Building OpenSim from Source

This document provides instructions for building OpenSim from source. Please note the following:

- Steps and settings may vary depending on the configuration of your system.
- Building from source is challenging and, as a non-commercial entity, we have limited resources to provide support for the build process.
- In most cases, you can use the OpenSim API without building from source (see the Developer's Guide) or by accessing the OpenSim API through Scripting. We recommend you try these approaches first!

In this document, we refer to OpenSim as the collection of C++ libraries and standalone executables distributed under the umbrella name "OpenSim". This does not include the OpenSim application GUI, and the tools necessary to build it and support it (Java, NetBeans, VTK); however, it does include the underlying SimTK dynamics engine (Simbody), downloaded from its source repository with its own build instructions.

The document is targeted to both developers on the OpenSim team who build from the GitHub repository and to end users building from a source zip distribution. If you're in the latter group please ignore sections regarding the Git repository below, as they do not apply. The source zip distribution does NOT contain the necessary files to build the MATLAB interface.

OpenSim Release	Simbody Release
OpenSim 3.3	Simbody 3.5.4
OpenSim 3.2	Simbody 3.3.1
OpenSim 3.1	Simbody 3.1
OpenSim 3.01	Simbody 3.0

If you'd like to build a more recent un-released "bleeding edge" version of OpenSim, then use the instructions at https://github.com/simbody/simbody and https://github.com/opensim-org/opensim-core.

Windows Instructions

Prerequisites

Category	Software	How to obtain
Operating system	Windows XP, 7, 8, 8.1	
Cross-platform build system	CMake >= 2.8	Cmake Download
Compiler / Integrated Development Environment	• Microsoft Visual Studio 2010 (OpenSim 3.2 and earlier) • Microsoft Visual Studio 2013 (OpenSim 3.2 and later) • Microsoft Visual Studio 2015 (OpenSim 3.3 and later) • Microsoft Visual Studio Community (same versions)	Dreamspark (academic licence) MS Visual Studio Express/Community (Free):
Python access to OpenSim API	• SWIG 2.0.12 (not 3.0+) • Python 2.7 (tested with 2.7.10)	swigwin-2.0.12 python-2.7.10 (be sure to get 32 or 64 bit to match your OpenSim build)
Git repository and version control (optional)	Any of the following: • GitBash • TortoiseGit • GitHub for Windows	Git Bash (Git Shell) TortoiseGit GitHub for windows

Build Simbody

You must build and install Simbody first. The instructions below should be sufficient; if you need more detail see the README.md file for the appropriate Simbody branch on GitHub at https://github.com/simbody/simbody (Simbody 3.5.3 build instructions at GitHub).

1. Create a local Simbody directory to hold the Simbody source code (e.g. C:/Projects/Simbody)

2. Obtain the Simbody source code from its GitHub repository . Look for "releases" and get the right source zip file (3.3.1 for use with OpenSim 3.2, 3.5.3 with OpenSim 3.3). Unzip the Simbody source into your chosen directory; this will create a subdirectory with a name like *simbody-Simbody3.5.3*.
3. Launch CMake:
    a. **Where is the source code?** Choose the Simbody source code directory you created in step 2 (e.g. C:/Projects/Simbody/simbody-Simbody3.5.3).
    b. **Where to build the binaries?** Choose a build location (e.g. C:/Projects/Simbody/build).
4. Press the **Configure** button (near the bottom). Let CMake create the build directory. Choose a “generator” that best matches the compiler you are using, for example “Visual Studio 12 2013”, or "Visual Studio 12 2013 Win64" to make 64-bit binaries. Select “Use default native compilers” and click Finish.
5. Specify an installation location for Simbody (e.g. **CMAKE_INSTALL_PREFIX** = C:/Projects/Simbody/install). (Older versions used SimTK_INSTALL_PREFIX for this purpose.)
6. Hit **Configure** again. There should be nothing in red now and the **Generate** button will be available.
7. Hit **Generate**, which will create the Visual Studio solution files in the Simbody build directory (e.g., C:/Projects/Simbody/build).
8. Go to the directory that you specified as your Simbody build directory. Click on *Simbody.sln* to launch it in Visual Studio.
9. If you want a Debug build, build it first. Select the Debug configuration (selection in the top center of the screen). Otherwise use RelWithDebInfo.
10. Build the project **ALL_BUILD** (right click and select "Build").
11. Build **RUN_TESTS** to make sure everything built correctly.
12. Build the project **INSTALL** to make a local installation of binaries (dll’s and exe’s) and documentation (into, e.g., C:/Projects/Simbody/install).
13. If you built a Debug configuration, now repeat steps 10 to 12 in the “RelWithDebInfo” configuration (right click project - Properties - Configuration drop down box). While the debug configuration is very useful for testing and debugging, it will be too slow to use for most purposes. The production configuration we recommend is RelWithDebInfo. That runs just as fast as Release, but keeps extra debugging information around that can be helpful if you encounter a problem.

Build OpenSim

Building OpenSim consists of three primary steps; obtain the source code, generate solution file, then build binaries.

1. Obtain OpenSim Source Code

- You can download the OpenSim 3.x source code in a zip file from the OpenSim download page. As of 3.0, this is the main way we are distributing the OpenSim source code.
- Unzip the source to a project folder on your machine (e.g. C:/Projects/OpenSim3x). This will create a subdirectory with a name like *OpenSim33-source*.

2. Use CMake to Build the Visual Studio Solution (.sln) File

1. Launch CMake:
    a. **Where is the source code?** Choose the OpenSim source code directory you specified (e.g. C:/Projects/OpenSim3x/OpenSim33-source). This designates the folder containing the top-level CMake file named CMakeLists.txt.
    b. **Where to build the binaries?** Choose a build location (e.g. C:/Projects/OpenSim3x/build). This designates the folder for compiler files, debug and release sub-folders, etc.
2. Press the Configure button (near the bottom). Let CMake create the build directory. Choose a “generator” that best matches the compiler you are using, for example “Visual Studio 12 2013 Win64”. **Important**: your generator choice must be the same as you used when you built Simbody. Select “Use default native compilers” and click Finish.
3. To avoid building the command line utilities and associated tests (faster), check BUILD_API_ONLY. This is an Advanced Option in the CMake GUI.
4. Specify an installation location for OpenSim (e.g. CMAKE_INSTALL_PREFIX = C:/OpenSimLocal). You should choose a location for which you have write permission.
5. To build python wrapping, set BUILD_PYTHON_WRAPPING to ON.
6. Verify that BUILD_USING_NAMESPACE is set to the same value used when building Simbody (Blank is the default).
7. Point SIMBODY_HOME (SimTK_INSTALL_DIR in earlier versions) to where you installed Simbody (e.g. C:/Projects/Simbody/install).
    a. After you hit configure again, make sure Simbody_DIR have been set based on SIMBODY_HOME (these are shown only in Advanced View in CMake)
8. Hit Configure again. This time there should be nothing in red and the Generate button will be available.
9. Hit "Generate", which will create the Visual Studio solution files in the OpenSim build directory

3. Build Binaries and Install

1. Click on *OpenSim.sln* from the OpenSim build directory to launch Visual Studio.
2. If you want to build Debug libraries, make sure you are in "Debug" configuration (selection in the top center of the screen). Otherwise use RelWithDebInfo.
3. Build the project ALL_BUILD (right click and select "Build").
4. Build RUN_TESTS to make sure everything built correctly. Some tests may time-out in the Debug configuration.
5. Build the project INSTALL to make a local installation of binaries (dll’s and exe’s) and documentation. The INSTALL may fail if you do not have write permission for the OpenSim install location that you chose in CMake. Running Visual Studio as Administrator (right click on Visual Studio and select "Run as Administrator") should address any permission issues. Note that if you start Visual Studio as Administrator, you will need to navigate to the OpenSim.sln file rather than double clicking it.
6. Repeat steps 2 to 5 in the “RelWithDebInfo” configuration (right click project - Properties - Configuration drop down box). While the debug configuration is very useful for testing and debugging, it will be too slow to use for most purposes. The production configuration we recommend is RelWithDebInfo, which runs just as fast as Release, but keeps extra debugging information around that can be helpful if you encounter a problem.
7. Ensure the ${install directory}/bin folder is on a path specified in your computer’s PATH environment variable, (e.g., ensure PATH includes C:\OpenSimLocal\bin)

8. If you turned on BUILD_PYTHON_WRAPPING above, you can install the "opensim" python package by opening a command window, navigating to C:/OpenSimLocal/sdk/python and running "python setup.py install" from a Command Prompt. See this page for more information and some troubleshooting tips. If you are having trouble, please post to the OpenSim forum.

Mac Instructions

Prerequisites

Category	Software	How to obtain
Operating system	OS X 10.6 and above	
Cross-platform build system	CMake >= 2.8	Cmake Download
Compiler / Integrated Development Environment	Any of the following: • Xcode >=4 (uses Clang) • Clang >=3.0 • gcc >=4.2.1	OS X App store
Git repository and version control (optional)	git GitHub for Mac	Git comes with Xcode Command-line tools. GitHub for Mac

Special Note

You will need a clean directory for downloading, extracting, configuring, and compiling code. We highly recommend choosing a directory with no spaces in its file path because some configuration and compilation tools do not handle whitespace well. For example "/Users/Me/DevProjects /OpenSim3.3" should cause no problems for cmake and GNU make, but something like "/Users/Me/Dev Projects/OpenSim 3.3", which contains spaces, could cause trouble for some development environments.

Install Simbody

- You must build and install Simbody first. Starting with Simbody 3.5, detailed instructions for building/installing Simbody are located in Simbody's README: https://github.com/simbody/simbody(Simbody 3.5.4 build instructions at GitHub). If you need an earlier version of Simbody, go down to the Linux instructions and use the instructions there for building Simbody.
  - If you are using Xcode 8 or greater (more specifically, MacOSX10.12.sdk or greater), or your computer is running macOS 10.12 (or greater), you must use Simbody 3.5.4 or greater, as this version contains a vital fix.
  - **Important:** For Simbody 3.5, you should set the Simbody CMake variable SIMBODY_STANDARD_11 to OFF before building Simbody; this disables the use of C++11 features. Otherwise, you will get mysterious errors. Alternatively, if you want to use C++11 features with OpenSim (3.3 only), leave SIMBODY_STANDARD_11 set to ON; when building OpenSim, set OPENSIM_STANDARD_11 to ON.

1. Obtain OpenSim Source Code

- You can download the OpenSim 3.x source code in a zip file from the OpenSim downloads page.
- Unzip the source to a project folder on your machine (e.g. ~/simtk/OpenSim3x).

2. Use CMake to generate an Xcode project or Makefiles.

- Launch CMake:
  1. "Where is the source code?" Choose the OpenSim source code directory you specified (e.g. ~/simtk/OpenSim3x).
  2. "Where to build the binaries?" Choose a build location (e.g. ~/simtk/OpenSim3x-build).
- Press the Configure button (near the bottom). Let CMake create the build directory. A window will open asking you to specify a generator. Choose either Xcode or Unix Makefiles, depending on your preference.
- Specify an installation location for OpenSim (e.g. CMAKE_INSTALL_PREFIX = ~/opensim3x).
- Point SIMBODY_HOME to where you installed Simbody (e.g. ~/simbody).
- Specify OPENSIM_STANDARD_11 to be on or off, depending on the Simbody build settings you used (see comment in Simbody installation directions above).
- Hit Configure again. This time there should be nothing in red and the Generate button will be available.
- Hit "Generate". Based on which generator you specified earlier, this will create either the Xcode project or the necessary Makefiles in the OpenSim build directory.

3. Build Binaries and Install

- If using Xcode:
  1. Launch OpenSim.xcodeproj from the OpenSim build directory.
  2. Follow the instructions here to properly set your build configuration: https://github.com/opensim-org/opensim-core#build-and-install-1
  3. Build the target ALL_BUILD.
  4. Build the target INSTALL to make an installation of binaries (dylib's and executables) and documentation.
- If building using Makefiles:
  1. Launch a terminal window and navigate to the OpenSim build directory (e.g. cd ~/simtk/OpenSim3x-build).
  2. Build the binaries (the -jn flag lets you parallelize the build using n processor cores).

        - >> make -j8
3. Install OpenSim onto your computer.
    - >> make -j8 install
4. Make sure the build and installation was successful by running the tests.
    - >> ctest -j8
    - Note: prior to doing this, it may be necessary to first set the library path (see instructions below).

Notes

- Set the DYLD_LIBRARY_PATH variable so that the SimTK and OpenSim libraries can be found. Do this by running the following command in a terminal:
    - export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:<OpenSim install dir>/lib
- The above instruction only works in that current terminal session. To modify your DYLD_LIBRARY_PATH for all future terminal sessions, add the above line to the file /etc/profile on your system. Restart your terminal and check that DYLD_LIBRARY_PATH contains the OpenSim lib path by running:
    - echo $DYLD_LIBRARY_PATH
- Set OPENSIM_HOME to your installation directory:
    - export OPENSIM_HOME=<OpenSim install dir>

Linux Instructions

Prerequisites

Category	Software	How to obtain	Notes
Operating system	Ubuntu 12.04 and above		Tested on Ubuntu 14.04.
Cross-platform build system	CMake >= 2.8	Cmake Download	
Compiler / Integrated Development Environment	Any of the following: • Clang >=3.3 • gcc >=4.8	apt-get clang. Comes with gcc.	Tested with: • clang-3.6 • gcc-5.1
Git repository and version control (optional)	git svn >= 1.7	apt-get git subversion	
Python wrapping	Python 2.7	apt-get python2.7-dev	

Install Simbody

- You must build and install Simbody first. Starting with Simbody 3.5, detailed instructions for building /installing Simbody are located in Simbody's README: https://github.com/simbody/simbody (Simbody 3.5.3 build instructions at GitHub). If you need an earlier version of Simbody, expand the instructions below.
    - **Important:** If using Simbody 3.5, you should set the Simbody CMake variable SIMBODY_STANDARD_11 to OFF before building Simbody; this disables the use of C++11 features. Otherwise, you will get mysterious errors. Alternatively, if you want to use C++11 features with OpenSim (3.3 only), leave SIMBODY_STANDARD_11 set to ON; when building OpenSim, set OPENSIM_STANDARD_11 to ON.

1. Create a local Simbody directory to hold the Simbody source code (e.g. ~/simtk/Simbody331)
2. Obtain the Simbody source code from its GitHub repository . Look for "releases" and get the right one (3.3.1 for use with OpenSim 3.2).
3. Launch the CMake GUI.
    a. **Where is the source code?** Choose the Simbody source code directory you specified in step 1 (e.g. ~/simtk/Simbody331).
    b. **Where to build the binaries?** Choose a build location (e.g. ~/simtk/Simbody331-build).
4. Press the **Configure** button (near the bottom). Let CMake create the build directory. Choose **Unix Makefiles**.
5. Specify an installation location for Simbody (e.g. **CMAKE_INSTALL_PREFIX** = ~/simbody). (Older versions used SimTK_INSTALL_PREFIX for this purpose.)
6. If you want a Debug build, build it first. Select the Debug configuration by setting **CMAKE_BUILD_TYPE** to **Debug**. Otherwise set **CMAKE_BUILD_TYPE** to **RelWithDebInfo**.
7. Hit **Configure** again. This time there should be nothing in red and the **Generate** button will be available.
8. Hit **Generate**, which will setup the Simbody build directory (e.g., ~/simtk/Simbody331-build).
9. Open a terminal and change directories to the Simbody build directory.
10. Build the binaries (the -jn flag lets you build using n processor cores). For example, the terminal command to build with 8 processor cores is as follows
    - make -j8
11. Run the tests.
    - ctest -j8
12. Install Siimbody.
    - make -j8 install
13. If you built a Debug configuration, now repeat steps 10 to 12 in the **RelWithDebInfo** configuration. While the debug configuration is very useful for testing and debugging, it will be too slow to use for most purposes. The production configuration we recommend is **RelWithDebInfo**. That runs just as fast as Release, but keeps extra debugging information around that can be helpful if you encounter a problem.

Build OpenSim

1. Obtain OpenSim Source Code

- You can download the OpenSim 3.x source code in a zip file from the OpenSim downloads page.
- Unzip the source to a project folder on your machine (e.g. ~/simtk/OpenSim32).

2. Use CMake to generate Makefiles.

- Launch the CMake GUI.
    1. "Where is the source code?" Choose the OpenSim source code directory you specified (e.g. ~/simtk/OpenSim32).
    2. "Where to build the binaries?" Choose a build location (e.g. ~/simtk/OpenSim32-build).
- Press the Configure button (near the bottom). Let CMake create the build directory. To build python wrapping, turn on BUILD_PYTHON_WRAPPING.
- Specify an installation location for OpenSim (e.g. CMAKE_INSTALL_PREFIX = ~/opensim32). It is preferable to choose a location where sudo access is not required.
- Point SIMBODY_HOME to where you installed Simbody (e.g. ~/simbody).
- Hit Configure again. This time there should be nothing in red and the Generate button will be available.
- Hit "Generate", which will setup the OpenSim build directory.

3. Build Binaries and Install

- Open a terminal and change directories to the OpenSim build directory.
    - cd ~/simtk/OpenSim32-build
- Build the binaries (the -jn flag lets you build using n processor cores).
    - make -j8
- Run the tests.
    - ctest -j8
- Install OpenSim onto your computer (e.g., to ~/opensim32).
    - make -j8 install

Notes

- Set the LD_LIBRARY_PATH variable so that the SimTK and OpenSim libraries can be found. Do this by running the following command in a terminal:
    - export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<OpenSim install dir>/lib
- The above instruction only works in that current terminal session. To modify your LD_LIBRARY_PATH for all future terminal sessions, add the above line to the file ~/.bashrc on your system. Restart your terminal and check that LD_LIBRARY_PATH contains the OpenSim lib path by running:
    - echo $LD_LIBRARY_PATH
- Set OPENSIM_HOME to your installation directory:
    - export OPENSIM_HOME=~/opensim32

Troubleshooting

- Check out the instructions at the Simbody Project Site. These documents have lots of helpful information about setting up your system to build, getting cmake, etc.
- Windows has two PATH environment variables, one under "User variables" and another under "System variables." OpenSim\bin must be prepended to the PATH environment variable under "System variables," otherwise you may receive runtime errors.
- On Windows you can't mix Release and Debug libraries, make sure if you're building debuggable OpenSim to have Simbody debug built as well, same for Release.
- One key issue you need to be aware of is the possible interaction between a development environment and installed versions of OpenSim on the same machine. This can happen if you allow the OpenSim installer to add the install directory/bin to the path and have it ahead of the path to the dynamic libraries that you build/install yourself. The symptom is that you'll see some random unexpected behavior and/or crashes. Things to do to minimize these possible interactions:
    1. Do not add the install directory of OpenSim to your "Path" environment variable (or remove it if it's already there). The **GUI DOES NOT NEED** path setting but command line tools do. If you want to know what's on your path, open a command prompt window and type "path".
    2. Make debug builds only from source. Debug libraries have different names than release libraries (with trailing _d for debug), so there's no possibility of collision/mixup. The code runs an order of magnitude slower in some cases but should be good enough for troubleshooting which is the most time consuming task.
    3. Be aware of library name conventions, for example the simtk libraries included with OpenSim distributions come with a prefix OpenSim_ you can configure your libraries to use or not use a prefix by setting the "BUILD_USING_NAMESPACE" CMake variable, if you do please sure it's set consistently in both Simbody and OpenSim builds.