Installing Madym

# Prebuilt binaries

I have built binaries for Windows, Linux and MacOs. These have not been exhaustively tested across different operating systems, but if what I’ve built works on your system and you just want to use the programs rather than contribute to the code, please do so. However if these binaries don’t work for you, or you just love building things from source, follow the instructions in the next section.

## Windows

Currently these are in the form of a zip file containing the executables and any DLLs they require. Simply copy to someone sensible on your computer and unzip. In time I might package this up as a proper self-installer.

## Linux

Try this tar archive

## Mac

# Building from source

Building Madym from source should in theory be pretty easy – it uses standard C++11 code and has only 2 external dependencies, boost (requiring the filesystem and system libraries) and VXL (which it currently uses for VUL’s argument parsing – this could probably be rewritten out of the code easily enough, although maintaining a link to VXL makes it easy to extend if we want to add any of the fancy imaging capabilities included in VXL). It uses CMake to find and link everything, again making life (in theory) easy.

That said, I have run into problems across different operating systems, all of which a variations on the same problem – making sure all three components (Madym, VXL and boost) are built with the same (or at least compatible) binaries.

Regardless of your operating system, get the Madym and VXL source code:

* git clone <https://github.com/vxl/vxl.git>
* git clone <https://gitlab.com/manchester_qbi/cxx.git>

You can put these wherever you like. Now follow the steps for your OS.

## Windows

### Software prerequisites:

Download and install Visual Studio 2017 (you can use the free community edition, it’s great), if you don’t already have it.

Download and install CMake from …

### Build VXL

Use the CMake GUI to configure and generate project files in the folder you want to store your VXL binaries. When CMake starts in Windows it provides a list of compilers from which to choose, I suggest using “Visual Studio 14 2015 64-bit” as this is what I have used to build and test the projects. However later versions should also be ok. Whichever you pick, make sure you do the same for Madym \*and\* boost (noting that the way you set the compiler when building boost is not the same as you won’t be configuring using CMake).

If you want the smallest, quickest install, use CMake to turn off all options except BUILD\_CORE\_UTILITIES. Once you’ve configured and generated, open the VXL.sln solution file CMake has created in Visual Studio. Choose whether you’re building Debug or Release, then right-click on ALL\_BUILD in the projects pane and click Build. This will take a few minutes.

### Build boost filesystem and system

Download boost from (anything > version 1.69 should be fine).

The tricky bit here is making sure you configure the build settings (achieved using boosts own bootstrap.sh script) to use the correct compiler.

Open a developer terminal (this ports with VS and is essentially a Windows command terminal, with the environment pre-set for your Visual Studio version – start typing “developer” at the Windows toolbar search box and Developer Command Prompt for Visual Studio should be the first app that appears).

CD into the top level boost folder you’ve just downloaded.

Run bootstrap.sh --with-libraries=filesystem,system

The with-libraries option isn’t essential, but will save \*lots\* of time and disk space compared to build all the libraries, which is the default if we don’t set this option. If this works you should get a message about Building the Boost.Build engine, and then a message saying bootstrapping is done.

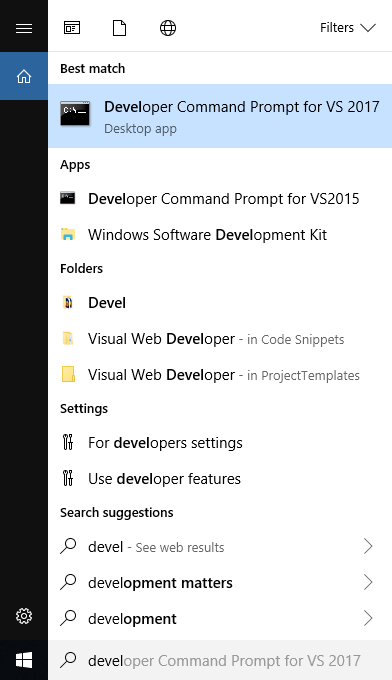
Now call b2 to build the filesystem and system libraries, setting the following options:

1. --toolset=msvc-14.0 this tells boost to use correct compiler
2. address-model=64 this makes sure we build 64-bit binaries
3. --build-type=complete tells boost to build both debug and release versions (if you know you only want one, you can put Debug or Release instead of complete

So the final command should look like:

b2 --toolset=msvc-14.0 address-model=64 --build-type=complete

If the above all works…



# Mac

1. xcode-select install
2. brew install boost
3. brew install llvm
4. brew install cmake
5. Set C++ and C compiler options in ccmake to Clang++ and Clang wherever LLVM got installed by brew in both VXL and Madym
6. CMake may not pick up boost installation, in which case manually set the paths to boost filesystem and system libraries in CCmake

### Software prerequisites:

Homebrew (<https://brew.sh/>)

CMake (brew install cmake)

### Build boost filesystem and system

You can just install boost from homebrew (brew install boost). This appears to work, although can be slow as it will install all the boost libraries.

Alternatively you can build from source, only building the filesystem and system libraries. The instructions are essentially the same as per Windows/Linux . If you choose to build from source, jump to the next step to install LLVM (and thus an up-to-date Clang compiler), and then make sure you configure boost to use this compiler.

### Try the pre-built binaries

With boost installed, the pre-built binaries should \*just work\*. Try them if they do, great!

However, if that doesn’t work, or you want to help develop the code base, continue building from source…

# Madym versioning

There is a single version in (Major).(Minor).(Patch) format for all of the main Madym tools (as opposed to separately incrementing versions for each tool). This version is automatically generated from the git tag on the source repository.

Currently MB is in charge of implementing this. All merges to the master branch should be tagged with a new version.

The built executables should then be transferred to the shared drive for everyone else to access. The install scripts can then be used to update your local machine.