How to build Simbody (and SimTK) 2.1 from source on Windows

Michael Sherman, 19 Aug 2010

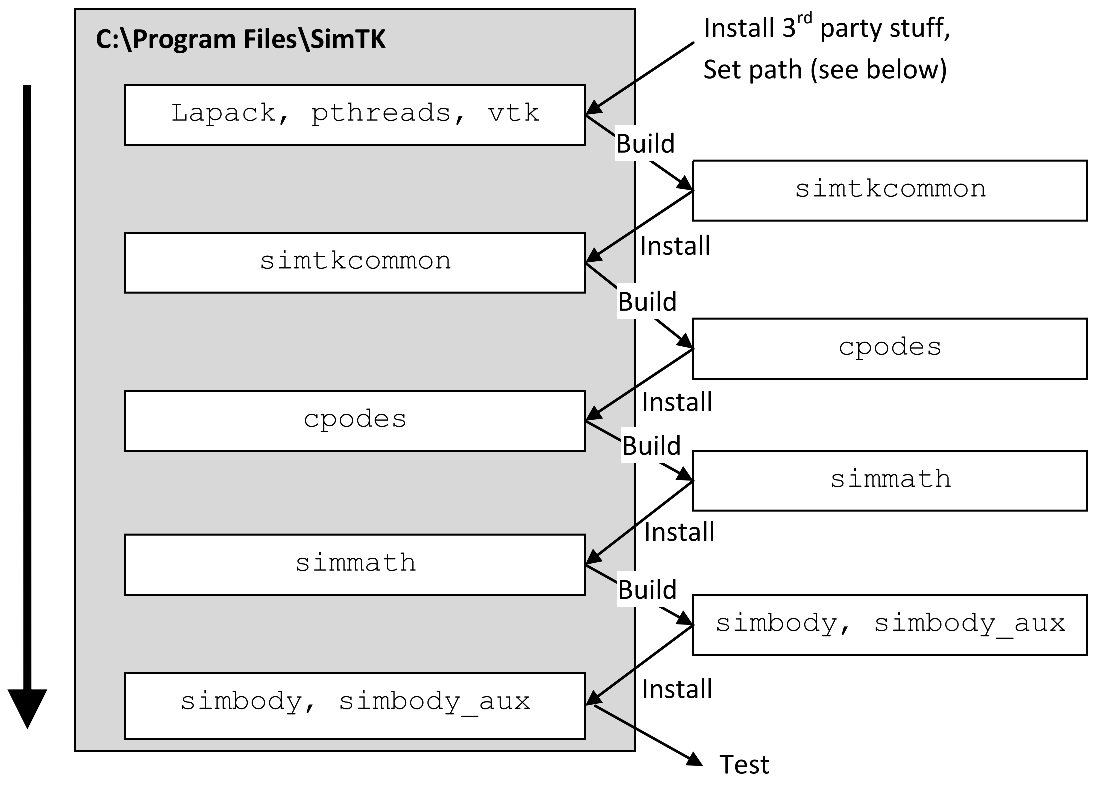
Here is the build-from-source procedure as it currently stands. We are going to reorganize the source for the next version to make this take fewer steps. But meanwhile:

Outline

- You will need CMake 2.8, Visual Studio 2005 or later, and a Subversion (SVN) client. You should have doxygen so you can generate nice API docs.
- Install 3rd party binaries Lapack, pthreads, vtk (optional).
- Check out each library's source code from Subversion (four checkouts required: simtkcommon, cpodes, simmath, simbody).
- Run CMake to create four Visual Studio solution files.
- Build/test/install four libraries in a particular order.
- Test installation.

[There is a fifth library, molmodel, for molecular simulation. It is built similarly but not covered here.]

Here's a picture of this process, assuming you are going to install SimTK in C:\Program Files\SimTK (you can actually put it anywhere but that's the default so it's easiest to install there if you can):



So that executables can find the DLLs, your path needs to include C:\Program Files\SimTK\lib and C:\Program Files\SimTK\bin.

Step 1: acquire needed goodies

If you don't have them already, find and install these tools:

- Visual Studio C++ 8 (2005), 9 (2008), or Visual Studio 9 Express. Visual Studio 10 should also work but we haven't tested it yet; if you have any trouble with it please let us know.
- CMake 2.8.1 or later (cmake.org)
- Doxygen 1.6.1 or later (doxygen.org)
- A subversion client; I'll assume below that you have Tortoise SVN (tortoisesvn.tigris.org) which is a very nice GUI-based client that integrates itself into Windows Explorer.
- You should also have a command line client so CMake can use it to extract version number information; we have used the CollabNet command line client that you can get here: http://www.open.collab.net/downloads/subversion/; be sure to download "CollabNet Subversion Command-Line Client (for Windows)" not one of the other options there. You can build without this but CMake will complain SVNVERSION_NOTFOUND.

Step 2: install 3rd party binaries

We are going to need three sets of binaries:

- Lapack and blas
- Pthreads (needed for windows)
- VTK (optional, but used by the simbody_aux library if you build it, which we recommend, at least for initial testing)

There is a very easy way to get all these binaries at once: download and install a binary version of SimTK from https://simtk.org/home/simtkcore, Downloads tab. Get the Windows version that best matches your version of Visual Studio and install it. Either SimTK 2.0 or SimTK 2.1 binaries will work fine for this purpose since these are very stable. This will install the SimTK version of Lapack and blas (called SimTKlapack), pthreads, and an older but functional version of VTK (5.0). You can use different Lapack and VTK binaries if you want by making suitable choices while running CMake, but we're not going to address that here.

So: download and install a SimTK 2.0 or SimTK 2.1 binary. Allow the installer to set your path or manually set it to include C:\Program Files\SimTK\lib and \bin; replace "C:\Program Files\SimTK" with your installation directory if necessary. We are going to overwrite the contents of this installation directory as we build SimTK from source.

Make sure your path is set before going to the next step!

Step 3: check out the source code

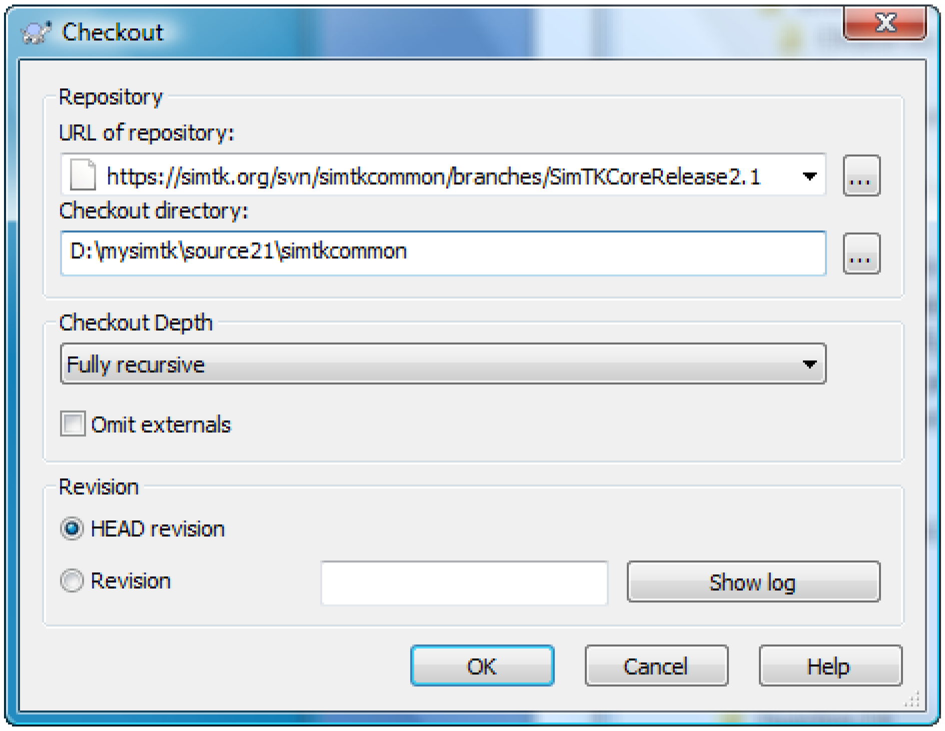
I'm assuming you are using TortoiseSVN as your SVN client. You'll have to modify these instructions if you're using a different client but the steps will be the same.

1. Create a directory (folder) to contain the source and binaries while we're building. This should *not* be in the installation directory. Here I'm going to assume you called it "mysimtk".
2. Create subdirectories mysimtk/source21 and mysimtk/bin. All four projects' sources will go in mysimtk/source21 (in four subdirectories), and we'll tell CMake to put all the binaries under mysimtk/bin. **Warning**: don't use "src" in the name for your top-level source directory because our Doxygen configuration excludes path names matching "*/src/*" so that only include files are used to generate API docs; you'll get blank Doxygen documentation if you use "src" in the path name (src21 would be OK).
3. With Windows Explorer, go to the (empty) mysimtk/source21 folder. Right-click and select "SVN Checkout…".
4. In the dialog window that appears, set the following values for the text fields:

   URL of repository: https://simtk.org/svn/simtkcommon/branches/SimTKCoreRelease2.1
   Checkout directory: …\mysimtk\source21\simtkcommon

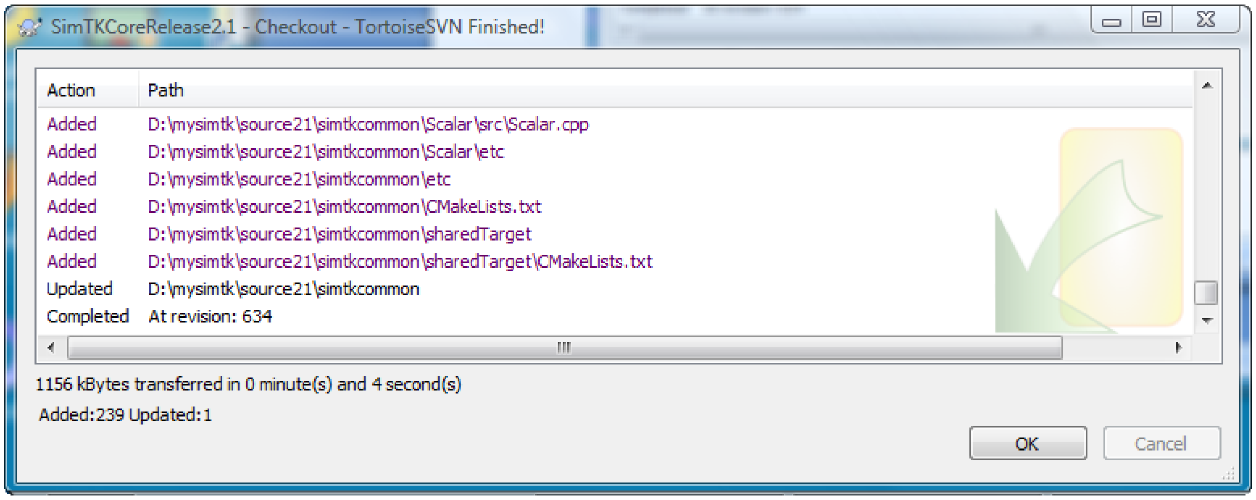
   where the "…" should contain the location at which you created mysimtk (should already be filled in by Tortoise)

Here's what it looked like on my machine:



URL of repository: https://simtk.org/svn/simtkcommon/branches/SimTKCoreRelease2.1
Checkout directory: D:\mysimtk\source21\simtkcommon
Checkout Depth: Fully recursive
Omit externals: ☐
Revision: ● HEAD revision ○ Revision

Then you should see the files being checked out, with the end looking something like this:

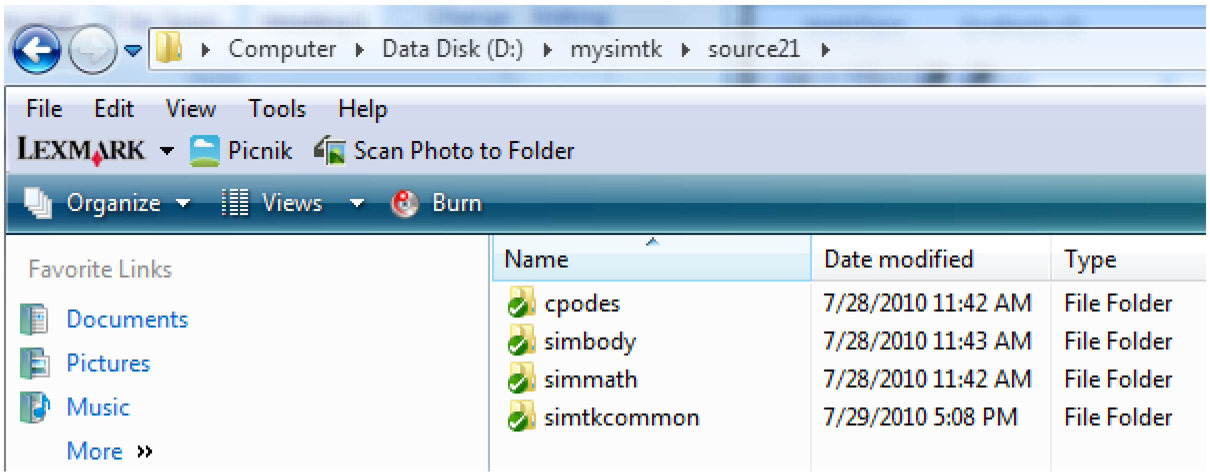

SimTKCoreRelease2.1 - Checkout - TortoiseSVN Finished!

Action	Path
Added	D:\mysimtk\source21\simtkcommon\Scalar\src\Scalar.cpp
Added	D:\mysimtk\source21\simtkcommon\Scalar\etc
Added	D:\mysimtk\source21\simtkcommon\etc
Added	D:\mysimtk\source21\simtkcommon\CMakeLists.txt
Added	D:\mysimtk\source21\simtkcommon\sharedTarget
Added	D:\mysimtk\source21\simtkcommon\sharedTarget\CMakeLists.txt
Updated	D:\mysimtk\source21\simtkcommon
Completed	At revision: 634

1156 kBytes transferred in 0 minute(s) and 4 second(s)

Added:239 Updated:1

5. Repeat this for the other three libraries: cpodes, simmath, and simbody. Be sure to replace "simtkcommon" in both fields of the checkout form with the correct name of the library being checked out. Now your mysimtk/source21 directory should look something like this:


Computer ▸ Data Disk (D:) ▸ mysimtk ▸ source21
File Edit View Tools Help
LEXMARK Picnik Scan Photo to Folder
Organize Views Burn
Favorite Links
Documents
Pictures
Music
More »

Name	Date modified	Type
cpodes	7/28/2010 11:42 AM	File Folder
simbody	7/28/2010 11:43 AM	File Folder
simmath	7/28/2010 11:42 AM	File Folder
simtkcommon	7/29/2010 5:08 PM	File Folder

[You can add molmodel to this list if you are going to use SimTK for molecular simulation; it is checked out in the same manner as the other libraries.]

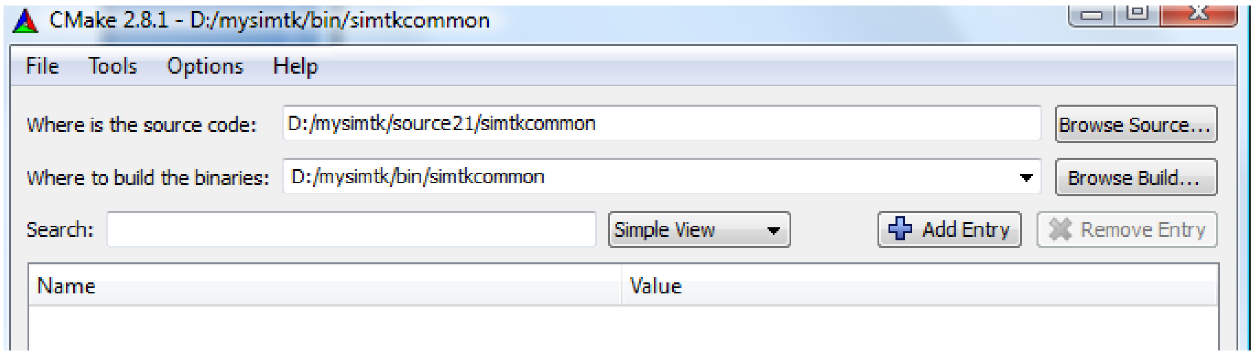
Step 4: run CMake

CMake is used to create a Visual Studio solution file that can be used to compile and build the code. Here we are going to get all the CMaking over with. The result will be four Visual Studio "solution" (.sln) files which we will then use to build the four projects.

1. Run the CMake GUI. You may have a shortcut on your Desktop, or you can run it from the Start button. Look in the upper-left corner and make sure you have CMake 2.8.1 or later.
2. Set the following fields:

Where is the source code: .../mysimtk/source21/simtkcommon
Where to build the binaries: .../mysimtk/bin/simtkcommon

Again, the "..." should contain the location at which you created mysimtk.


CMake 2.8.1 - D:/mysimtk/bin/simtkcommon

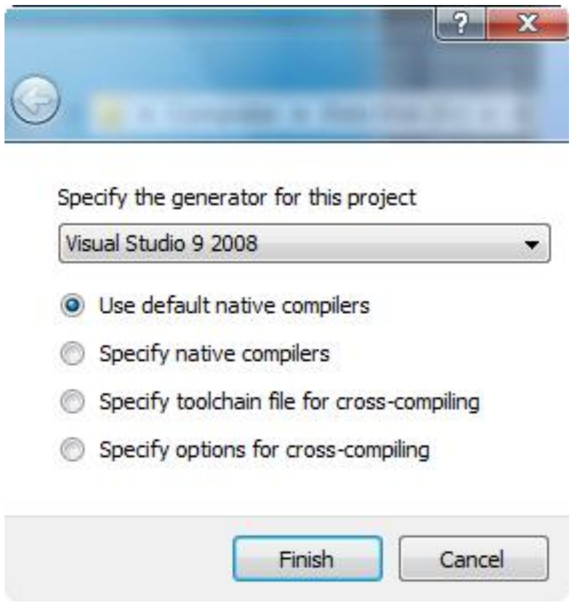
Where is the source code: D:/mysimtk/source21/simtkcommon

Where to build the binaries: D:/mysimtk/bin/simtkcommon

Search:

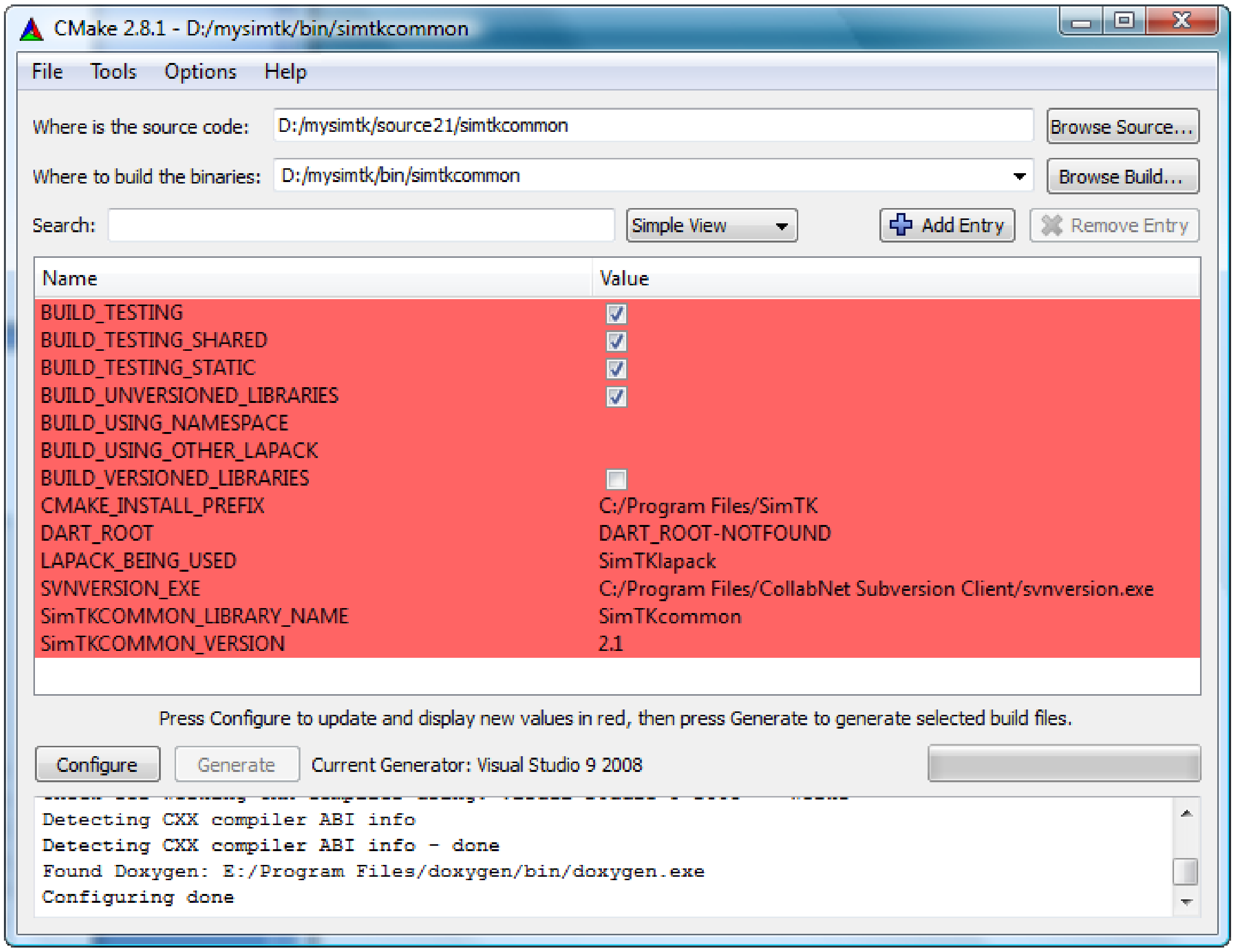
Simple View

3. Press the Configure button. Let CMake create the binary directory. Choose a "generator" that best matches the compiler you are using, for example "Visual Studio 9 2008." Select "Use default native compilers" and click Finish.


Specify the generator for this project: Visual Studio 9 2008

[✓ Use default native compilers]
Specify native compilers
Specify toolchain file for cross-compiling
Specify options for cross-compiling

You should now see something like this:



Where is the source code: D:/mysimtk/source21/simtkcommon

Where to build the binaries: D:/mysimtk/bin/simtkcommon

Search:

Name	Value
BUILD_TESTING	✓
BUILD_TESTING_SHARED	✓
BUILD_TESTING_STATIC	✓
BUILD_UNVERSIONED_LIBRARIES	✓
BUILD_USING_NAMESPACE	
BUILD_USING_OTHER_LAPACK	
BUILD_VERSIONED_LIBRARIES	☐
CMAKE_INSTALL_PREFIX	C:/Program Files/SimTK
DART_ROOT	DART_ROOT-NOTFOUND
LAPACK_BEING_USED	SimTKlapack
SVNVERSION_EXE	C:/Program Files/CollabNet Subversion Client/svnversion.exe
SimTKCOMMON_LIBRARY_NAME	SimTKcommon
SimTKCOMMON_VERSION	2.1

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

Current Generator: Visual Studio 9 2008

```
Detecting CXX compiler ABI info
Detecting CXX compiler ABI info - done
Found Doxygen: E:/Program Files/doxygen/bin/doxygen.exe
Configuring done
```

CMake highlights in red anything that has changed; since this is new, everything is red.

To use a different installation directory, change "Simple View" to "Advanced View" (on some versions, you just need to check the box for "Advanced"). Then, set the values for the fields SimTK_SDK and SimTK_INSTALL_PREFIX to the desired directory; don't use CMAKE_INSTALL_PREFIX.

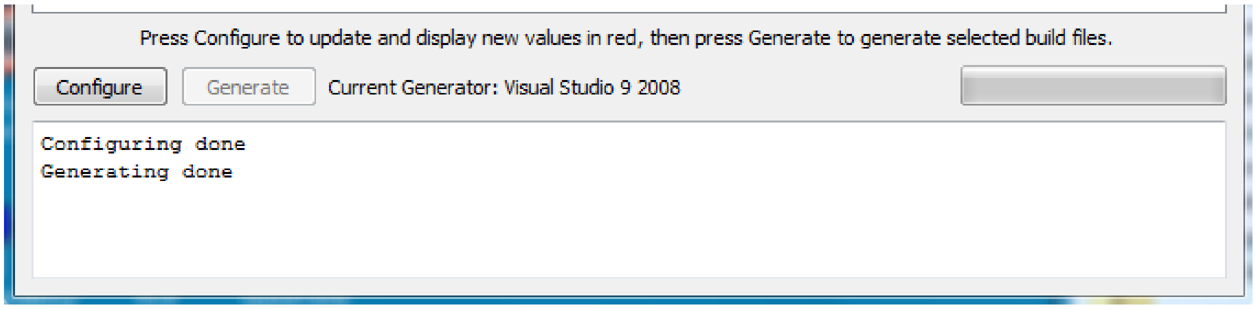
Note: by default, we are building all the regression test cases, and each test will be linked twice, once with the shared version of the simtkcommon library, and once with the static version. That takes longer to build so you can save some time by unchecking one or more of the BUILD_TESTING options. However, at least the first time you build these libraries it can be useful to have all the various test cases around, so we suggest you build everything at least this once.

Hit Configure again. This time there should be nothing in red and the Generate button will be available. Now hit the Generate button. The bottom panel should say "Generating done" and not have error messages. If you don't have a command-line Subversion client, you'll see a warning like:

```
Could not find 'svnversion' executable; 'about' will be
wrong. (Cygwin provides one on Windows.)
```

This will not prevent you from building but see Step 1 above for how to get a suitable Subversion client.

If everything works, you will see messages like that below:

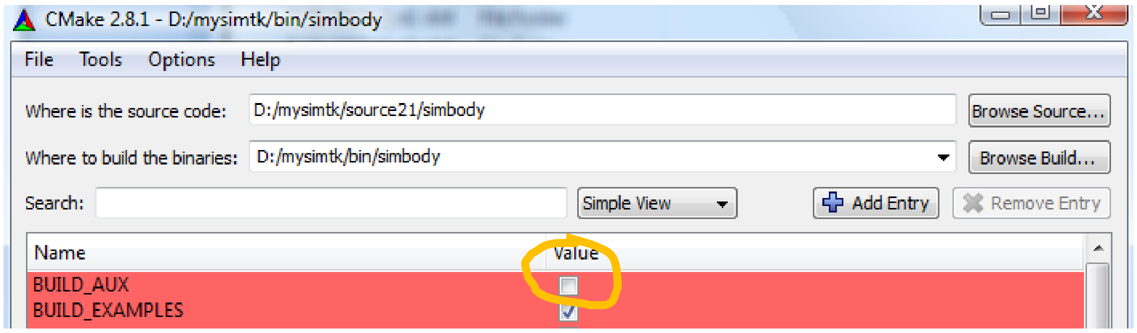

Press Configure to update and display new values in red, then press Generate to generate selected build files.
Configure
Generate
Current Generator: Visual Studio 9 2008

```
Configuring done
Generating done
```

That last step should have created the first Visual Studio solution file for you. It will be `mysimtk/bin/simtkcommon/SimTKcommon.sln`. You can look to see if it got created, but don't open that file yet. Let's create the rest of them first.

4. **Generate CPodes**.
    a. Without leaving the CMake GUI (or start it again if you exited), change the source and binary directories from simtkcommon to mysimtk/source21/cpodes and mysimtk/bin/cpodes, respectively. Be sure to change the name in both fields! The main panel will go blank again.
    b. Press Configure (everything is red). Create the binary directory and set the "generator" as for simtkcommon. If you used a different installation directory, remember to switch to Advanced View and change the values for SimTK_INSTALL_PREFIX and SimTK_SDK again.
    c. Press Configure again, and again if needed until nothing is red.
    d. Then press Generate and we're done creating `mysimtk/bin/cpodes/SimTKcpodes.sln`.
5. **Generate Simmath**.
    a. Again in the CMake GUI, change the source and binary directories from cpodes to mysimtk/source21/simmath and mysimtk/bin/simmath, respectively.
    b. Press Configure. Change the install directory if necessary as above.
    c. Press Configure again, and again if needed until nothing is red.
    d. Then press Generate to create `mysimtk/bin/simmath/SimTKmath.sln`.
6. **Generate Simbody**. Pay attention -- this time there will be a choice about whether to build the VTK visualizer, which results in a simbody_aux library in addition to the simbody library. That's not the default, but you should select it.
    a. Change the source and binary directories from simmath to mysimtk/source21/simbody and mysimtk/bin/simbody, respectively.

b. Press Configure. The first variable in the red section is BUILD_AUX and it has an empty checkbox to its right. Check this box.



Where is the source code: D:/mysimtk/source21/simbody

Where to build the binaries: D:/mysimtk/bin/simbody

Name	Value
BUILD_AUX	☐
BUILD_EXAMPLES	✓

c. Press Configure. Now you will see in red the version of VTK that CMake found. This should be the VTK binary that was installed in Step 2, the one in C:\Program Files\SimTK or your installation directory. If you have a different VTK installed that CMake found, you should change this manually to use the binaries from Step 2 (VTK 5.0) instead so you won't have to deal with version compatibility issues. [We have successfully used the Simbody visualizer with VTK 5.4 and 5.6 as well, but you do have to be very careful that the compiler and options match between VTK and SimTK.]

d. Press Configure again, and again if needed until nothing is red.

e. Then press Generate to create `mysimtk/bin/simbody/Simbody.sln`.

[Molmodel is configured similarly to simbody if you are building that.]

That's it for CMake. Exit the CMake GUI; we won't be needing it any more for the rest of the build.

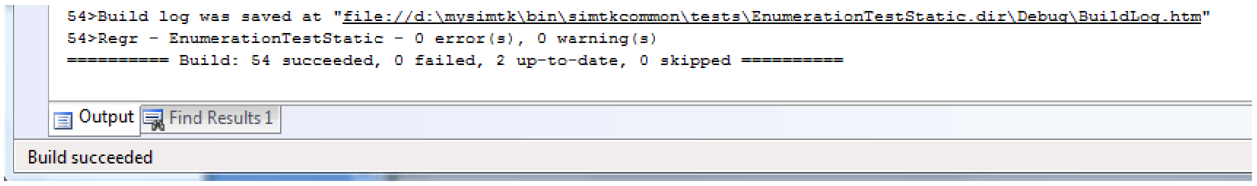
Step 5: build and install the SimTKcommon library

The libraries can be built using a number of different configurations: Debug, RelWithDebInfo (Release with Debug Information), and Release. The default "configuration" is Debug. This configuration is very useful for testing and debugging, but 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. This makes the libraries bigger, but not slower. If you want the smallest possible libraries, use the Release configuration. One set of Debug libraries and one set of Release (or RelWithDebInfo) libraries can coexist in the installation directory; all Debug libraries are suffixed with "_d". In all cases, both shared (DLL) and static libraries are built; static libraries have "_static" in their names. We'll go through the build process for the SimTKcommon library in some detail and then do the other three quickly. For all libraries, we will first build the libraries in Debug mode and then build it in RelWithDebInfo mode. Note that while the Debug version can be very helpful for you when you are debugging, it is *much* (more than 10X) slower than the RelWithDebInfo version, so you will definitely need to build the RelWithDebInfo version to do anything substantial.

Here is the procedure to build, test, and install the SimTKcommon library. The process is similar for the other libraries.

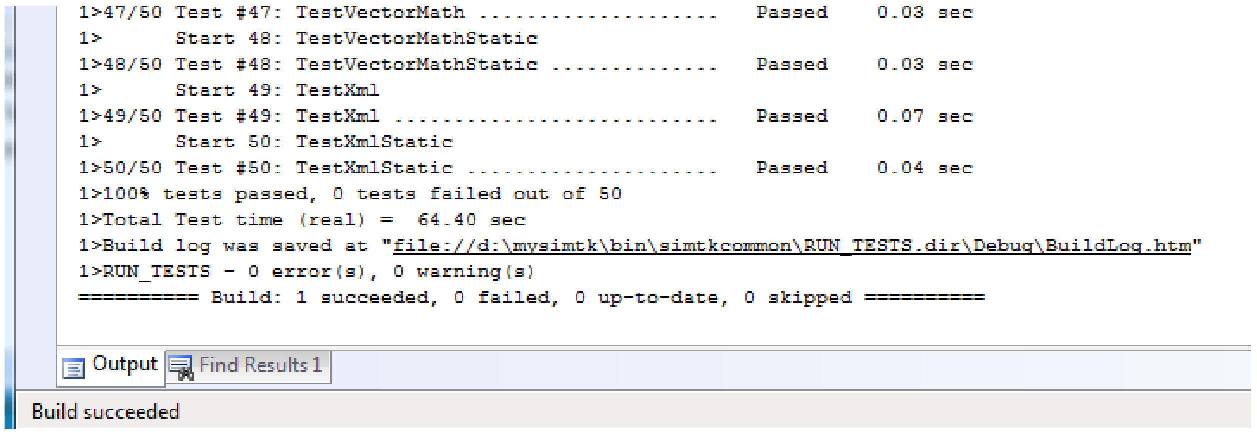
1. Use Windows Explorer to go to mysimtk/bin/simtkcommon. Double-click on the file SimTKcommon.sln there. This brings up Visual Studio. Set configuration to Debug.

2. Look for the ALL_BUILD target in the Solution Explorer panel. Right click on ALL_BUILD, and select "Build" from the drop down. This will build simtkcommon_d and simtkcommon_static_d libraries (remember, the naming convention appends "_d" for debug libraries and "_static" for static libraries), as well as a complete set of dynamic and static test cases. The end of the "Output" window should look like this:



```
54>Build log was saved at "file://d:\mysimtk\bin\simtkcommon\tests\EnumerationTestStatic.dir\Debug\BuildLog.htm"
54>Regr - EnumerationTestStatic - 0 error(s), 0 warning(s)
========== Build: 54 succeeded, 0 failed, 2 up-to-date, 0 skipped ==========
```

3. Find the RUN_TESTS target in the Solution Explorer panel. Right click on RUN_TESTS and select "Build". This will run all the tests (slowly since we're in Debug). Every test will run twice, once for the dynamically linked version and once for the statically linked version. The end of the "Output" window should look like this:



```
1>47/50 Test #47: TestVectorMath .....................   Passed    0.03 sec
1>      Start 48: TestVectorMathStatic
1>48/50 Test #48: TestVectorMathStatic ...............   Passed    0.03 sec
1>      Start 49: TestXml
1>49/50 Test #49: TestXml ............................   Passed    0.07 sec
1>      Start 50: TestXmlStatic
1>50/50 Test #50: TestXmlStatic ......................   Passed    0.04 sec
1>100% tests passed, 0 tests failed out of 50
1>Total Test time (real) =  64.40 sec
1>Build log was saved at "file://d:\mysimtk\bin\simtkcommon\RUN_TESTS.dir\Debug\BuildLog.htm"
1>RUN_TESTS - 0 error(s), 0 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

4. Build the Doxygen documentation. Find the DoxygenApiDocs target in the Solution Explorer panel. Right click on it and select "Build".

5. Find the INSTALL target. Right click on it to install the Debug libraries and doxygen docs.

6. Now change the configuration to "RelWithDebInfo". This is the production configuration we recommend.

7. Build the ALL_BUILD target. Check that the build was successful.

8. Build the RUN_TESTS target . Check that all tests passed. [Note: TestXML and TestXMLStatic may fail in RelWithDebInfo using Visual Studio Express; ignore that.]

9. Build the INSTALL target. This installs all the Release (RelWithDebInfo) libraries.

Step 6: build and install the CPodes, Simmath, and Simbody libraries

The procedure here is similar to that in Step 5. Depending on how much of a hurry you are in, you may want to skip building the Debug libraries, or at least don't build the RUN_TESTS target after a Debug build. You should at least do RUN_TESTS after the Release (or RelWithDebInfo) build, however, to make sure everything is installed and working properly.

1. **Cpodes**: Use Windows Explorer to go to mysimtk/bin/cpodes. Double-click on SimTKcpodes.sln to bring up Visual Studio.
   - Set configuration to Debug. Build the following: ALL_BUILD, RUN_TESTS (optional), DoxygenApiDocs, INSTALL.
   - Set configuration to RelWithDebInfo. Build the following: ALL_BUILD, RUN_TESTS, INSTALL.
   - Exit Visual Studio.

   Cpodes is small and builds quickly.

2. **Simmath**: Use Windows Explorer to go to mysimtk/bin/simmath. Double-click on SimTKmath.sln to bring up Visual Studio.
   - Set configuration to Debug. Build the following: ALL_BUILD, RUN_TESTS (optional), DoxygenApiDocs, INSTALL.
   - Set configuration to RelWithDebInfo. Build the following: ALL_BUILD, RUN_TESTS, INSTALL.
   - Exit Visual Studio.

   Simmath contains a large third-party application, constrained optimizer IpOpt, which takes a while to build.

3. **Simbody**: Use Windows Explorer to go to mysimtk/bin/simbody. Double-click on Simbody.sln to bring up Visual Studio.
   - Set configuration to Debug. Build the following: ALL_BUILD, RUN_TESTS (optional), DoxygenApiDocs, INSTALL.
   - Set configuration to RelWithDebInfo. Build the following: ALL_BUILD, RUN_TESTS, INSTALL.

   This will install both the simbody library and simbody_aux library that uses VTK.

That’s it – you’ve now built and installed SimTK and Simbody. [Note: we did not build the Molmodel library that is used for molecule modeling; you may see references to that in some of the examples. If you want to build molmodel also, treat it similarly to simbody.]

Step 7: Play with Simbody

While you are in Visual Studio with the Simbody solution file, you will see a bunch of “AdHoc” tests in the Solution Explorer window. These use the VTK visualizer to do some crude animation. At this point you should be able to run those and see the animation.

1. Experiment with AdHoc tests in the Simbody build environment.
   - Make sure you have the configuration set to “RelWithDebInfo,” not “Debug” – everything will run very slowly in Debug.
   - Right click on one of the targets whose name begins with “Adhoc –“. Select “Set as Startup Project” from the drop down list.
   - Hit Ctrl-F5 (short for Debug/Start Without Debugging) to start the program. For most of them you will get a console window and a VTK display window. Some require input at the console before they start the simulation.
   - Repeat for other tests.
2. From outside of the Simbody build environment, you can create projects that link with the installed libraries. A good place to start is to download the examples from the SimTKCore project downloads page on Simtk.org (http://simtk.org/home/simtkcore, click on “Downloads”). There are instructions there listing all the library names.
3. Try working through the first tutorial from the Documents page of the SimTKCore project (https://simtk.org/home/simtkcore).

Getting help

If you have any trouble along the way, you can post to the forum in the SimTKCore project (http://simtk.org/home/simtkcore, click on Advanced -> Public Forums), or contact Michael Sherman (Sherm) directly at msherman@stanford.edu.