Compile Met.3D from source code: Linux (conda)

This page describes how Met.3D can be compiled from its source code using conda under Linux. If you would like to simply use the software, please use the precompiled binary package instead. If you would like to use a specific branch, or change parts of the source code yourself, this page is for you.

System requirements

You require an OpenGL 4.3 (or higher) capable graphics card and an appropriate Linux driver to run Met.3D. The driver will most likely be a proprietary driver (we've only tested Nvidia so far); open-source drivers for Linux currently do not provide the required capabilities. Before you continue with the installation, make sure that graphics card and driver are installed. If everything is installed correctly, the `glxinfo` command should output something similar to (the important thing is the `OpenGL core profile` version > 4.3):

```
# glxinfo | grep OpenGL

OpenGL vendor string: NVIDIA Corporation
OpenGL renderer string: GeForce GTX TITAN/PCIe/SSE2
OpenGL core profile version string: 4.4.0 NVIDIA 340.96
OpenGL core profile shading language version string: 4.40 NVIDIA via Cg compiler
```

Note: So far we've only been able to test Nvidia drivers. Met.3D uses an Nvidia-specific OpenGL extension that queries the available GPU memory. If you are using a different driver and get an error message stating "`GPU memory limit: 0 kb`" then your driver unfortunately won't work at the moment. However, since Met.3D is meant to be an open-source community project: If you are a programmer and would like to help resolve this issue, please let us know!

Overview

Our recommended approach to compile Met.3D from source and/or to set up a development environment under Linux is using conda. The conda system provides package management that is independent from your Linux system and safely keeps all packages in an environment. This way, everything you install for Met.3D will not interfere with your base system. Also, using conda is independent of your Linux distribution, hence the described approach should work with any Linux distribution and version.

The installation procedure described here works for **Met.3D 1.7** and higher versions (note that it does not work for the 1.6 and earlier versions).

Most dependencies for Met.3D are available as conda packages. After you have created a new conda environment for Met.3D (Step 1), you need to install these dependencies (Step 2). Some dependencies are not (yet) available via conda and need to be installed from source (Step 3). Once this is completed, Met.3D sources can be checked out from the git repository and be compiled (Step 4).

If you haven't done so yet, you need to install conda first. Here, we use the miniconda distribution. We recommend to install miniconda into a directory that provides enough disk space (default is in you home directory, you may want to use a different directory).

Install miniconda

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

Setting up a development environment and compiling Met.3D

Step 1: Create and activate a fresh conda environment

Create and activate a fresh conda environment

```
conda create -n met3d
conda activate met3d
```

Step 2: Install dependencies and development packages

Note: The following installation command may take a while, since all packages need to be downloaded and installed.

```
conda install -c conda-forge cxx-compiler fortran-compiler make cmake pkg-config gdb glew log4cplus libgdal
eccodes netcdf-cxx4 freetype gsl proj qt git mesa-libgl-devel-cos7-x86_64 mesa-dri-drivers-cos7-aarch64 libxau-
devel-cos7-aarch64 libselinux-devel-cos7-aarch64 libxdamage-devel-cos7-aarch64 libxxf86vm-devel-cos7-aarch64
libxext-devel-cos7-aarch64 xorg-libxfixes xorg-libxau
```

(Step 2a: Starting with version 1.12, Met.3D includes a rudimentary Python interface and requires the following additional dependencies)

```
conda install -c conda-forge python pybind11 eigen
```

Step 3: Manual installation of dependencies not available via conda

```
# Create a "met.3d-base" directory in your home directory, with sub-dirs "local" and "third-party".
cd ~
mkdir met.3d-base && cd met.3d-base
mkdir local
mkdir third-party

# Checkout and install glfx.
cd ~/met.3d-base/third-party/
git clone https://github.com/maizensh/glfx.git
cd glfx
cmake -DCMAKE_INSTALL_PREFIX:PATH=~/met.3d-base/local CMakeLists.txt
make -j 12
make install

# Download and install QCustomPlot (download packages from website):
cd ~/met.3d-base/third-party/
wget https://www.qcustomplot.com/release/2.1.0fixed/QCustomPlot.tar.gz
wget https://www.qcustomplot.com/release/2.1.0fixed/QCustomPlot-sharedlib.tar.gz
tar xvfz QCustomPlot.tar.gz
tar xvfz QCustomPlot-sharedlib.tar.gz
mv qcustomplot-sharedlib/ qcustomplot/

cd qcustomplot/qcustomplot-sharedlib/sharedlib-compilation/
qmake
make -j 12
cp libqcustomplot* ~/met.3d-base/local/lib/
cd ../..
cp qcustomplot.h ~/met.3d-base/local/include/

# section C), download remaining third-party dependencies
cd ~/met.3d-base/third-party
git clone https://github.com/qtproject/qt-solutions.git

wget http://ftp.gnu.org/gnu/freefont/freefont-ttf-20120503.zip
unzip freefont-ttf-20120503.zip

mkdir naturalearth
cd naturalearth
wget https://naciscdn.org/naturalearth/50m/physical/ne_50m_coastline.zip
unzip ne_50m_coastline.zip
wget https://naciscdn.org/naturalearth/50m/cultural/ne_50m_admin_0_boundary_lines_land.zip
unzip ne_50m_admin_0_boundary_lines_land.zip
wget https://naciscdn.org/naturalearth/50m/raster/HYP_50M_SR_W.zip
unzip HYP_50M_SR_W.zip

# section D), download remaining but optional third-party dependencies
cd ~/met.3d-base/third-party
wget https://zenodo.org/record/5501399/files/ScientificColourMaps7.zip
unzip ScientificColourMaps7.zip
```

Step 4: Checkout and compile Met.3D

```
cd ~/met.3d-base/
git clone https://gitlab.com/wxmetvis/met.3d.git
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_PREFIX_PATH=~/met.3d-base/local ../met.3d
make -j 12
```

Running Met.3D

```
# Make sure your "met3d" conda environment is activated!
# The Met.3D binary is now located at "~/met.3d-base/build/". To run:
cd ~/met.3d-base/build/
MET3D_HOME=~/met.3d-base/met.3d MET3D_BASE=~/met.3d-base ./Met3D
```

Using QtCreator within the conda envionment for development

This section is only relevant if you want to contribute to Met.3D development using the QtCreator IDE.

Unfortunately, there is no conda package for QtCreator, hence the system installation needs to be used (Ubuntu and OpenSuSE provide QtCreator through their package managers). When using the above conda environment for development, you need to make sure that QtCreator sees the correct development paths.

- Start QtCreator from the command line within the activated "met3d" conda environment

- Open the Met.3D project (File Open File or Project *Select Met.3D's CMakeLists.txt*)
- Click "Manage Kits" Make sure that all tools in the correct PATH are used (i.e., gcc, gdb, etc. from the conda environment - in some attempts, a "wrong" gdb was selected by default)
- Click "Make default" for "Qt 5.12.9 in PATH (met3d) - temporary" (or whichever Qt version is current as you read this)
- Possibly adjust the Debug and Release paths for the Met.3D binaries
- Click "Configure Project"

- In the "Build Settings":
    - In "CMake", set CMAKE_PREFIX_PATH to ~/met.3d-base/local
    - In "Build Steps", add "-j 12" (or however many core you have available) to build arguments for parallel compilation
- In the "Run Settings":
    - Change the Met.3D working directory to point to the source directory
    - Add the MET3D_BASE and MET3D_HOME environment variables

- (Note on the code model analyzer: QtCreator by default uses Clang to analyze the code. On my Ubuntu 20.04 LTS, clang is installed from the Ubuntu system but not in the conda environment above. This seems to interfere, and QtCreator displays a number of error messages related to not finding system files. I switched off the plugins "C++/ClangCodeModel" and "Code Analyzer/ClangTools" in the Help/About plugins dialog to make the error message disappear. An alternative could be to install clang in the conda environment, or to compile QtCreator from source in the conda environment.)