will make first the building step and, if it succeeds in building, then will run to create the installer. This will be a file named `DAMQT_2.1.3_setup.exe`, ready to be run in a MSWindows environment.

The installer has been tested in windows 7, 8 and 10, and it seems to work. However, the 3D viewer fails in the windows 7 version.