



Download

- Binaries
- Source Code
- Developer Zone

Help

- CC3D User Forum
- Manuals
- Tutorials
- F.A.Q.

Demos

- Simulation Movies
- Screenshots
- Model Repository

Publications

- Publications
- Theses
- Talks and Posters

Events

- Workshops

About

- People
- Contact Us
- Mailing List

Search Site

= Building CompuCell3D on Red Hat Enterprise Linux 6 and CentOS 6= Building CompuCell3D from source on Unix/Linux systems is fairly straightforward once all of it's dependencies have been satisfied. The following commands should build and install CC3D on RHEL 6 systems.

Prerequisites

Hardware

CompuCell3D may build and run with less capable hardware, but has been tested with the following:

- 1024MB RAM
- Hardware 3D Graphics Acceleration (most modern graphics cards)

Build Tools and Dependencies

we will assume that you have already installed Software Developer Workstation Package Group from Red Hat 6 distribution. Typically when you install RHEL 6 you have a choice of the package groups and Software Developer Workstation is one of such groups. If you talk to your sys-admin they can certainly help you install this package set. RHEL ships with Python 2.6 by default and CC3D runs best with Python 2.7. Therefore we will build CC3D based on Miniconda Python distribution from Continuum Analytics.

Using miniconda package manager we will install

- pyqt4
- numpy
- scipy
- vtk

This will save us a lot of time compared to the manual compilation of those packages.

CompuCell3D requires also the following packages to be present:

- swig
- cmake
- PyQwt
- QScintilla (with python bindings)
- pcre-devel

We will install those packages either using **yum** package manager or will compile them from source.

Let's start with the easy part first -**swig** and **cmake** installation. Open a terminal and become root by typing :

```
su -
```

enter your root password and type the following:

```
yum install cmake-gui swig pcre-devel
```

Now let's install miniconda - here I will choose to perform a local installation i.e. I will install it in my home directory. First let's get appropriate package - by visiting <http://conda.pydata.org/miniconda.html>

and choosing Python 2.7 64-bit installer. Navigate to your download directory and make the downloaded script executable :

```
chmod +x Miniconda2-latest-Linux-x86_64.sh
```

After that, run the installation by typing from your download directory

```
./Miniconda2-latest-Linux-x86_64.sh
```

```

m@localhost:~/Downloads
File Edit View Search Terminal Tabs Help
mc [... X] m@l... X m@l... X mc[r... X mc[... X m@lo... X m@lo... X m@lo... X
[m@localhost ~]$ cd Downloads/
[m@localhost Downloads]$ ./Miniconda2-latest-Linux-x86_64.sh

Welcome to Miniconda2 4.1.11 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> 

```

Towards the end of the installation you will be asked if you want the installer script to add path to miniconda Python distribution to the **PATH** environment variable. I recommend you say **yes**.

```

m@localhost:~/Downloads
File Edit View Search Terminal Tabs Help
mc [... X] m@l... X m@l... X mc[r... X mc[... X m@lo... X m@lo... X m@lo... X
PREFIX=/home/m/miniconda2
installing: python-2.7.12-1 ...
installing: conda-env-2.5.2-py27_0 ...
installing: openssl-1.0.2h-1 ...
installing: pycosat-0.6.1-py27_1 ...
installing: pyyaml-3.11-py27_4 ...
installing: readline-6.2-2 ...
installing: requests-2.10.0-py27_0 ...
installing: ruamel_yaml-0.11.14-py27_0 ...
installing: sqlite-3.13.0-0 ...
installing: tk-8.5.18-0 ...
installing: yaml-0.1.6-0 ...
installing: zlib-1.2.8-3 ...
installing: conda-4.1.11-py27_0 ...
installing: pycrypto-2.6.1-py27_4 ...
installing: pip-8.1.2-py27_0 ...
installing: wheel-0.29.0-py27_0 ...
installing: setuptools-23.0.0-py27_0 ...
Python 2.7.12 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Miniconda2 install location
to PATH in your /home/m/.bashrc ? [yes|no]
[no] >>> yes

```

If you change your mind you can always open **.bashrc** script and remove path to miniconda. Adding path to miniconda python will make it your default Python and this is a good thing on a system that ships with obsolete version of Python. Additionally installing additional Python packages using miniconda toolset is super easy. So let's do it- I will show you how to install **pyqt**, **numpy**, **vtk**, and **scipy** using miniconda. I assume that if you type

```
conda
```

you will get **conda** usage help. If not then for the reminder of presented installation steps you will need to specify full path to the **conda** program.

Let's install the required dependencies now:

```
conda install pyqt numpy scipy vtk
```

```
m@localhost:~/Downloads
File Edit View Search Terminal Tabs Help

mc [... X] m@l... X] m@l... X] mc[r... X] mc[...] X] m@lo... X] m@lo... X] m@lo... X]

installing: conda-4.1.11-py27_0 ...
installing: pycrypto-2.6.1-py27_4 ...
installing: pip-8.1.2-py27_0 ...
installing: wheel-0.29.0-py27_0 ...
installing: setuptools-23.0.0-py27_0 ...
Python 2.7.12 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Miniconda2 install location
to PATH in your /home/m/.bashrc ? [yes|no]
[no] >>> yes

Prepending PATH=/home/m/miniconda2/bin to PATH in /home/m/.bashrc
A backup will be made to: /home/m/.bashrc-miniconda2.bak

For this change to become active, you have to open a new terminal.

Thank you for installing Miniconda2!

Share your notebooks and packages on Anaconda Cloud!
Sign up for free: https://anaconda.org

[m@localhost Downloads]$ conda install pyqt4 numpy scipy vtk
```

Answer **yes** if **conda** asks whether to install the dependencies, wait few minutes and you will have most of cc3d dependencies needed to run CC3D on your machine.

```
m@localhost:~/Downloads
File Edit View Search Terminal Tabs Help

mc [... X] m@l... X] m@l... X] mc[r... X] mc[...] X] m@lo... X] m@lo... X] m@lo... X]

Total: 243.1 MB

The following NEW packages will be INSTALLED:

  cairo: 1.12.18-6
  fontconfig: 2.11.1-6
  freetype: 2.5.5-1
  glib: 2.43.0-1
  harfbuzz: 0.9.39-1
  libffi: 3.2.1-0
  libgfortran: 3.0.0-1
  libpng: 1.6.22-0
  libxml2: 2.9.2-0
  mkl: 11.3.3-0
  numpy: 1.11.1-py27_0
  pango: 1.39.0-1
  pixman: 0.32.6-0
  pyqt: 4.11.4-py27_4
  qt: 4.8.7-4
  scipy: 0.18.0-np111py27_0
  sip: 4.18-py27_0
  vtk: 6.3.0-py27_1

Proceed ([y]/n)?
```

To check if everything went OK open a terminal and type

```
python
```

followed by

```
from PyQt4 import QtCore
```

The output should look as shown below:



```

m@localhost:~
File Edit View Search Terminal Tabs Help
mc [m@l... X] m@local... X] m@local... X] mc [root... X] mc [m@l... X] m@localh... X]
[m@localhost ~]$ python
Python 2.7.12 |Continuum Analytics, Inc.| (default, Jul  2 2016, 17:42:40)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> from PyQt4 import QtCore
>>> 

```

Now we will need to compile two packages

- QScintilla (with Python bindings) - needed by CC3D player and Twedit++
- [PyQt](#) 5.2.0 (needed for plotting capabilities within Player)

Let's start with QScintilla. First let's navigate to QScintilla download page - <https://www.riverbankcomputing.com/software/qscintilla/download>

We will get source code tarball for Linux. The installation instructions are in <http://pyqt.sourceforge.net/Docs/QScintilla2/>

and I suggest you follow them. But first let's make sure that you are picking the correct installation of the Qt. You want qt bundled with miniconda. To make sure this is the case type

```
qmake --version
```

The output should show that indeed your shell sees qt version bundled with miniconda:



```

m@localhost:~/builds/QScintilla_gpl-2.9.3/Qt4Qt5
File Edit View Search Terminal Tabs Help
mc [... X] m@l... X] m@l... X] mc [r... X] mc [... X] m@lo... X] m@lo... X] m@lo... X]
[m@localhost QScintilla_gpl-2.9.3]$ cd Qt4Qt5/
[m@localhost Qt4Qt5]$ qmake --version
QMake version 2.01a
Using Qt version 4.8.7 in /home/m/miniconda2/lib
[m@localhost Qt4Qt5]$ 

```

If this is not the case simply prepend path to miniconda's **bin** directory to your **PATH** environment variable.

If everything works as follows you should be able to follow the installation instructions for QScintilla"

```
cd Qt4Qt5
qmake qscintilla.pro
make
make install
```

After QScintilla is build we will build python bindings for it. Assuming we are in the Qscintilla source main directory we would type

```
cd Python
python configure.py
make
make install
```

At this point we should have QScintilla and Python bindings to it installed on our machine

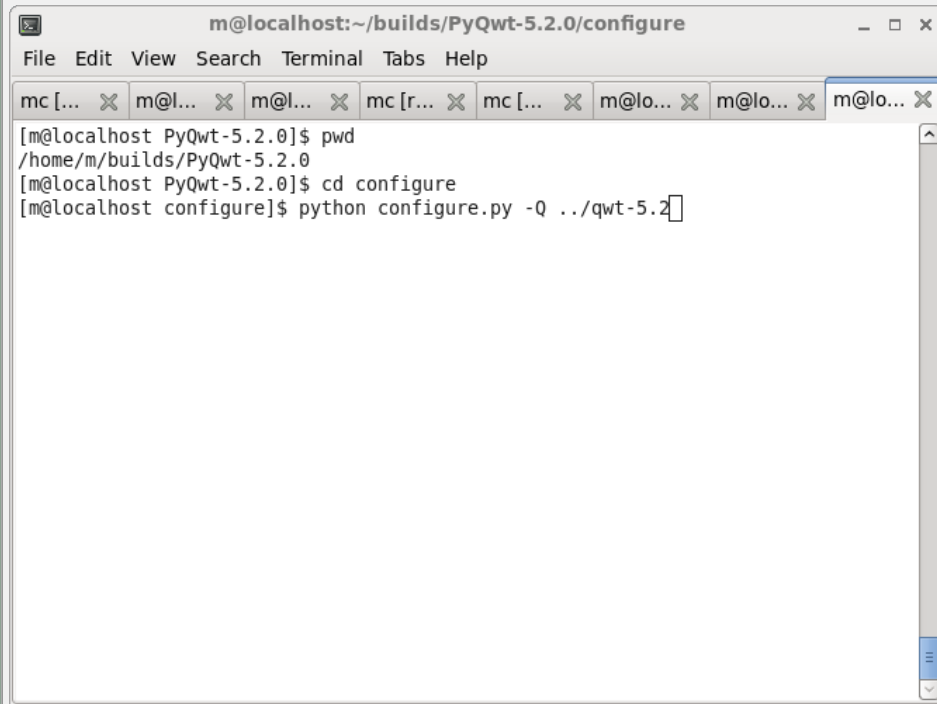
Now let's look at the installation of plotting library **PyQt-5.2.0**

You get it from <https://sourceforge.net/projects/pyqwt/files/pyqwt5/PyQwt-5.2.0/PyQwt-5.2.0.tar.gz/download>

Unpack it and go to the unpacked **PyQwt** folder

You should be able to follow the installation instructions shown here - <http://pyqwt.sourceforge.net/doc5/installation.html>

Here is my console window showing basis steps of **PyQwt** installation:



```
m@localhost:~/builds/PyQwt-5.2.0/configure
File Edit View Search Terminal Tabs Help
mc [... X] m@l... X] m@l... X] mc[r... X] mc[... X] m@lo... X] m@lo... X] m@lo... X]
[m@localhost PyQwt-5.2.0]$ pwd
/home/m/builds/PyQwt-5.2.0
[m@localhost PyQwt-5.2.0]$ cd configure
[m@localhost configure]$ python configure.py -Q ../qwt-5.2
```

```

m@localhost: ~/builds/PyQwt-5.2.0/configure
File Edit View Search Terminal Tabs Help

mc [... X] m@lo... X] m@lo... X] mc [r... X] mc [... X] m@lo... X] m@lo... X] m@lo... X]

Copy tmp-qwt5qt4/qplt.py -> qwt5qt4/qplt.py.
Copy tmp-qwt5qt4/__init__.py -> qwt5qt4/__init__.py.
Copy tmp-qwt5qt4/grace.py -> qwt5qt4/grace.py.
Copy tmp-qwt5qt4/iqt.py -> qwt5qt4/iqt.py.
Copy tmp-qwt5qt4/pythonrc.py -> qwt5qt4/pythonrc.py.
Copy tmp-qwt5qt4/ipy_user_conf.py -> qwt5qt4/ipy_user_conf.py.
Copy tmp-qwt5qt4/anynumpy.py -> qwt5qt4/anynumpy.py.
Copy tmp-qwt5qt4/qwt5qt4.sbf -> qwt5qt4/qwt5qt4.sbf.
255 file(s) lazily copied.
Listing qwt5qt4 ...
Compiling qwt5qt4/__init__.py ...
Compiling qwt5qt4/anynumpy.py ...
Compiling qwt5qt4/grace.py ...
Compiling qwt5qt4/ipy_user_conf.py ...
Compiling qwt5qt4/iqt.py ...
Compiling qwt5qt4/pythonrc.py ...
Compiling qwt5qt4/qplt.py ...
Listing ../qt4lib/PyQt4/uic/widget-plugins ...
Compiling ../qt4lib/PyQt4/uic/widget-plugins/qwt.py ...

Setup the PyQwt build.

Great, run make or nmake to build and install PyQwt.
[m@localhost configure]$

```

One thing you have to be aware of here. There is a small bug in the [PyQwt](#) source code. Fortunately it is very easy to fix. Here we go - navigate to [PyQwt](#) installation folder located inside miniconda folder. In my case since miniconda was installed in

```
/home/m/miniconda2
```

the folder I am looking for is

```
/home/m/miniconda2/lib/python2.7/site-packages/PyQt4/Qwt5
```

We will fix anynumpy.py script as follows - we will replace original code (I am shogin only first part of the file- look for lines following if name == 'numpy':)

```

# import either NumPy, or numarray, or Numeric
for name in ('numpy', 'numarray', 'Numeric'):
    failed = False
    try:
        eval(compile('from %s import *' % name, 'eval', 'exec'))
        if name == 'numpy':
            from numpy.oldnumeric.compat import *
            Float = float
            UInt8 = uint8
    except ImportError:
        failed = True
    if not failed:
        break

```

with

```

for name in ('numpy', 'numarray', 'Numeric'):
    failed = False
    try:
        eval(compile('from %s import *' % name, 'eval', 'exec'))
        if name == 'numpy':
            pass
            #from numpy.oldnumeric.compat import *
            #Float = float
            #UInt8 = uint8
    except ImportError:
        failed = True
    if not failed:
        break

```

If you prefer downloading the fixed file , you can do it by clicking [here](#)

At this point we are done with the dependencies

Source Code

Once the dependencies have been satisfied make directory where you want to store source code, in my case it is in **/home/m/CC3D_GIT**

```
mkdir /home/m/CC3D_GIT
cd /home/m/CC3D_GIT
```

Once the directory has been created, obtain the source code from our GIT repository using the following command:

```
git clone https://github.com/CompuCell3D/CompuCell3D.git .
```

This will clone CC3D Git repository into current directory (remember about the `.` at the end of last command - it is important)

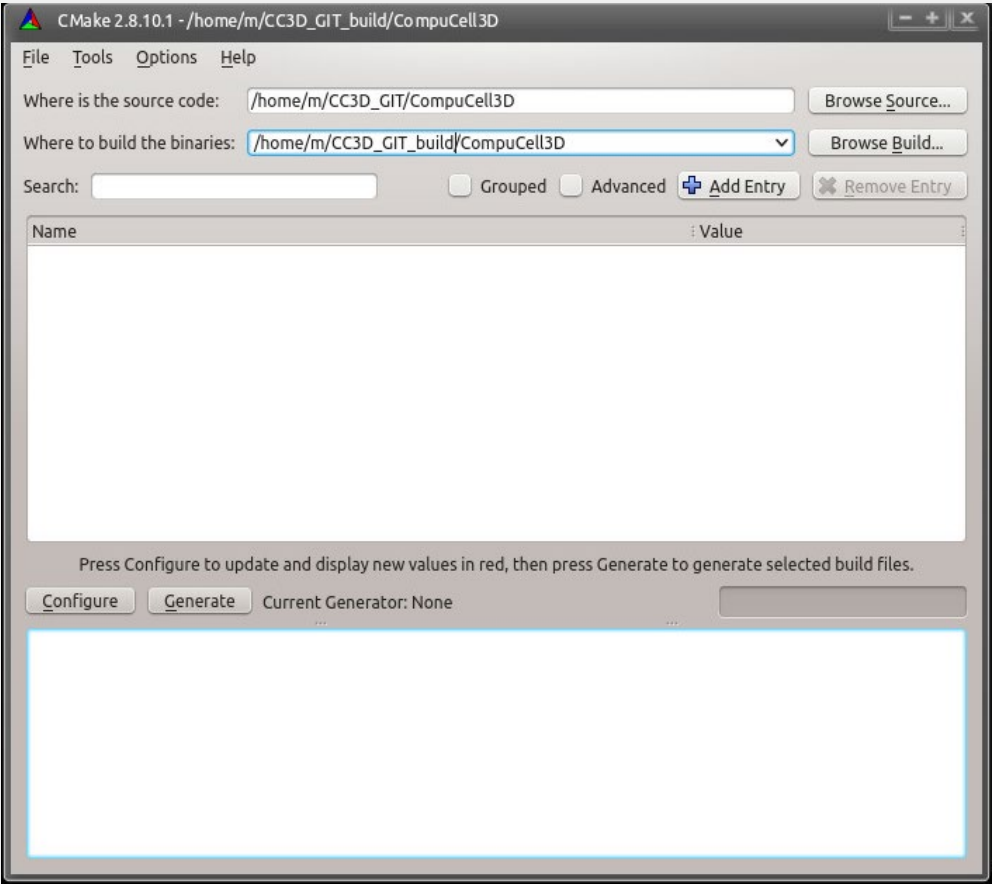
Build Configuration: Starting CMake

CompuCell3D is configured using the CMake build system. The following command starts the CMake GUI:

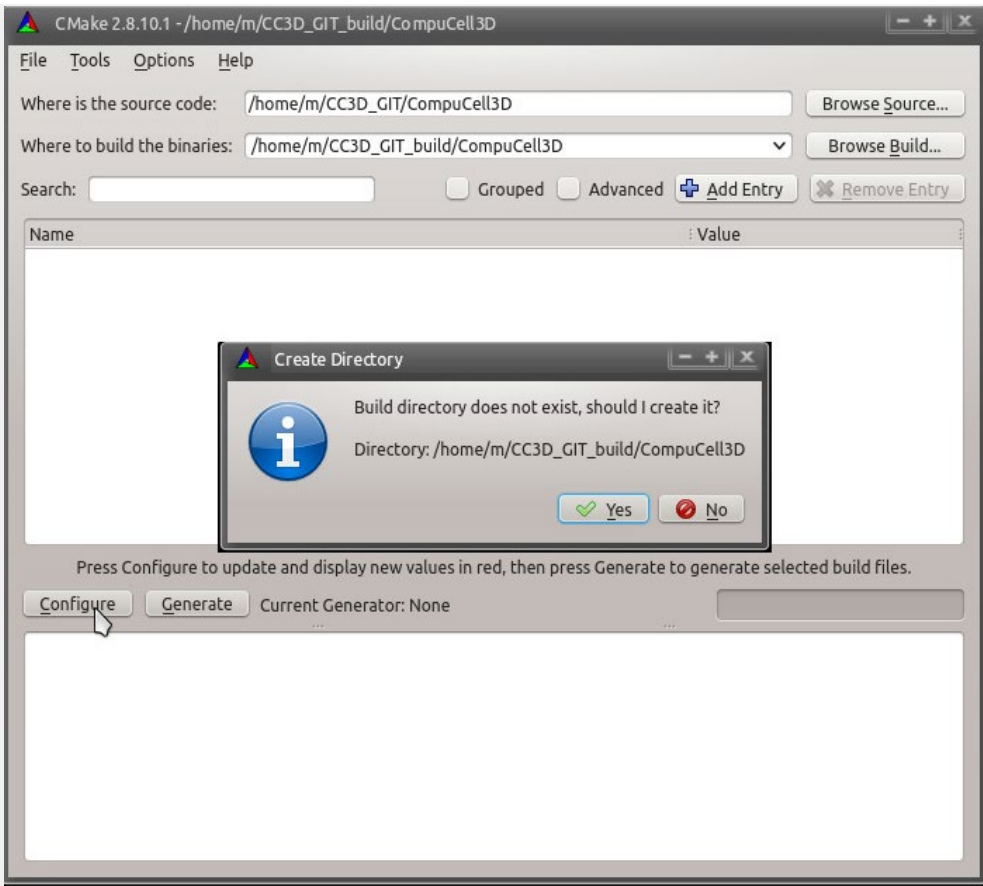
```
cmake-gui
```

That will start the build system. Having miniconda is in your **PATH** environment variable will help locate the dependencies . Click **Browse Source...** and select the **CompuCell3D** source directory from the CompuCell3D GIT directory we have created above - in my case the CompuCell3D source directory is located in **/home/m/CC3D_GIT/CompuCell3D**

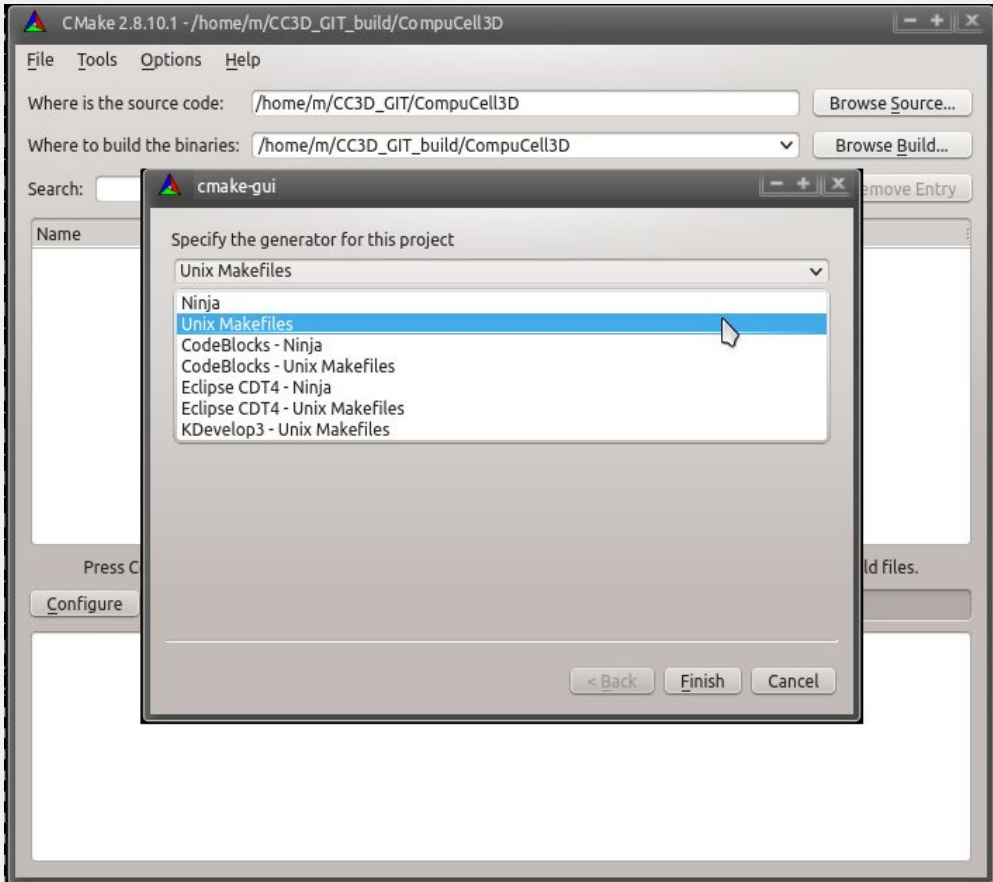
In addition to specifying source directory we also specify the location of the build directory i.e. a directory where compilation files will be stored. in my case it is **/home/m/CC3D_GIT_build/CompuCell3D**



We are ready to click **Configure**. A dialog box asking to create the build directory will appear:



and then one asking about the build system:

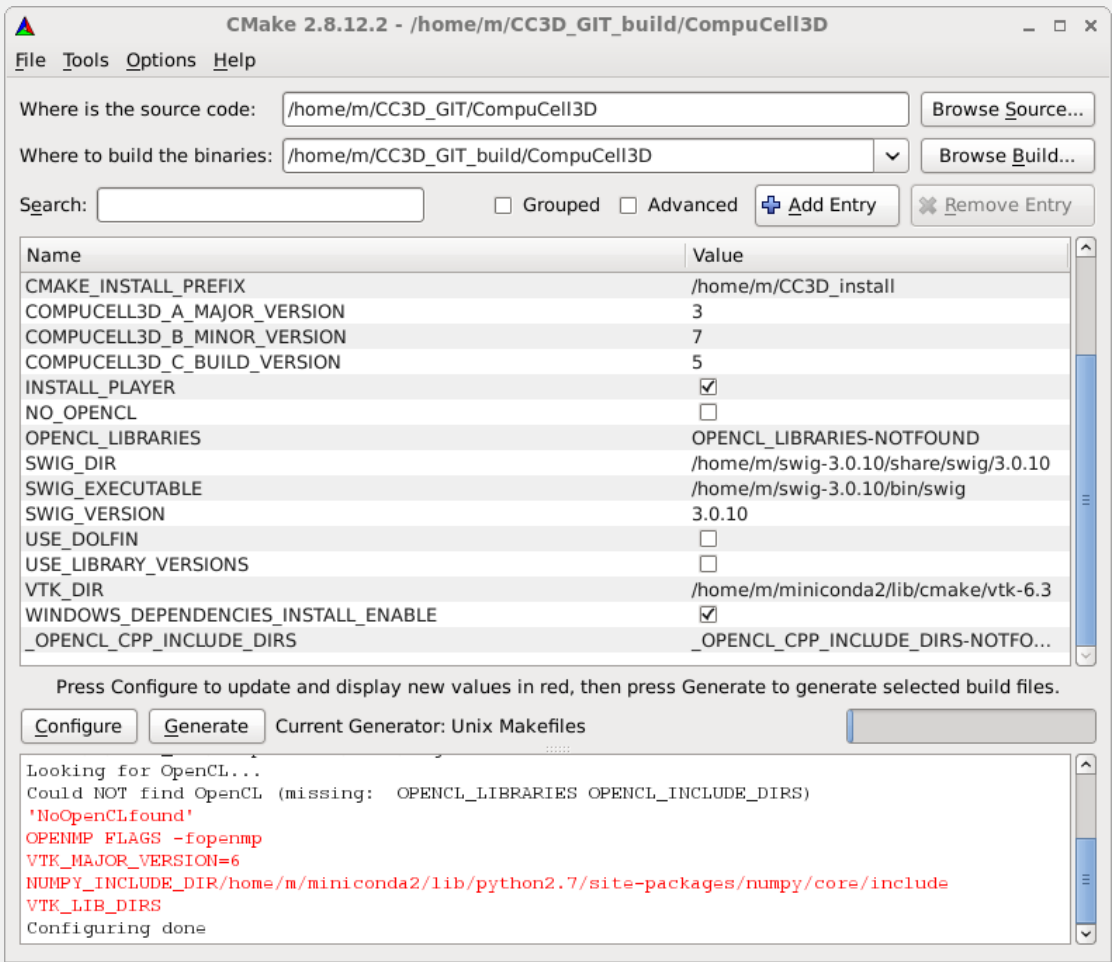


Select **Unix Makefiles** for the generator, select **Use default native compilers** and then click **Finish**. CMake will begin the configuration process. CMake will attempt to locate all of the dependencies installed above.

After the initial configuration has completed select **Grouped** and **Advanced** to make entering configuration values easier.

Build Configuration: CMAKE

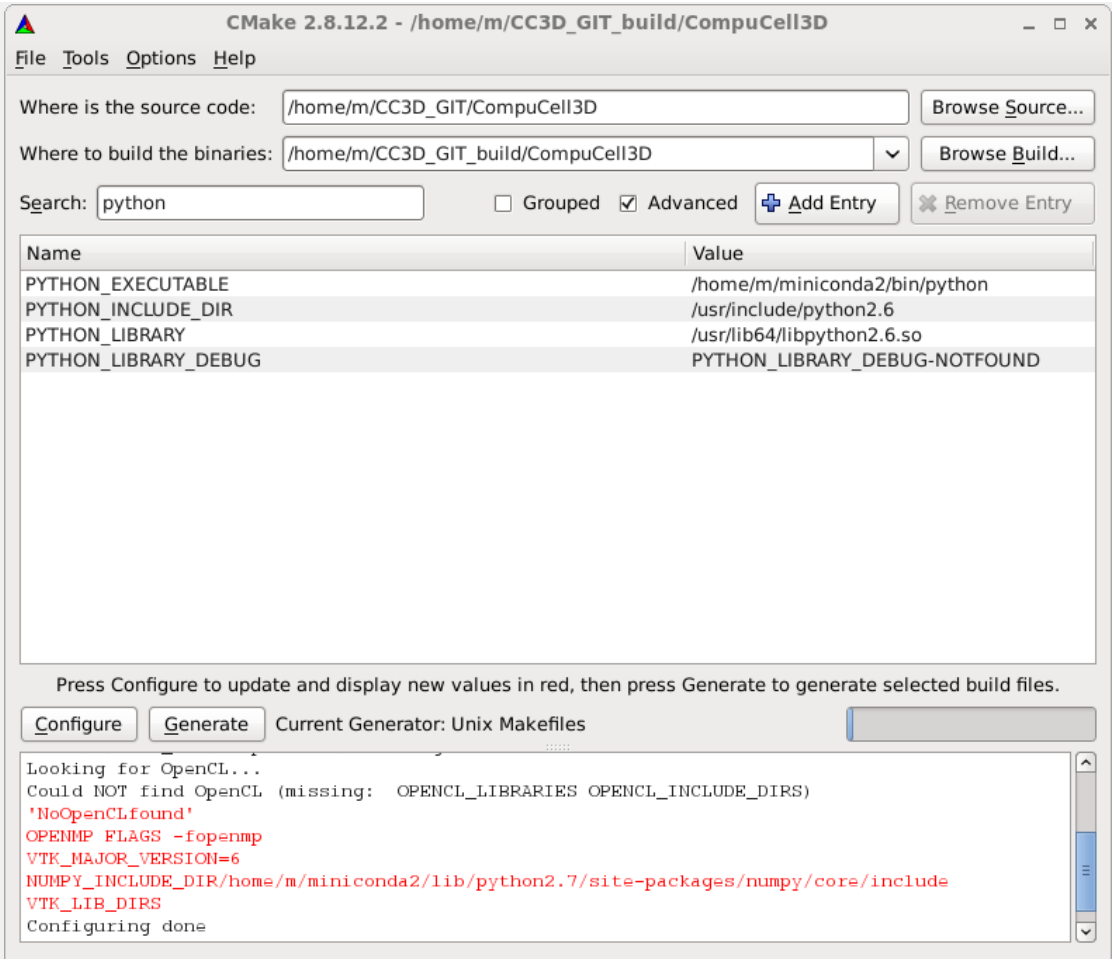
In **CMAKE** confirm that CMake has located all the build tools. In the **CMAKE BUILD TYPE** field you may enter **Debug**, **RelWithDebInfo** or **Release** to specify the type of the binary you want to have - if you are developing extra modules compiling in the **Debug** or **RelWithDebInfo** can be helpful. By default CompuCell3D build type is set to **Release**:



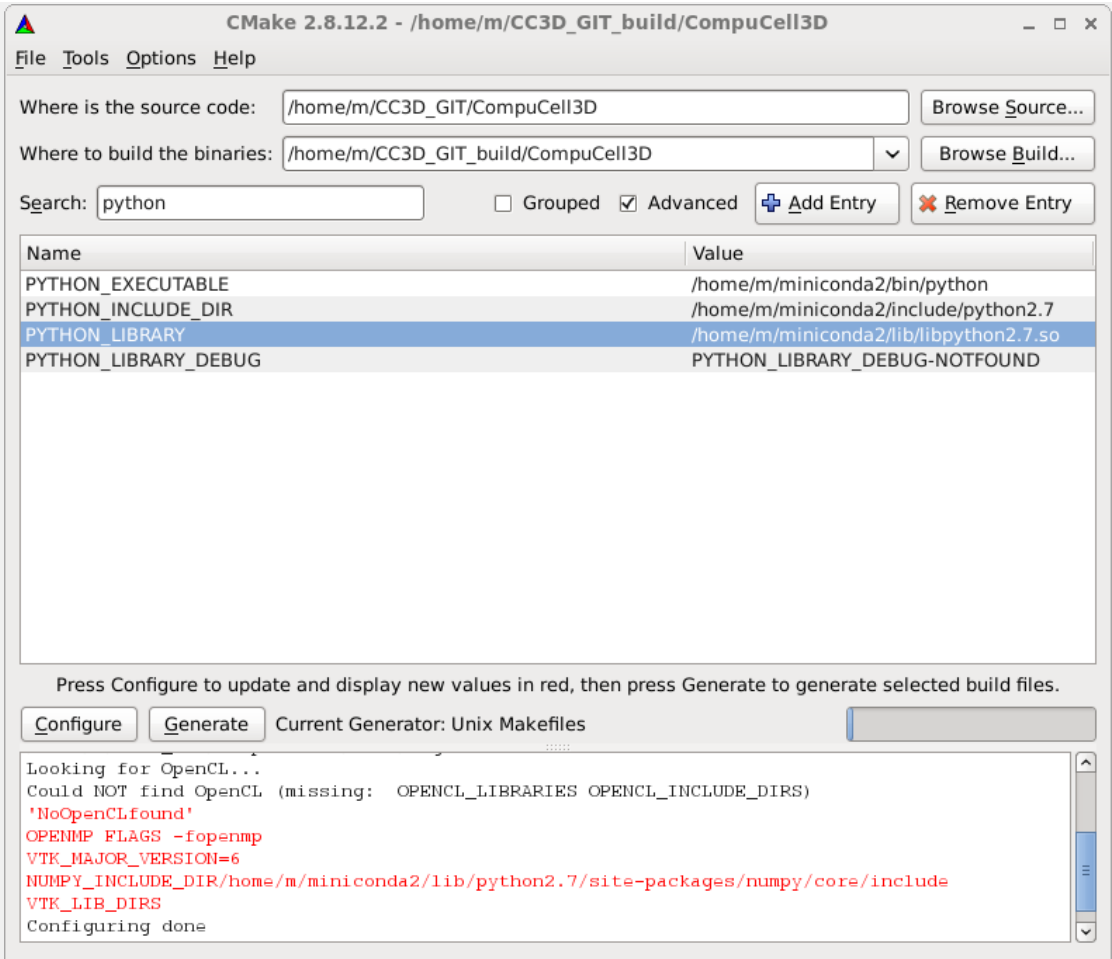
In the **CMAKE_INSTALL_PREFIX** field enter the directory you would like to install CompuCell3D into - this is important. We recommend that unless necessary you should install CC3D into local directory - in my case it is **/home/m/CC3D_install**, .

Click on **Configure** to have all the values updated.

IMPORTANT CMake will try to locate all the dependencies and Python is one of the dependencies. Alas, CMake script that locates Python distribution (Python executable, library and header directory) mixes different Python distributions if you have more than one Python distribution on your system. Therefore it is important to fix the appropriate paths in the cmak-gui before proceeding further. To do this. Check **Advanced** box, and in the search line next to the box type **python** and you will see the following Python dependencies listed:



As you can see Python executable comes from miniconda distribution while library and header files come from system-default Python (which is obsolete 2.6 edition). We need to fix it, specifying paths to the directory containing **Python.h** file and the path to the python library. In my case those are /home/m/miniconda2/include/python2.7 and /home/m/miniconda2/lib/libpython2.7.so respectively - see screenshot below



After this fix we can move further - click on **Configure** and then click on **Generate** to generate the Makefiles, ignore any warnings. Exit CMake

CMake 2.8.12.2 - /home/m/CC3D_GIT_build/CompuCell3D

FileToolsOptionsHelp

Where is the source code: /home/m/CC3D_GIT/CompuCell3D

Where to build the binaries: /home/m/CC3D_GIT_build/CompuCell3D

Search:

Grouped

Advanced

Add Entry

Remove Entry

Name	Value
BUILD_CPP_ONLY_EXECUTABLE	<input type="checkbox"/>
BUILD_PYINTERFACE	<input checked="" type="checkbox"/>
BUILD_QT_WRAPPERS	<input type="checkbox"/>
BUILD_SHARED_LIBS	<input checked="" type="checkbox"/>
CMAKE_BUILD_TYPE	Release
CMAKE_INSTALL_PREFIX	/home/m/CC3D_install
COMPUCELL3D_A_MAJOR_VERSION	3
COMPUCELL3D_B_MINOR_VERSION	7
COMPUCELL3D_C_BUILD_VERSION	5
INSTALL_PLAYER	<input checked="" type="checkbox"/>
NO_OPENCL	<input type="checkbox"/>
OPENCL_LIBRARIES	OPENCL_LIBRARIES-NOTFOUND
SWIG_DIR	/home/m/swig-3.0.10/share/swig/3.0.10
SWIG_EXECUTABLE	/home/m/swig-3.0.10/bin/swig
SWIG_VERSION	3.0.10
USE_BOLEIN	<input type="checkbox"/>

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Configure

Generate

Current Generator: Unix Makefiles

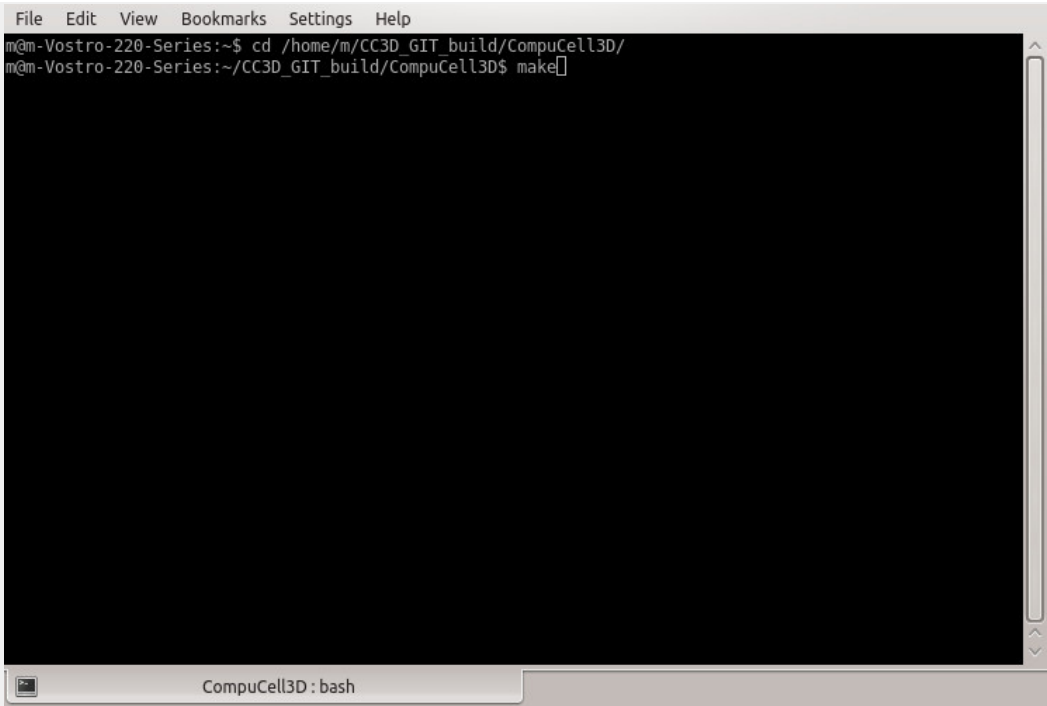
Could NOT find OpenCL (missing: OPENCL_LIBRARIES OPENCL_INCLUDE_DIRS)
'NoOpenCLfound'
OPENMP_FLAGS -fopenmp
VTK_MAJOR_VERSION=6
NUMPY_INCLUDE_DIR/home/m/miniconda2/lib/python2.7/site-packages/numpy/core/include
VTK_LIB_DIRS
Configuring done
Generating done

Building

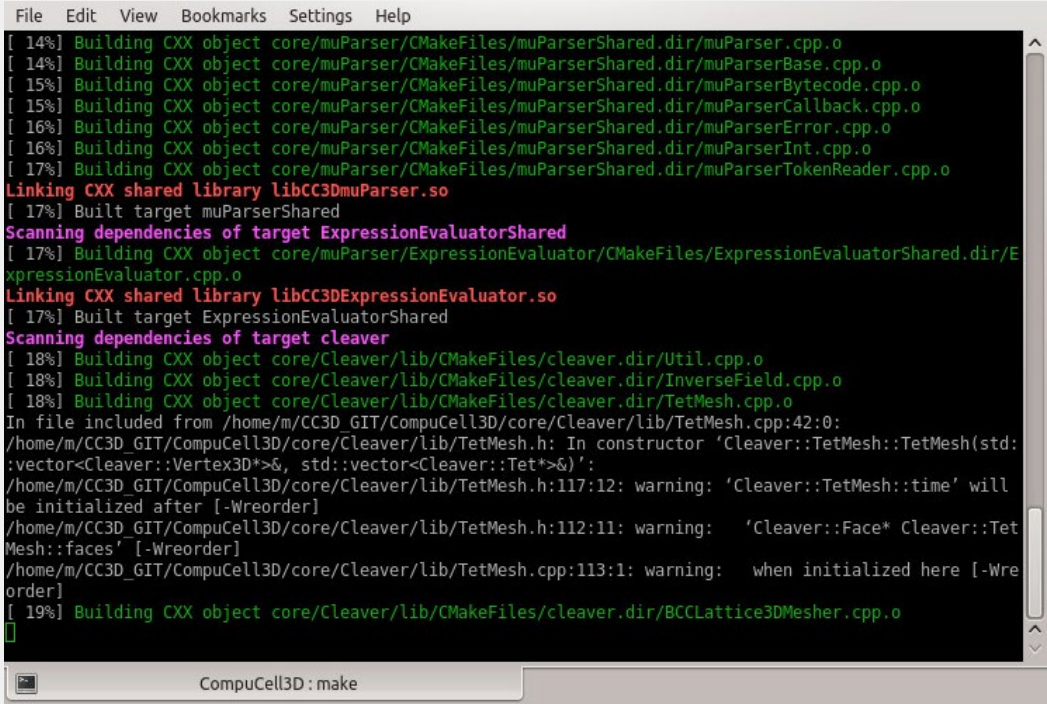
To begin compiling CompuCell3D enter the build directory specified in the **Where to build the binaries** field of CMake earlier and use the following command:

In my case I type:

```
cd /home/m/CC3D_GIT_build/CompuCell3D
make
```



After the build begins you may see screen like this one:



Installing

Once compiling has completed CompuCell3D can be installed into the directory specified in the `CMAKE_INSTALL_PREFIX` field earlier by issuing the following command:

```
make install
```

FileEditViewBookmarksSettingsHelp

```
/home/m/CC3D_GIT/CompuCell3D/core/BasicUtils/BasicException.h:153: Warning 314: 'print' is a python key
word, renaming to '_print'
/home/m/CC3D_GIT/CompuCell3D/core/BasicUtils/BasicException.h:153: Warning 314: 'print' is a python key
word, renaming to '_print'
/home/m/CC3D_GIT/CompuCell3D/core/Utils/Coordinates3D.h:49: Warning 362: operator= ignored
/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D/Field3D/Point3D.h:65: Warning 362: operator= ignored
/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D/Field3D/Dim3D.h:52: Warning 362: operator= ignored
/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D/Field3D/Dim3D.h:96: Warning 389: operator[] ignored (cons
ider using %extend)
/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D/Field3D/Field3D.h:142: Warning 389: operator[] ignored (c
onsider using %extend)
/home/m/CC3D_GIT/CompuCell3D/core/BasicUtils/BasicException.h:168: Warning 503: Can't wrap 'operator <<
' unless renamed to a valid identifier.
/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D/Field3D/Point3D.h:106: Warning 503: Can't wrap 'operator
<<' unless renamed to a valid identifier.
/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D/Field3D/Dim3D.h:98: Warning 503: Can't wrap 'operator <<'
unless renamed to a valid identifier.
Scanning dependencies of target _Fields
[100%] Building CXX object core/pyinterface/Fields/CMakeFiles/_Fields.dir/FieldsPYTHON_wrap.o
In file included from /usr/include/python2.7/numpy/ndarraytypes.h:1728:0,
                 from /usr/include/python2.7/numpy/ndarrayobject.h:17,
                 from /usr/include/python2.7/numpy/arrayobject.h:15,
                 from /home/m/CC3D_GIT_build/CompuCell3D/core/pyinterface/Fields/FieldsPYTHON_wrap.cxx:
3476:
/usr/include/python2.7/numpy/npymath/npymath.h:11:2: warning: #warning "Using deprecated NumPy API,
disable it by #defining NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION" [-Wcpp]
Linking CXX shared module _Fields.so
[100%] Built target _Fields
m@m-Vostro-220-Series:~/CC3D_GIT_build/CompuCell3D$ make install[]
```

CompuCell3D: bash

m: mc

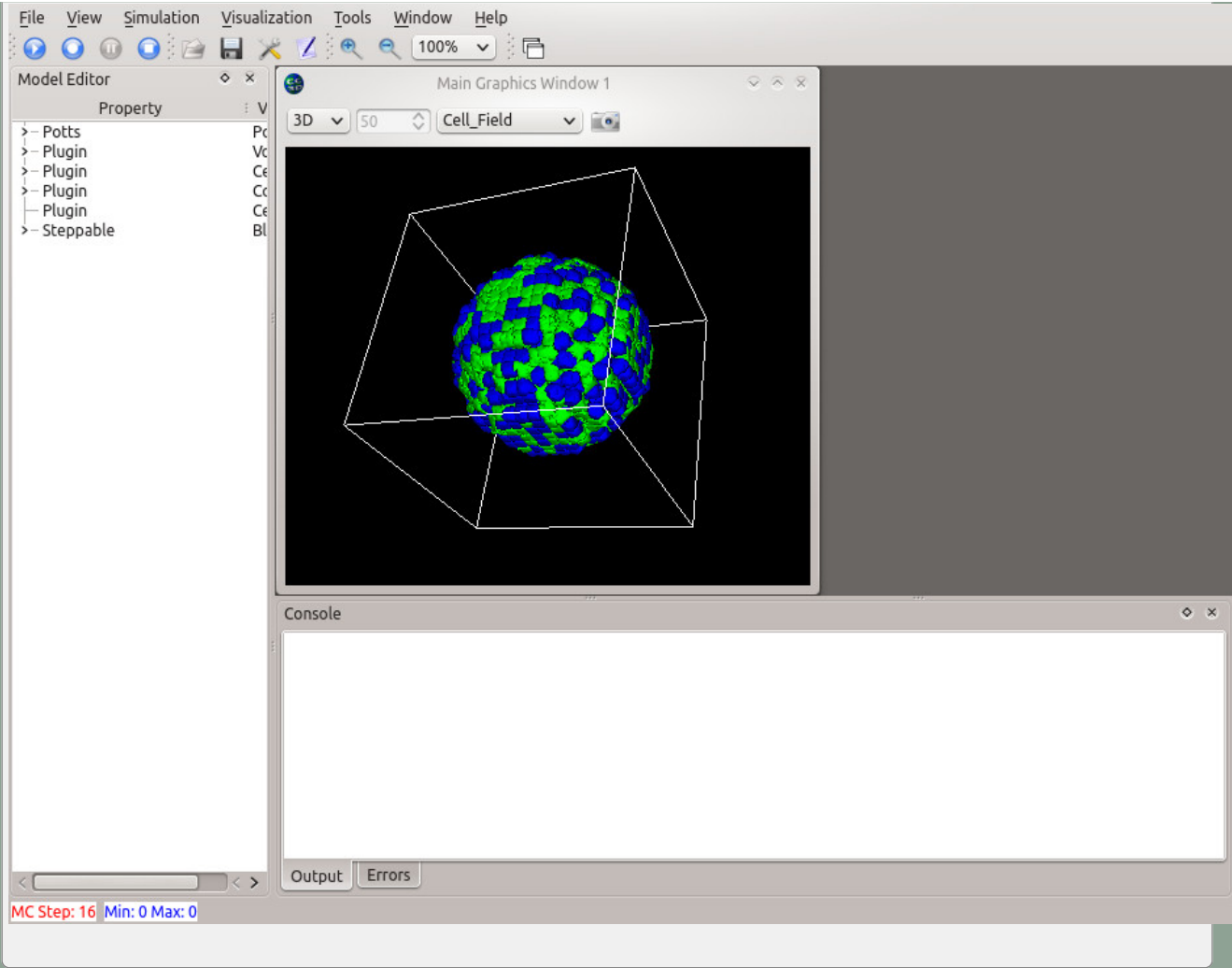
Running


After CompuCell3D has finished installing you can start CompuCell3D by entering the installation directory (in my case it is in **/home/m/CC3D_install**):

```
cd /home/m/CC3D_install
```

and issuing the following command:

```
./compuCell3d.sh
```



Maintained by [IU](#) and the [Biocomplexity Institute](#)
CC3D® and  The CompuCell 3D Environment® logo are registered trademarks.