Generating C++ Steppable Inside DeveloperZone Folder - One-Click Twedit++ shortcut

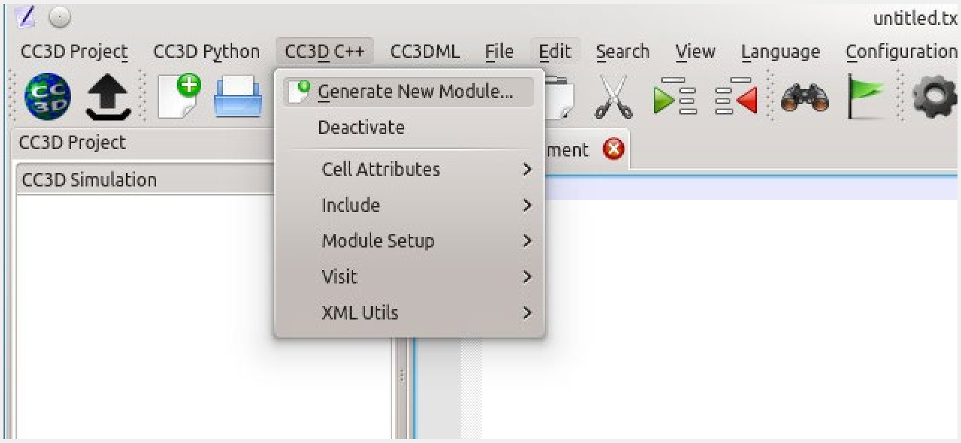
In this tutorial I will show you how to autogenerate code for a C++ CompuCell3D Steppable using Twedit++. **Note** we have developed a video tutorial that walks you through most of the steps outlined in this tutorial. There are some differences but if you prefer to watch video here is the video tutorial

Sometimes when working on your simulation you may find Python code to be a bit of a bottleneck and you may want to speed things up using C++ . This is a good strategy but as you will quickly discover it requires a bit more work than developing Python code. Therefore we suggest that before resorting to C++ you try to optimize your Python code first. However, there is no denying that properly written C++ code will almost always outperform its Python equivalent.

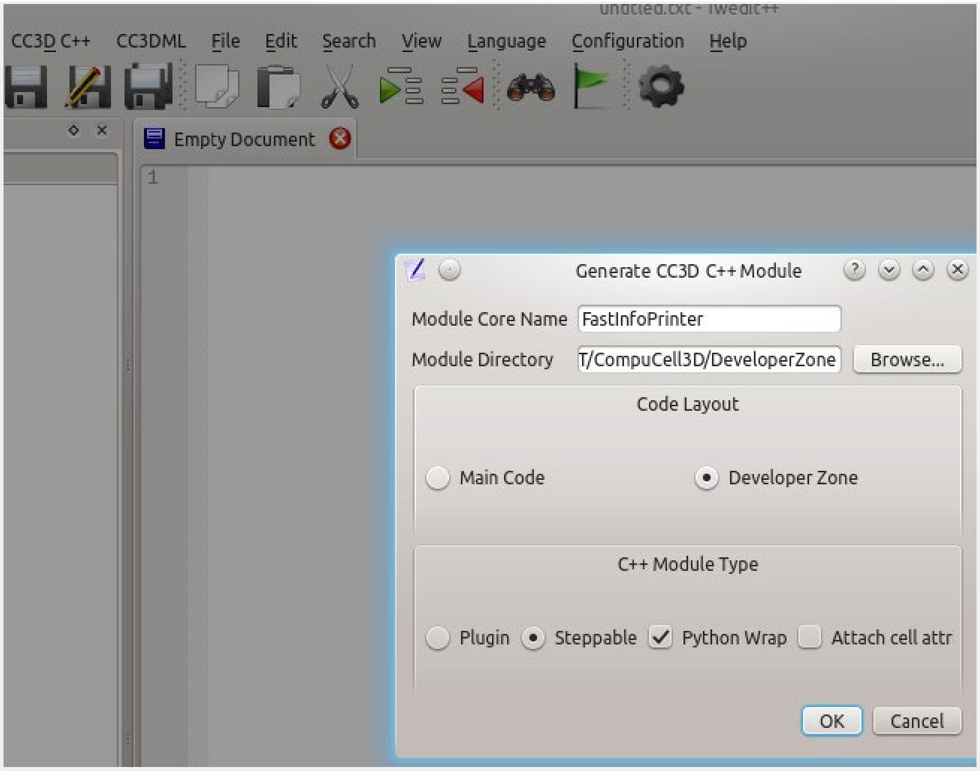
Before we begin I will assume that you already know how to compile CC3D from source and that you have already compiled CC3D and have it available somewhere on your computer. If you need help with basic CompuCell3D compilation check one of those tutorials

Generating C++ Steppable Code using Twedit++

To generate C++ steppable code , open up twedit and select **CC3D C++ -> Generate New Module ...** - as shown below



After that a pop-up dialog will display where you specify the name of your module (I used **FastInfoPrinter**) and the location of the folder in the CC3D source tree where to generate new code. In our case we will use **DeveloperZone** folder because this way we will not "pollute" main CC3D code but rather develop module that is external to the CC3D main source tree. In my case my CC3D git repository is stored in **/home/m/CC3D_GIT** and the **DeveloperZone** is located in **/home/m/CC3D_GIT/CompuCell3D/DeveloperZone** - and I will use it to populate **Module Directory** in the dialog below. in addition in the **Code Layout** section I pick **DeveloperZone** and in the **C++ Module Type** I select **Steppable** and also check **Python Wrap** to allow this steppable to be accessed from Python (to e.g. conveniently pass some parameters to it from Python script).



Generate CC3D C++ Module

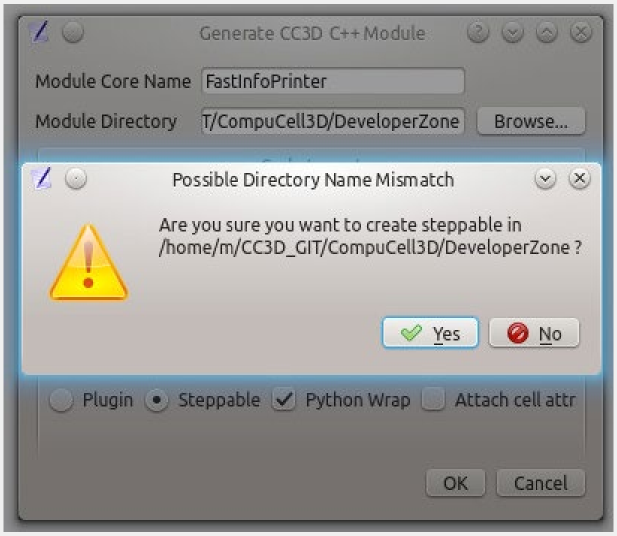
Module Core Name: FastInfoPrinter

Module Directory: T/CompuCell3D/DeveloperZone

Code Layout: ○ Main Code ● Developer Zone

C++ Module Type: ○ Plugin ● Steppable ☑ Python Wrap □ Attach cell attr

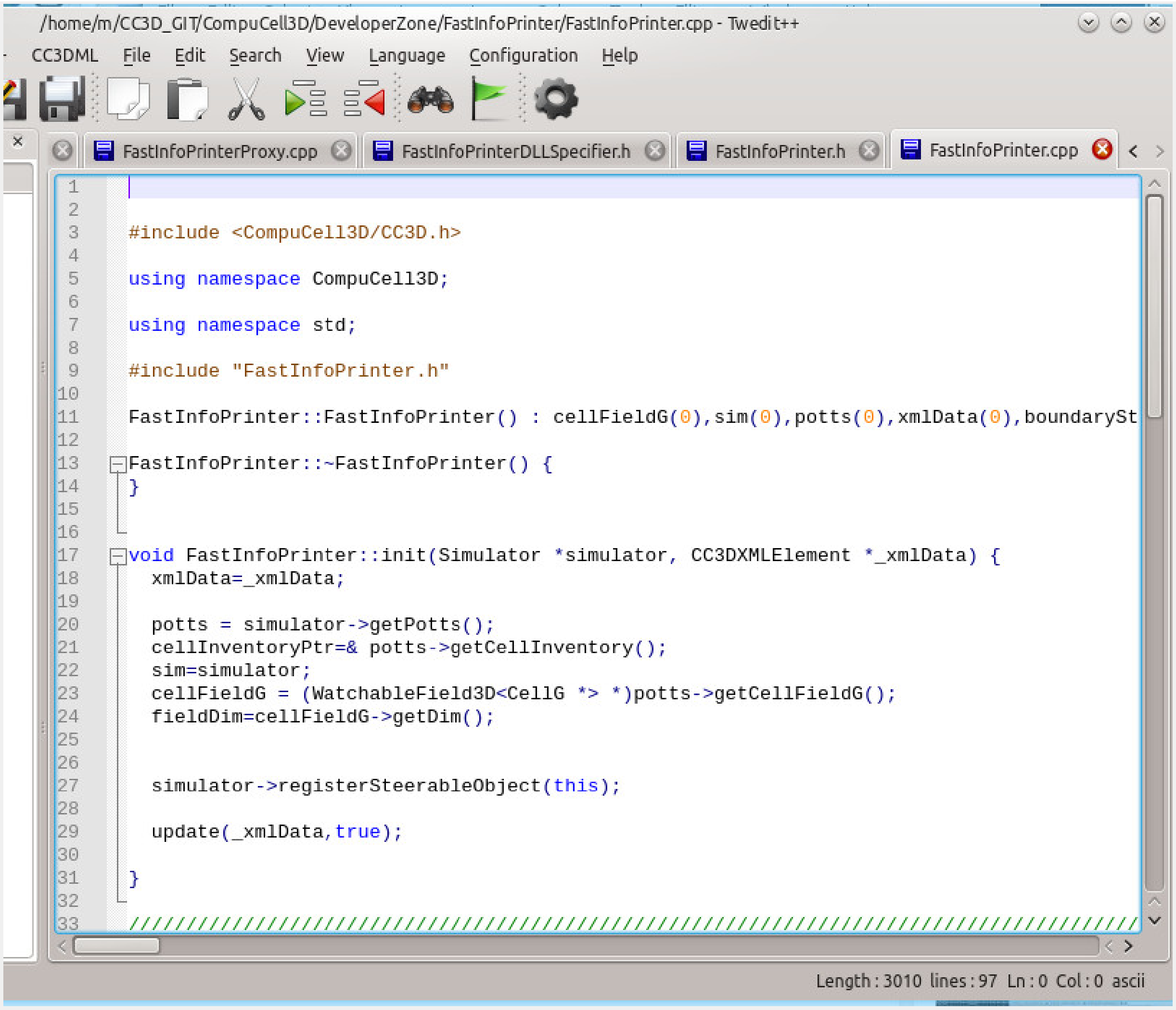
After we click OK button we will get warning dialog asking us to confirm whether we are sure that we weant to generate new code and we click **Yes** in that dialog:



Possible Directory Name Mismatch

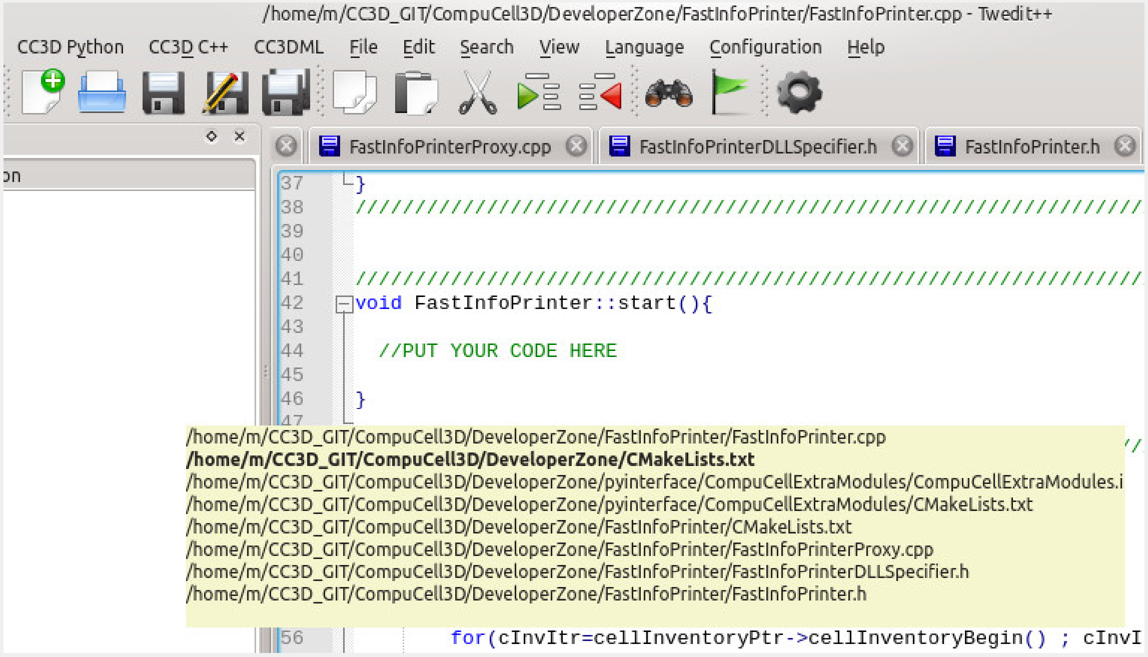
Are you sure you want to create steppable in /home/m/CC3D_GIT/CompuCell3D/DeveloperZone ?

This after completing this step Twedit++ will generate a template code for the new module:


/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinter.cpp - Twedit++
CC3DML File Edit Search View Language Configuration Help
FastInfoPrinterProxy.cpp
FastInfoPrinterDLLSpecifier.h
FastInfoPrinter.h
FastInfoPrinter.cpp
Length : 3010 lines : 97 Ln : 0 Col : 0 ascii

```

#include <CompuCell3D/CC3D.h>

using namespace CompuCell3D;

using namespace std;

#include "FastInfoPrinter.h"

FastInfoPrinter::FastInfoPrinter() : cellFieldG(0),sim(0),potts(0),xmlData(0),boundarySt

FastInfoPrinter::~FastInfoPrinter() {
}


void FastInfoPrinter::init(Simulator *simulator, CC3DXMLElement *_xmlData) {
  xmlData=_xmlData;

  potts = simulator->getPotts();
  cellInventoryPtr=& potts->getCellInventory();
  sim=simulator;
  cellFieldG = (WatchableField3D<CellG *> *)potts->getCellFieldG();
  fieldDim=cellFieldG->getDim();


  simulator->registerSteerableObject(this);

  update(_xmlData,true);

}

////////////////////////////////////////////////////////////////////////////////////////
```

In this particular case there were 7 files that were generated/modified during module addition process Pressing **Ctrl+Tab** in Twedit++ will display names of those files in the pop-up dialog:


/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinter.cpp - Twedit++
CC3D Python CC3D C++ CC3DML File Edit Search View Language Configuration Help
FastInfoPrinterProxy.cpp
FastInfoPrinterDLLSpecifier.h
FastInfoPrinter.h
on

```
}
/////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////
void FastInfoPrinter::start(){

  //PUT YOUR CODE HERE

}
```

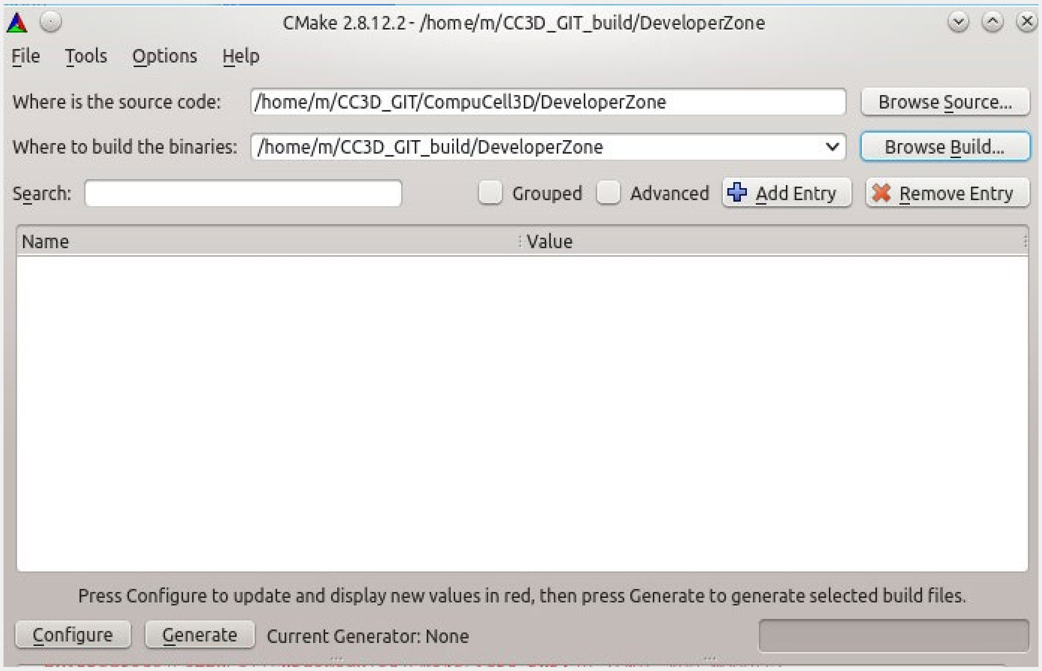
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinter.cpp
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/CMakeLists.txt
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/pyinterface/CompuCellExtraModules/CompuCellExtraModules.i
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/pyinterface/CompuCellExtraModules/CMakeLists.txt
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/CMakeLists.txt
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinterProxy.cpp
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinterDLLSpecifier.h
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinter.h

```
        for(cInvItr=cellInventoryPtr->cellInventoryBegin() ; cInvI
```

Now that our new code is generated. All we need to to is to compile it and make sure that the newly generated module gets placed in the existing CC3D installation directory. The next section describes all the steps necessary to accomplish it

Compiling Autogenerated C++ Steppable

To compile Modules in the **DeveloperZone** including the one we have just generated we use CMake and follow similar set of steps as during "regular" CC3D compilation



CMake 2.8.12.2 - /home/m/CC3D_GIT_build/DeveloperZone

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

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

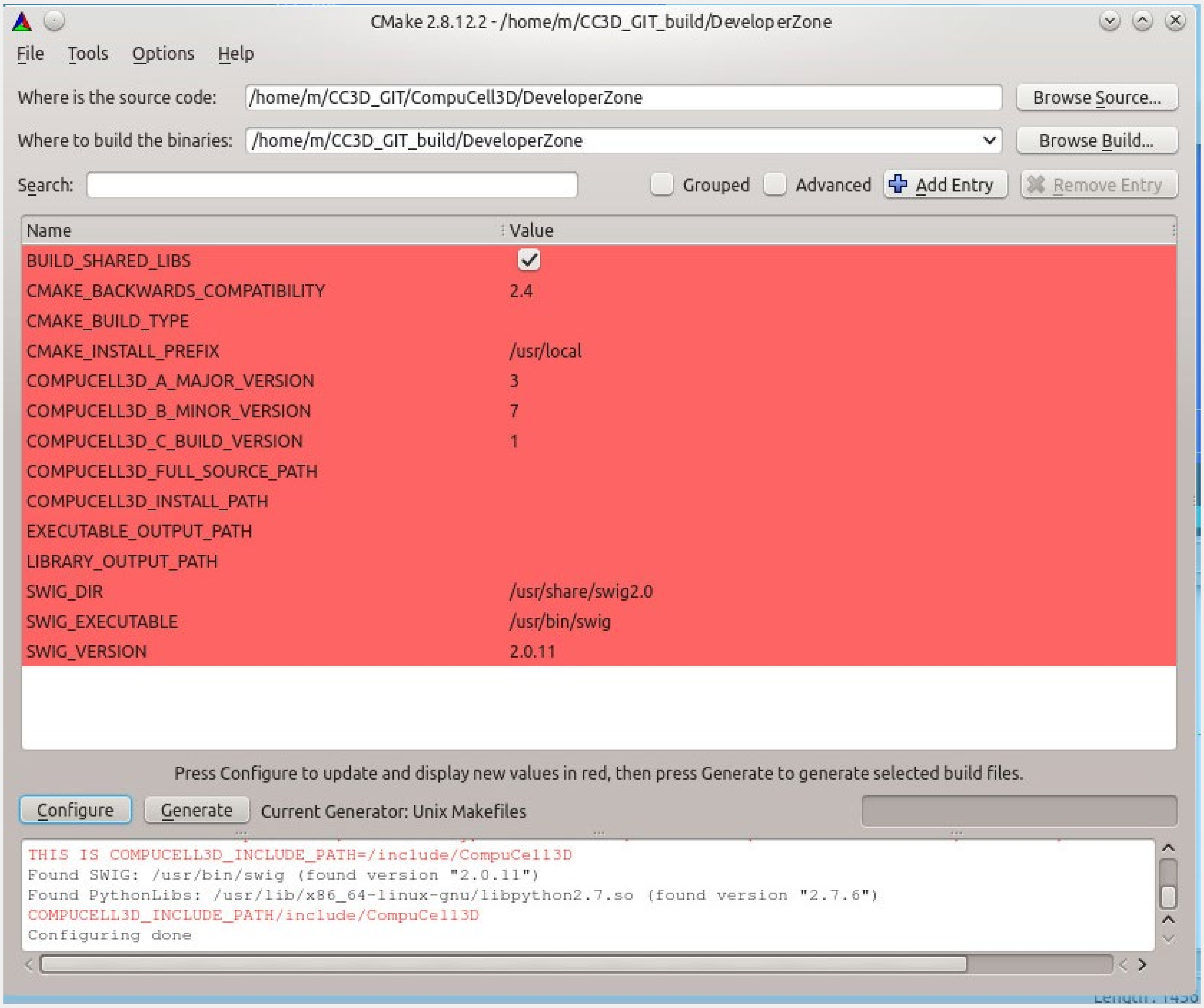
Search: □ Grouped □ Advanced

Name	Value

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

Current Generator: None

As shown in the screenshot above we select **/home/m/CC3D_GIT/CompuCell3D/DeveloperZone** as a location of source code with our new modules (**FastInfoPrinter** was generated in that folder) and select a build directory (in my case I used **/home/m/CC3D_GIT_build/DeveloperZone**). We click **Configure** in the CMake and this gets us to the next dialog:



CMake 2.8.12.2 - /home/m/CC3D_GIT_build/DeveloperZone

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

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

Search: □ Grouped □ Advanced

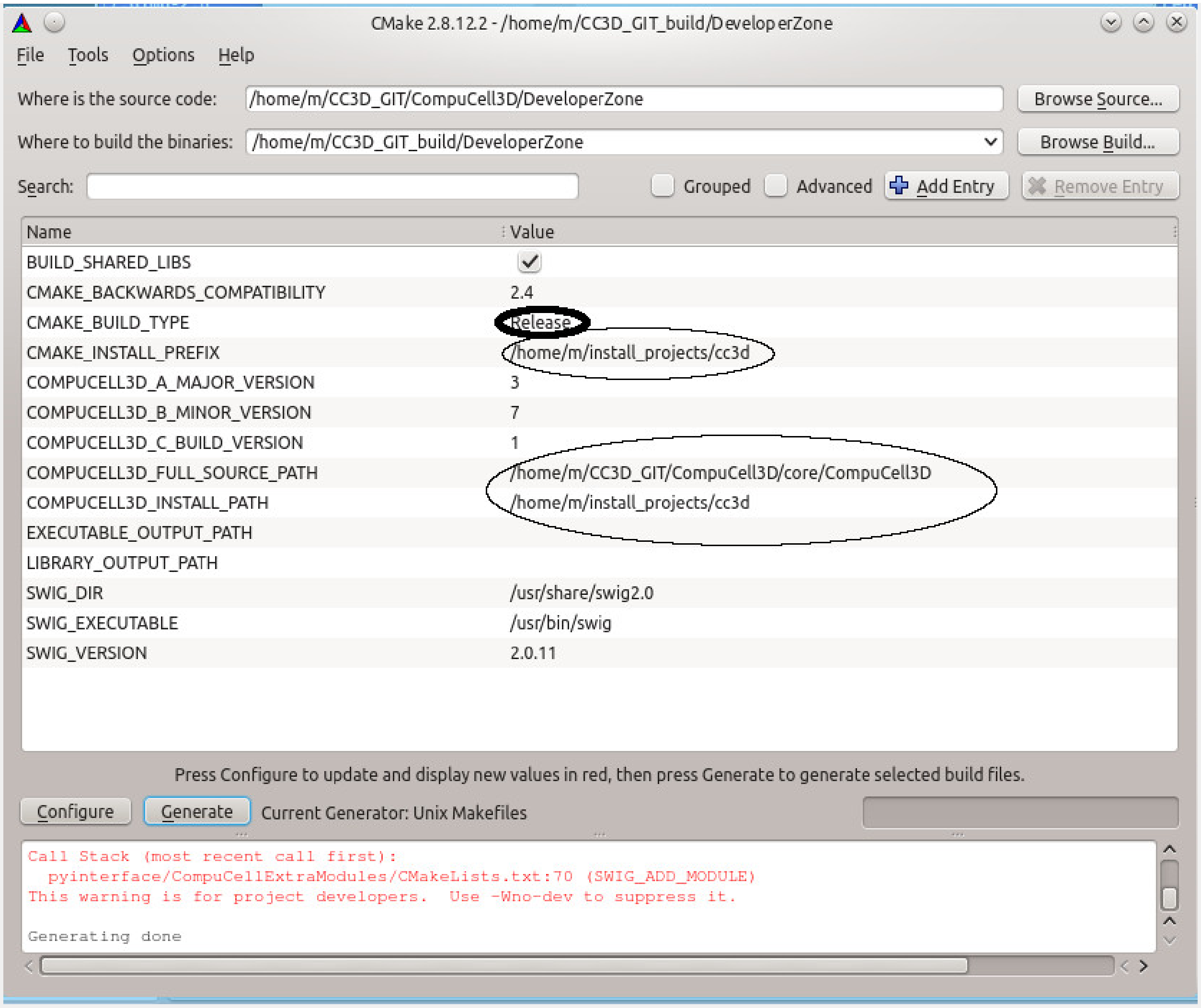
Name	Value
BUILD_SHARED_LIBS	☑
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	/usr/local
COMPUCELL3D_A_MAJOR_VERSION	3
COMPUCELL3D_B_MINOR_VERSION	7
COMPUCELL3D_C_BUILD_VERSION	1
COMPUCELL3D_FULL_SOURCE_PATH	
COMPUCELL3D_INSTALL_PATH	
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
SWIG_DIR	/usr/share/swig2.0
SWIG_EXECUTABLE	/usr/bin/swig
SWIG_VERSION	2.0.11

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

Current Generator: Unix Makefiles

```
THIS IS COMPUCELL3D_INCLUDE_PATH=/include/CompuCell3D
Found SWIG: /usr/bin/swig (found version "2.0.11")
Found PythonLibs: /usr/lib/x86_64-linux-gnu/libpython2.7.so (found version "2.7.6")
COMPUCELL3D_INCLUDE_PATH/include/CompuCell3D
Configuring done
```

Here we will need to fill several lines that describe build type , location of existing CC3d installation and the location of CompuCell3D main source tree. The screenshot belo summarizes that changes you have to make:



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

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

Name	Value
BUILD_SHARED_LIBS	✓
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	Release
CMAKE_INSTALL_PREFIX	/home/m/install_projects/cc3d
COMPUCELL3D_A_MAJOR_VERSION	3
COMPUCELL3D_B_MINOR_VERSION	7
COMPUCELL3D_C_BUILD_VERSION	1
COMPUCELL3D_FULL_SOURCE_PATH	/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D
COMPUCELL3D_INSTALL_PATH	/home/m/install_projects/cc3d
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
SWIG_DIR	/usr/share/swig2.0
SWIG_EXECUTABLE	/usr/bin/swig
SWIG_VERSION	2.0.11

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

Current Generator: Unix Makefiles

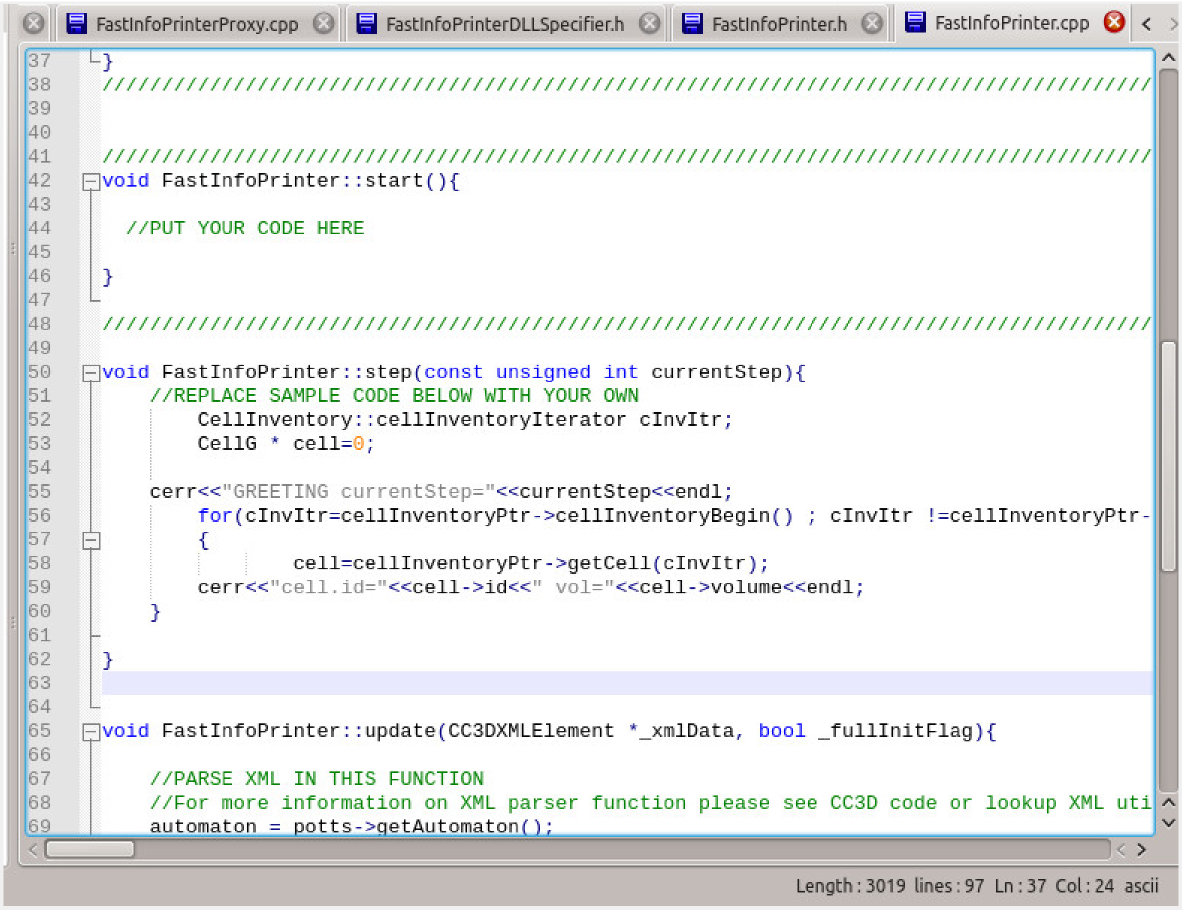
```
Call Stack (most recent call first):
  pyinterface/CompuCellExtraModules/CMakeLists.txt:70 (SWIG_ADD_MODULE)
This warning is for project developers.  Use -Wno-dev to suppress it.

Generating done
```

Let's go over those:

1. CMAKE_BUILD_TYPE - here we Type **Release** to indicate that we are building optimized version of the C++ module (you may choose **Debug** if you want to debug your steppable)
2. CMAKE_INSTALL_POREFIX - this line asks for a location of existing CC3D installation. Since I have compiled CC3D earlier and installed it into **/home/m/install_projects/cc3d** this is the directory I am picking.
3. COMPUCELL3D_FULL_SOURCE_PATH - this one is a bit tricky to guess right but this is a directory that has the following subfolders **Automaton**, **Boundary**, **Potts3D**, etc... In my case it is **/home/m/CC3D_GIT/CompuCell3D/core/CompuCell3D**. In your case it will be **<cc3d_git_dir>/CompuCell3D/code/CompuCell3D**
4. COMPUCELL3D_INSTALL_PATH - this one is exactly the same as CMAKE_INSTALL_PATH (item 1.) so we type **/home/m/install_projects/cc3d**

After putting all this information we click **Configure** followed by **Generate** and we are ready to compile our new module.

One thing to notice is that that I added a special printout to the generated code to make sure that when I run the code it is the module I generated and not some other module:

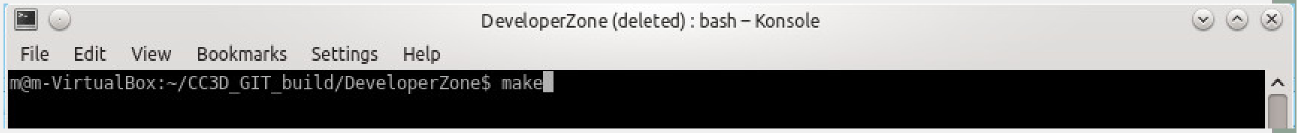

FastInfoPrinterProxy.cpp
FastInfoPrinterDLLSpecifier.h
FastInfoPrinter.h
FastInfoPrinter.cpp
Length : 3019 lines : 97 Ln : 37 Col : 24 ascii

```
}
////////////////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////////////////
void FastInfoPrinter::start(){

  //PUT YOUR CODE HERE

}

////////////////////////////////////////////////////////////////////////////////////////

void FastInfoPrinter::step(const unsigned int currentStep){
    //REPLACE SAMPLE CODE BELOW WITH YOUR OWN
        CellInventory::cellInventoryIterator cInvItr;
        CellG * cell=0;

    cerr<<"GREETING currentStep="<<currentStep<<endl;
        for(cInvItr=cellInventoryPtr->cellInventoryBegin() ; cInvItr !=cellInventoryPtr-
        {
                cell=cellInventoryPtr->getCell(cInvItr);
        cerr<<"cell.id="<<cell->id<<" vol="<<cell->volume<<endl;
    }

}


void FastInfoPrinter::update(CC3DXMLElement *_xmlData, bool _fullInitFlag){

    //PARSE XML IN THIS FUNCTION
    //For more information on XML parser function please see CC3D code or lookup XML uti
    automaton = potts->getAutomaton();
```

As you can see here I added word **GREETING** to the print statement in C++ code

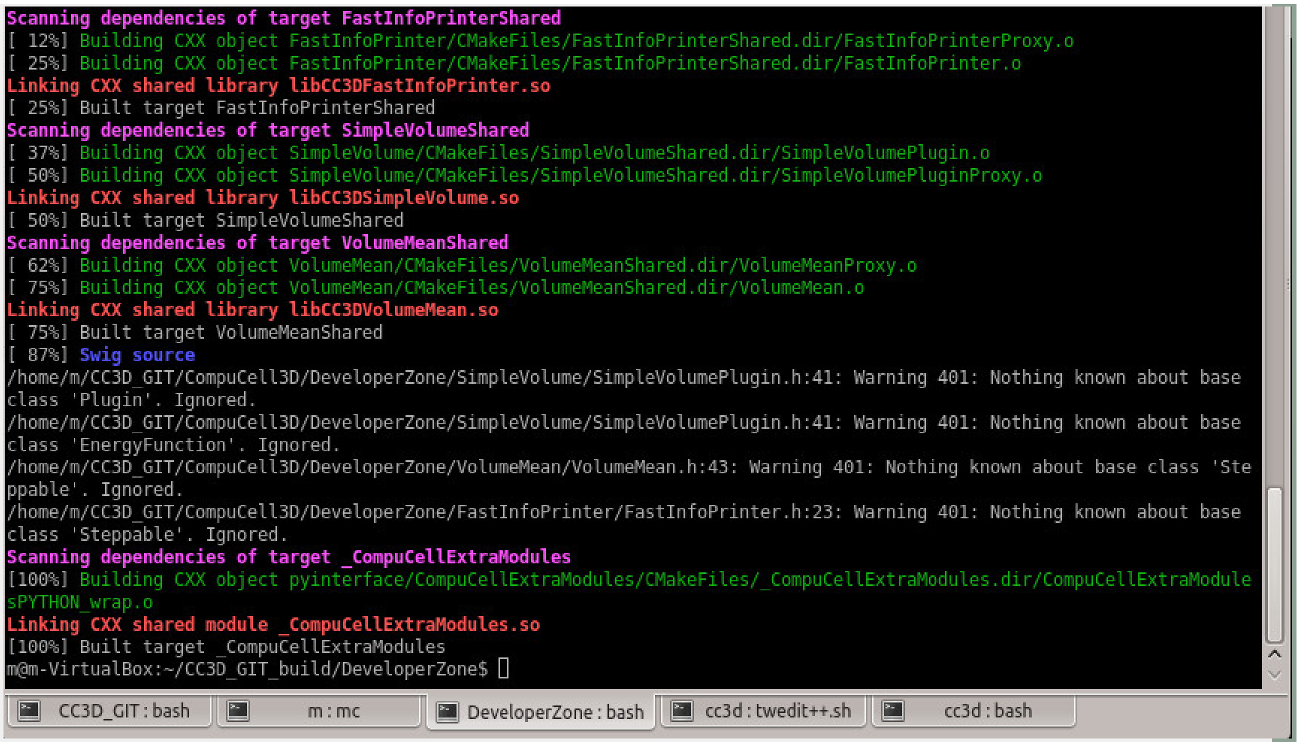
Compiling Newly Generated C++ Steppable

To compile our new code we go to the build directory (second line in the CMake Gui - **/home/m/CC3D_GIT_build/DeveloperZone**) and type **make**:



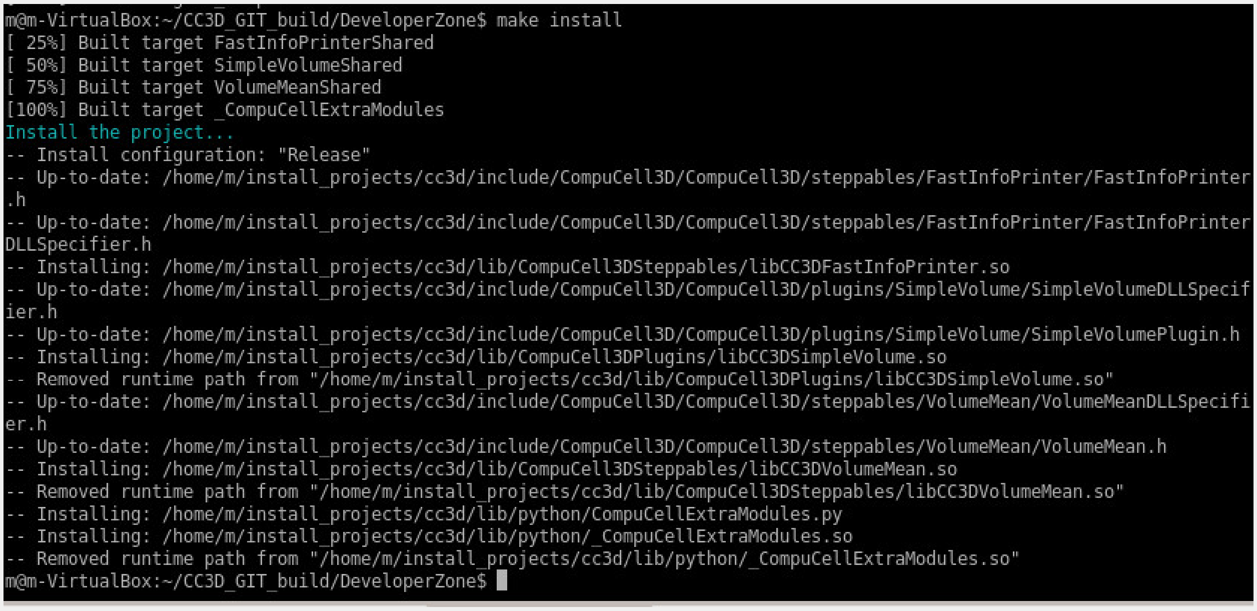
```
m@m-VirtualBox:~/CC3D_GIT_build/DeveloperZone$ make
```

After the compilation is complete you should see something like this:



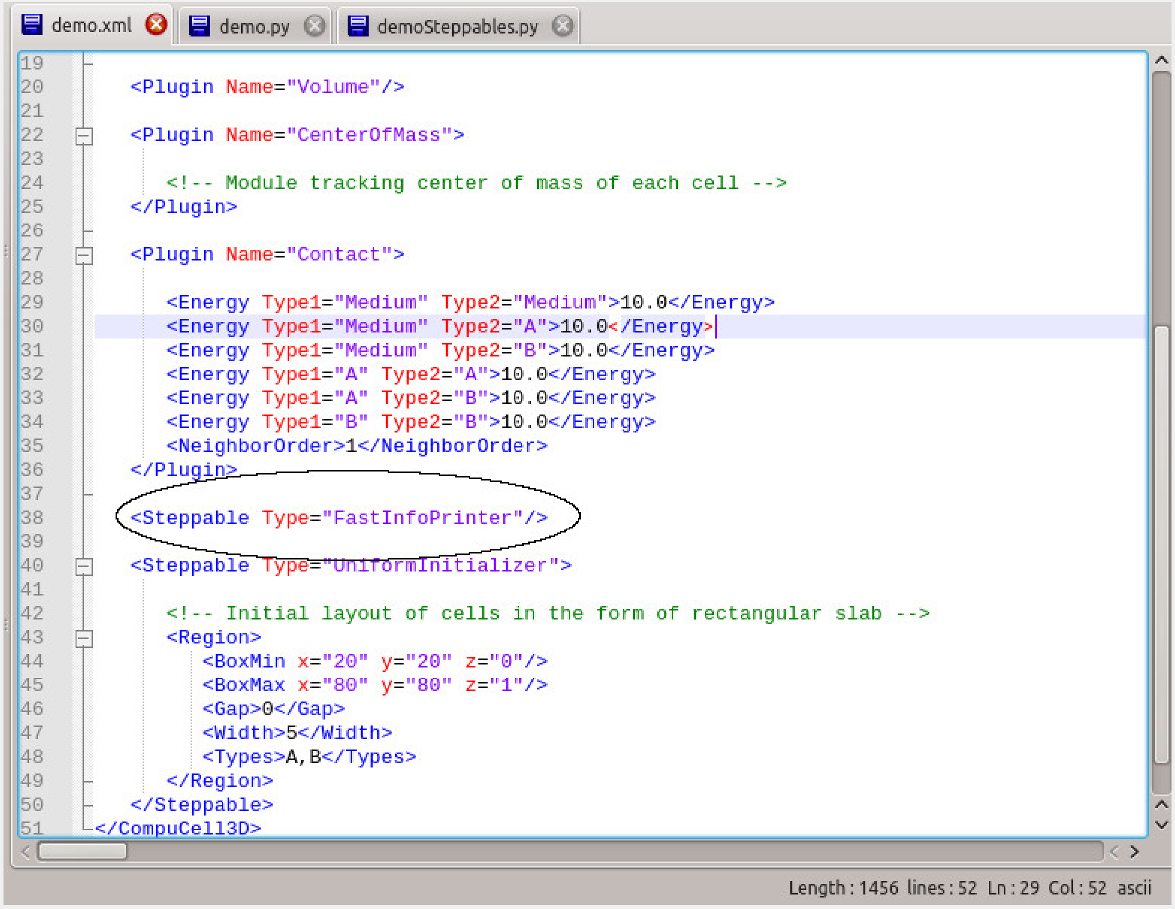
```
Scanning dependencies of target FastInfoPrinterShared
[ 12%] Building CXX object FastInfoPrinter/CMakeFiles/FastInfoPrinterShared.dir/FastInfoPrinterProxy.o
[ 25%] Building CXX object FastInfoPrinter/CMakeFiles/FastInfoPrinterShared.dir/FastInfoPrinter.o
Linking CXX shared library libCC3DFastInfoPrinter.so
[ 25%] Built target FastInfoPrinterShared
Scanning dependencies of target SimpleVolumeShared
[ 37%] Building CXX object SimpleVolume/CMakeFiles/SimpleVolumeShared.dir/SimpleVolumePlugin.o
[ 50%] Building CXX object SimpleVolume/CMakeFiles/SimpleVolumeShared.dir/SimpleVolumePluginProxy.o
Linking CXX shared library libCC3DSimpleVolume.so
[ 50%] Built target SimpleVolumeShared
Scanning dependencies of target VolumeMeanShared
[ 62%] Building CXX object VolumeMean/CMakeFiles/VolumeMeanShared.dir/VolumeMeanProxy.o
[ 75%] Building CXX object VolumeMean/CMakeFiles/VolumeMeanShared.dir/VolumeMean.o
Linking CXX shared library libCC3DVolumeMean.so
[ 75%] Built target VolumeMeanShared
[ 87%] Swig source
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/SimpleVolume/SimpleVolumePlugin.h:41: Warning 401: Nothing known about base
class 'Plugin'. Ignored.
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/SimpleVolume/SimpleVolumePlugin.h:41: Warning 401: Nothing known about base
class 'EnergyFunction'. Ignored.
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/VolumeMean/VolumeMean.h:43: Warning 401: Nothing known about base class 'Ste
ppable'. Ignored.
/home/m/CC3D_GIT/CompuCell3D/DeveloperZone/FastInfoPrinter/FastInfoPrinter.h:23: Warning 401: Nothing known about base
class 'Steppable'. Ignored.
Scanning dependencies of target _CompuCellExtraModules
[100%] Building CXX object pyinterface/CompuCellExtraModules/CMakeFiles/_CompuCellExtraModules.dir/CompuCellExtraModule
sPYTHON_wrap.o
Linking CXX shared module _CompuCellExtraModules.so
[100%] Built target _CompuCellExtraModules
m@m-VirtualBox:~/CC3D_GIT_build/DeveloperZone$ 
```

At this point we type **make install** which places our newly generated module in the CC3D installation folder - **/home/m/install_projects/cc3d**



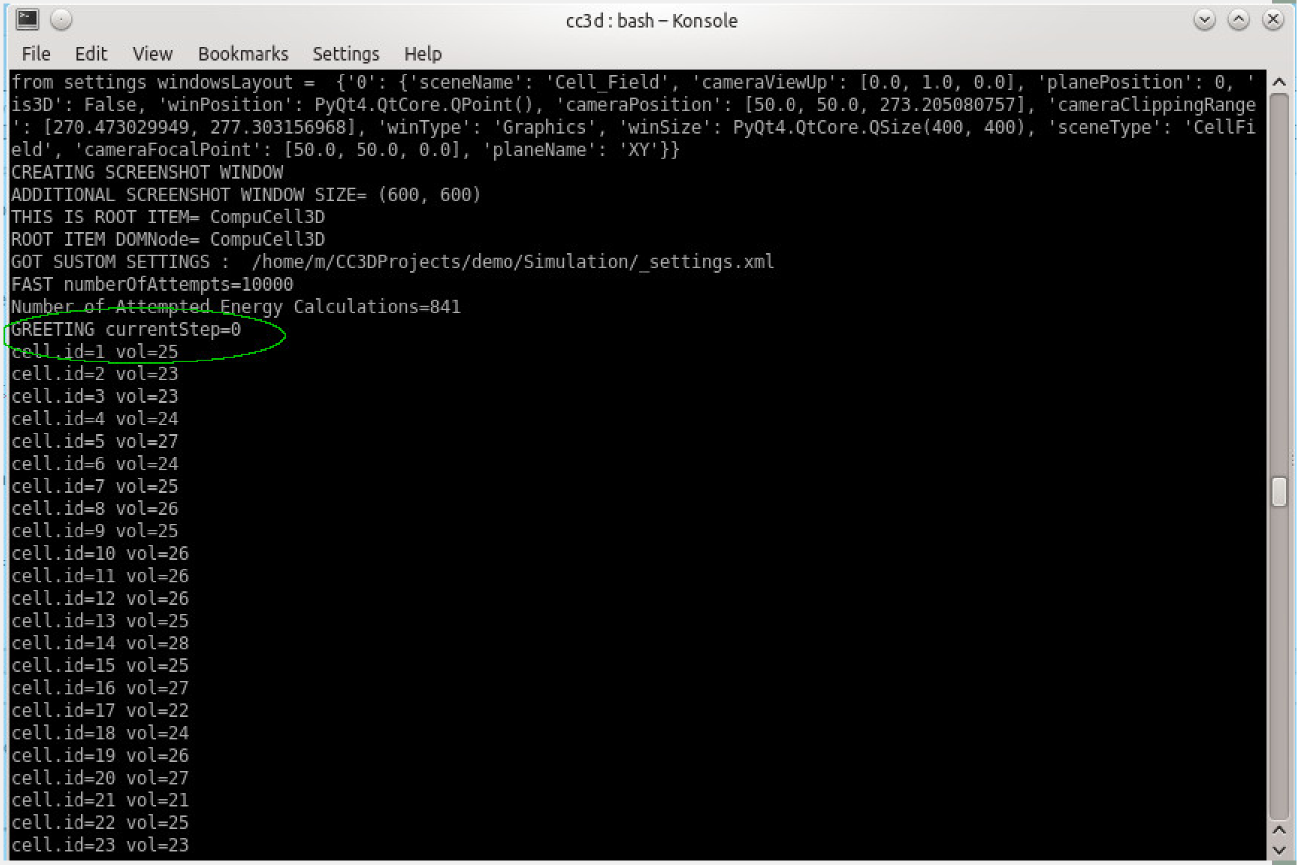
```
m@m-VirtualBox:~/CC3D_GIT_build/DeveloperZone$ make install
[ 25%] Built target FastInfoPrinterShared
[ 50%] Built target SimpleVolumeShared
[ 75%] Built target VolumeMeanShared
[100%] Built target _CompuCellExtraModules
Install the project...
-- Install configuration: "Release"
-- Up-to-date: /home/m/install_projects/cc3d/include/CompuCell3D/CompuCell3D/steppables/FastInfoPrinter/FastInfoPrinter
.h
-- Up-to-date: /home/m/install_projects/cc3d/include/CompuCell3D/CompuCell3D/steppables/FastInfoPrinter/FastInfoPrinter
DLLSpecifier.h
-- Installing: /home/m/install_projects/cc3d/lib/CompuCell3DSteppables/libCC3DFastInfoPrinter.so
-- Up-to-date: /home/m/install_projects/cc3d/include/CompuCell3D/CompuCell3D/plugins/SimpleVolume/SimpleVolumeDLLSpecif
ier.h
-- Up-to-date: /home/m/install_projects/cc3d/include/CompuCell3D/CompuCell3D/plugins/SimpleVolume/SimpleVolumePlugin.h
-- Installing: /home/m/install_projects/cc3d/lib/CompuCell3DPlugins/libCC3DSimpleVolume.so
-- Removed runtime path from "/home/m/install_projects/cc3d/lib/CompuCell3DPlugins/libCC3DSimpleVolume.so"
-- Up-to-date: /home/m/install_projects/cc3d/include/CompuCell3D/CompuCell3D/steppables/VolumeMean/VolumeMeanDLLSpecifi
er.h
-- Up-to-date: /home/m/install_projects/cc3d/include/CompuCell3D/CompuCell3D/steppables/VolumeMean/VolumeMean.h
-- Installing: /home/m/install_projects/cc3d/lib/CompuCell3DSteppables/libCC3DVolumeMean.so
-- Removed runtime path from "/home/m/install_projects/cc3d/lib/CompuCell3DSteppables/libCC3DVolumeMean.so"
-- Installing: /home/m/install_projects/cc3d/lib/python/CompuCellExtraModules.py
-- Installing: /home/m/install_projects/cc3d/lib/python/_CompuCellExtraModules.so
-- Removed runtime path from "/home/m/install_projects/cc3d/lib/python/_CompuCellExtraModules.so"
m@m-VirtualBox:~/CC3D_GIT_build/DeveloperZone$ 
```

To use our newly generated Steppable. we open up one of the CC3D simulations and in the XML file we add **FastInfoPrinter** as follows:



```xml

    <Plugin Name="Volume"/>

    <Plugin Name="CenterOfMass">

       <!-- Module tracking center of mass of each cell -->
    </Plugin>

    <Plugin Name="Contact">

       <Energy Type1="Medium" Type2="Medium">10.0</Energy>
       <Energy Type1="Medium" Type2="A">10.0</Energy>
       <Energy Type1="Medium" Type2="B">10.0</Energy>
       <Energy Type1="A" Type2="A">10.0</Energy>
       <Energy Type1="A" Type2="B">10.0</Energy>
       <Energy Type1="B" Type2="B">10.0</Energy>
       <NeighborOrder>1</NeighborOrder>
    </Plugin>

    <Steppable Type="FastInfoPrinter"/>

    <Steppable Type="UniformInitializer">

       <!-- Initial layout of cells in the form of rectangular slab -->
       <Region>
          <BoxMin x="20" y="20" z="0"/>
          <BoxMax x="80" y="80" z="1"/>
          <Gap>0</Gap>
          <Width>5</Width>
          <Types>A,B</Types>
       </Region>
    </Steppable>
</CompuCell3D>
```

When we run a simulation with **FastInfoPrinter** in it, we will see the following output:



```
from settings windowsLayout =  {'0': {'sceneName': 'Cell_Field', 'cameraViewUp': [0.0, 1.0, 0.0], 'planePosition': 0, 'is3D': False, 'winPosition': PyQt4.QtCore.QPoint(), 'cameraPosition': [50.0, 50.0, 273.205080757], 'cameraClippingRange': [270.473029949, 277.303156968], 'winType': 'Graphics', 'winSize': PyQt4.QtCore.QSize(400, 400), 'sceneType': 'CellField', 'cameraFocalPoint': [50.0, 50.0, 0.0], 'planeName': 'XY'}}
CREATING SCREENSHOT WINDOW
ADDITIONAL SCREENSHOT WINDOW SIZE= (600, 600)
THIS IS ROOT ITEM= CompuCell3D
ROOT ITEM DOMNode= CompuCell3D
GOT SUSTOM SETTINGS :  /home/m/CC3DProjects/demo/Simulation/_settings.xml
FAST numberOfAttempts=10000
Number of Attempted Energy Calculations=841
GREETING currentStep=0
cell.id=1 vol=25
cell.id=2 vol=23
cell.id=3 vol=23
cell.id=4 vol=24
cell.id=5 vol=27
cell.id=6 vol=24
cell.id=7 vol=25
cell.id=8 vol=26
cell.id=9 vol=25
cell.id=10 vol=26
cell.id=11 vol=26
cell.id=12 vol=26
cell.id=13 vol=25
cell.id=14 vol=28
cell.id=15 vol=25
cell.id=16 vol=27
cell.id=17 vol=22
cell.id=18 vol=24
cell.id=19 vol=26
cell.id=20 vol=27
cell.id=21 vol=21
cell.id=22 vol=25
cell.id=23 vol=23
```

This completes the process of generating and building C++ Compucell3D Steppable using Twedit++ and the **DeveloperZone** folder

Maintained by IU and the Biocomplexity Institute