ESyS-Particle-win Build Instructions
SoilVision Systems Ltd.
11 May 2013

Please carefully read the following instructions before start compiling ESyS-Particle-win on a windows system, our main test platform is Windows 7 however it should work on other recent windows platforms (XP, Vista etc.).

1 Prerequisites

In order to build ESyS-Particle on a windows platform, the following MANDATORY packages have to be installed prior to the ESyS-Particle-win building process, note that if any of these packages provides both 32-bit and 64-bit versions, only install *32-bit* version of that package, we have not yet tested any 64-bit version packages at this moment:

1.1 Microsoft Visual C++

We have tested Visual C++ 2005, 2008, 2010 Professional but Express version should also work.

1.2 Boost

One can always compile Boost from source; however it is easier to install the pre-compiled Boost binary package from the BoostPro download page: http://www.boostpro.com/download/

The Boost version we tested is BoostPro ***1.47.0***, you only need to install the version associated with your Visual Studio, e.g. only install VC9.0 version if you just have Visual C++ 2008.

1.3 Python

It is easier to install the pre-compiled Python package from the official python homepage: http://python.org/download/

The Python version we tested is Python ***2.7.2***.

1.4 OpenMPI

OpenMPI is required to build ESyS-Particle-win, download and install OpenMPI from http://www.open-mpi.org/software/ompi/v1.5/

You do not need to compile from the OpenMPI source.

1.5 CMake

Download and install the CMake package from:

http://www.cmake.org/cmake/resources/software.html

The CMake version we tested is CMake ***2.8.6***

2 Build ESyS-Particle-win

2.1 Download and extract package

Before trying to checkout esys-particle-win from Launchpad under windows, install bazaar for windows http://wiki.bazaar.canonical.com/Download, you will then need to follow the link https://help.launchpad.net/YourAccount/CreatingAnSSHKeyPair#Windows_.28PuTTY.29 in order to connect to Launchpad from a Windows PC. With pageant (PUTTY authentication agent, Note: you must use the latest pageant from the PUTTY homepage to avoid the access denied error reported from here) running, checkout the esys-particle-win branch from Launchpad repository lp:esys-particle/esys-particle-win to your local folder:

```
bzr branch lp:esys-particle/esys-particle-win
```

2.2 Check and modify CMake variables

Under the ROOT folder of local checked out source, open *CMakeLists.txt* (e.g. C:\esys-particle-win\CMakeLists.txt), check and modify the following CMake variables, note that you MUST change the backslash "\" to slash "/" while using CMake variables:

OMPI_INCLUDES: Change to the OpenMPI ***INCLUDE*** path on your local system, if your OpenMPI is installed by default, it should be something like:

C:/Program Files (x86)/OpenMPI_v1.5.3-win32/include

OMPI_LIB_PATH: Change to the OpenMPI ***LIB*** path on your local system, if your OpenMPI is installed by default, it should be something like:

C:/Program Files (x86)/OpenMPI_v1.5.3-win32/lib

PYTHON_INCLUDES: Change to the Python ***INCLUDE*** path on your local system, if your Python is installed by default, it should be something like:

C:/Python27/include

PYTHON_LIB_PATH: Change to the Python ***LIB*** path on your local system, if your Python is installed by default, it should be something like:

C:/Python27/libs

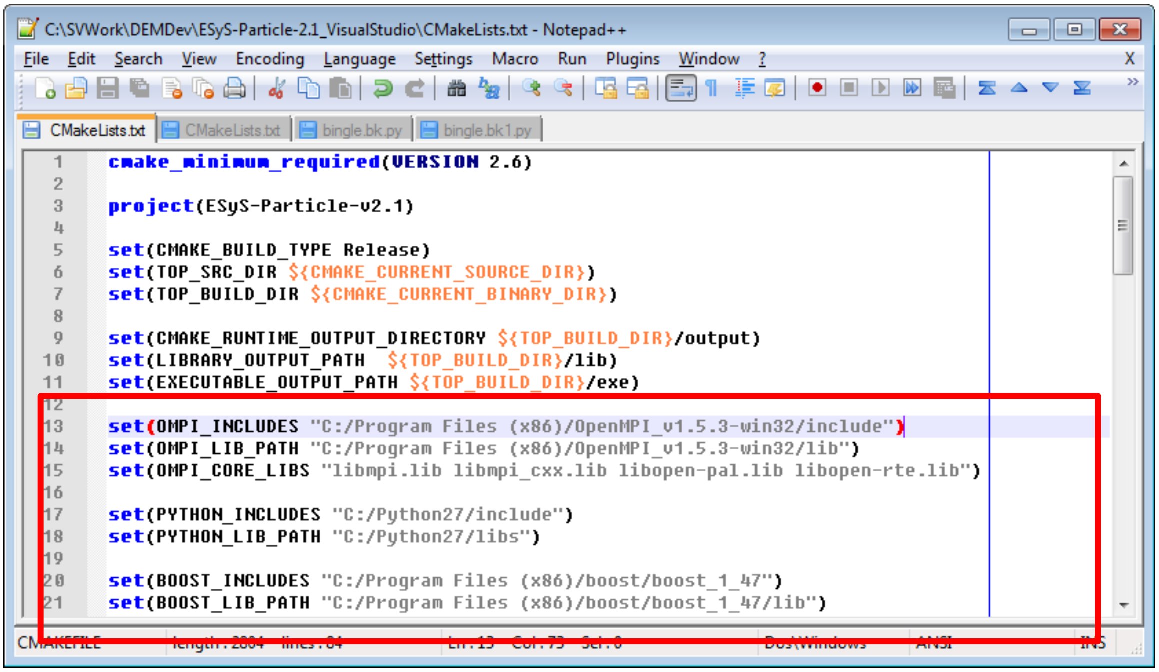
BOOST_INCLUDES: Change to the Boost ***INCLUDE*** path on your local system, if you installed BoostPro by default, it should be something like:

C:/Program Files (x86)/boost/boost_1_47

BOOST_LIB_PATH: Change to the Boost ***LIB*** path on your local system, if you installed BoostPro by default, it should be something like:

C:/Program Files (x86)/boost/boost_1_47/lib

An example of correctly modified CMakeLists.txt file should look like Figure 1.


C:\SVWork\DEMDev\ESyS-Particle-2.1_VisualStudio\CMakeLists.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
CMakeLists.txt
CMakeLists.txt
bingle.bk.py
bingle.bk1.py
CMAKEFILE length : 2004 lines : 24 Ln : 13 Col : 73 Sel : 0 Dos\Windows ANSI INS

```
cmake_minimum_required(VERSION 2.6)

project(ESyS-Particle-v2.1)

set(CMAKE_BUILD_TYPE Release)
set(TOP_SRC_DIR ${CMAKE_CURRENT_SOURCE_DIR})
set(TOP_BUILD_DIR ${CMAKE_CURRENT_BINARY_DIR})

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${TOP_BUILD_DIR}/output)
set(LIBRARY_OUTPUT_PATH  ${TOP_BUILD_DIR}/lib)
set(EXECUTABLE_OUTPUT_PATH ${TOP_BUILD_DIR}/exe)

set(OMPI_INCLUDES "C:/Program Files (x86)/OpenMPI_v1.5.3-win32/include")
set(OMPI_LIB_PATH "C:/Program Files (x86)/OpenMPI_v1.5.3-win32/lib")
set(OMPI_CORE_LIBS "libmpi.lib libmpi_cxx.lib libopen-pal.lib libopen-rte.lib")

set(PYTHON_INCLUDES "C:/Python27/include")
set(PYTHON_LIB_PATH "C:/Python27/libs")

set(BOOST_INCLUDES "C:/Program Files (x86)/boost/boost_1_47")
set(BOOST_LIB_PATH "C:/Program Files (x86)/boost/boost_1_47/lib")
```

Figure 1 CMakeLists.txt example file for ESyS-Particle-win (under ROOT folder of unpacked source)

2.3 Build ESyS-Particle-win

2.3.1 Generate CMake Makefiles

First open a Visual Studio command prompt (see Figure 2, the visual studio command prompt should be activated from: "Microsoft Visual Studio 2008->Visual Studio Tools->Visual Studio 2008 Command prompt", please do NOT set the "Visual Studio 2008 x64 Win64 Command Prompt".). Suppose you are using Visual Studio 2008 (VC 9.0), go to the root of your source folder and enter the “buildvs2008” subfolder and then type “mkvs9.bat”, if all your previous configurations are correct you should see output as shown in Figure 3.

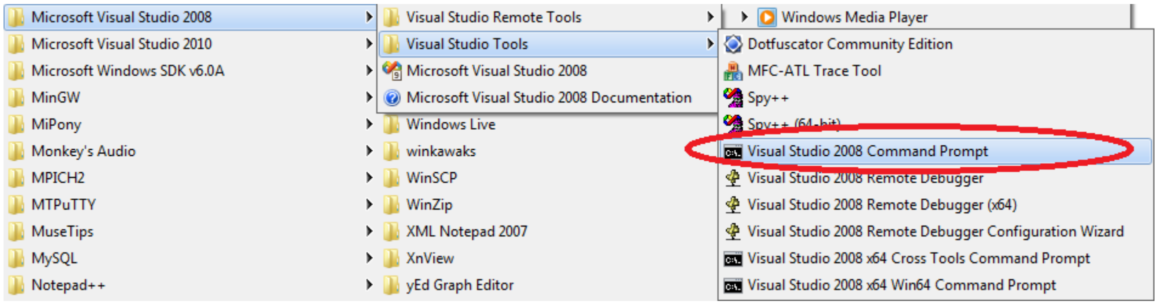
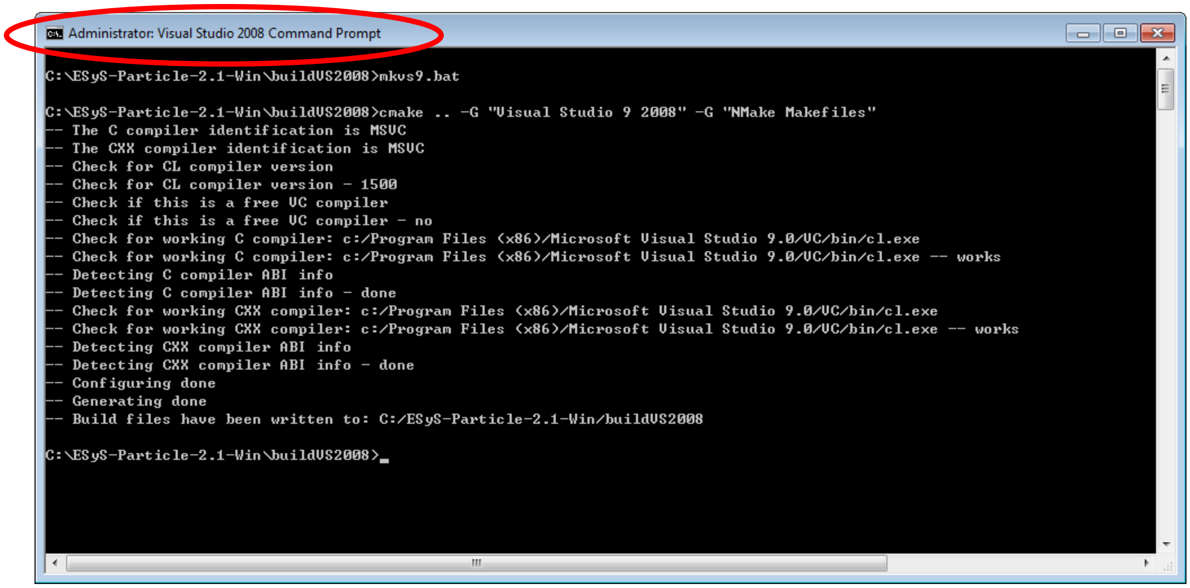

Microsoft Visual Studio 2008
Microsoft Visual Studio 2010
Microsoft Windows SDK v6.0A
MinGW
MiPony
Monkey's Audio
MPICH2
MTPuTTY
MuseTips
MySQL
Notepad++
Visual Studio Remote Tools
Visual Studio Tools
Microsoft Visual Studio 2008
Microsoft Visual Studio 2008 Documentation
Windows Live
winkawaks
WinSCP
WinZip
XML Notepad 2007
XnView
yEd Graph Editor
Windows Media Player
Dotfuscator Community Edition
MFC-ATL Trace Tool
Spy++
Spy++ (64-bit)
Visual Studio 2008 Command Prompt
Visual Studio 2008 Remote Debugger
Visual Studio 2008 Remote Debugger (x64)
Visual Studio 2008 Remote Debugger Configuration Wizard
Visual Studio 2008 x64 Cross Tools Command Prompt
Visual Studio 2008 x64 Win64 Command Prompt

Figure 2 Open Visual Studio Command Prompt

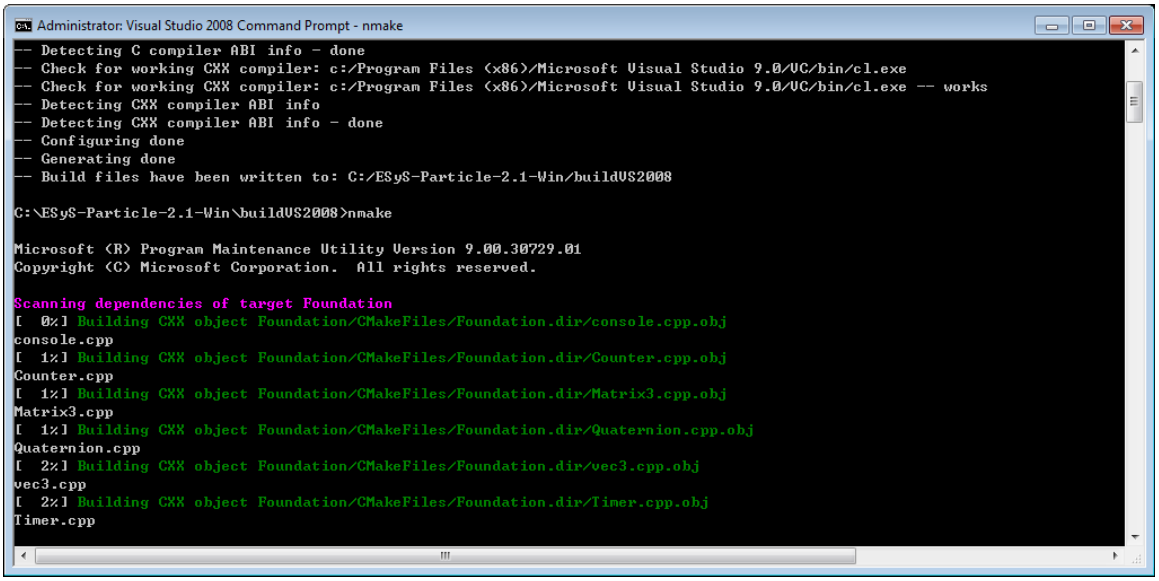


```
C:\ESyS-Particle-2.1-Win\buildVS2008>mkvs9.bat

C:\ESyS-Particle-2.1-Win\buildVS2008>cmake .. -G "Visual Studio 9 2008" -G "NMake Makefiles"
-- The C compiler identification is MSVC
-- The CXX compiler identification is MSVC
-- Check for CL compiler version
-- Check for CL compiler version - 1500
-- Check if this is a free VC compiler
-- Check if this is a free VC compiler - no
-- Check for working C compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
-- Check for working C compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
-- Check for working CXX compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/ESyS-Particle-2.1-Win/buildVS2008

C:\ESyS-Particle-2.1-Win\buildVS2008>_
```

Figure 3 Correct output from CMake generation

2.3.2 Build ESyS-Particle

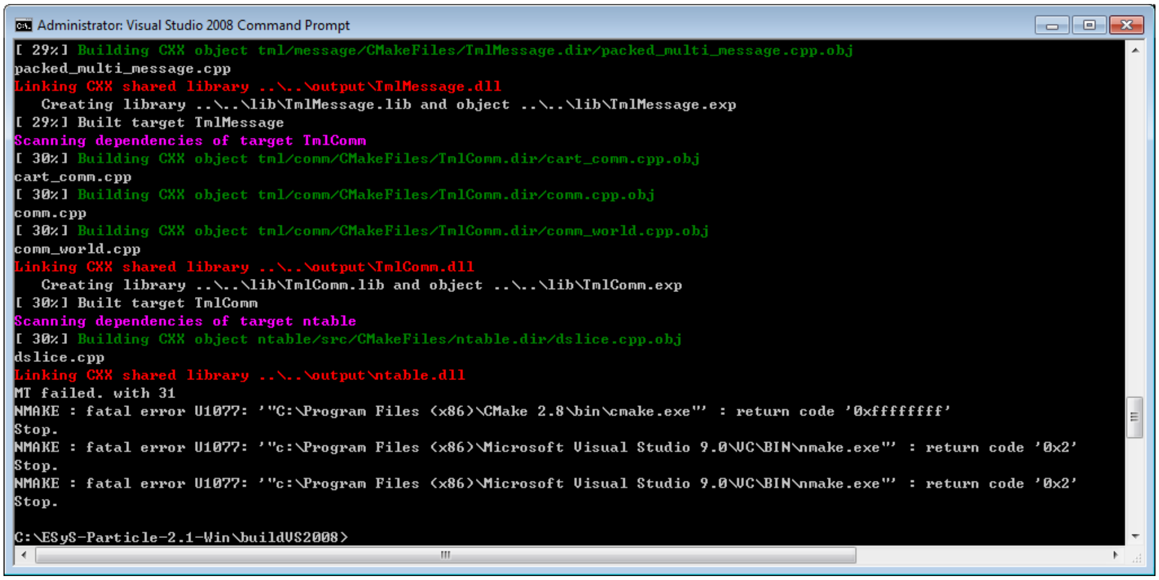
After the previous step, type “nmake” in the command window, the building process will then start. Be patient, it might take long time depending on the configuration of your hardware (Figure 4).



```
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
-- Check for working CXX compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/ESyS-Particle-2.1-Win/buildVS2008

C:\ESyS-Particle-2.1-Win\buildVS2008>nmake

Microsoft (R) Program Maintenance Utility Version 9.00.30729.01
Copyright (C) Microsoft Corporation.  All rights reserved.

Scanning dependencies of target Foundation
[  0%] Building CXX object Foundation/CMakeFiles/Foundation.dir/console.cpp.obj
console.cpp
[  1%] Building CXX object Foundation/CMakeFiles/Foundation.dir/Counter.cpp.obj
Counter.cpp
[  1%] Building CXX object Foundation/CMakeFiles/Foundation.dir/Matrix3.cpp.obj
Matrix3.cpp
[  1%] Building CXX object Foundation/CMakeFiles/Foundation.dir/Quaternion.cpp.obj
Quaternion.cpp
[  2%] Building CXX object Foundation/CMakeFiles/Foundation.dir/vec3.cpp.obj
vec3.cpp
[  2%] Building CXX object Foundation/CMakeFiles/Foundation.dir/Timer.cpp.obj
Timer.cpp
```

Figure 4 Building process of ESyS-Particle-win

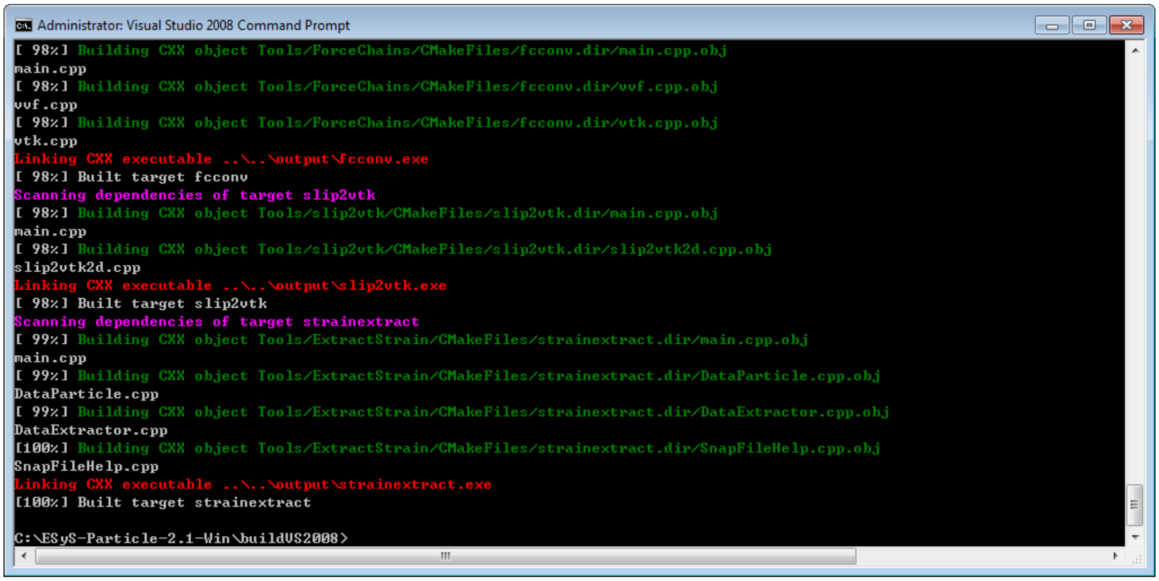
There could be error information like “MT failed. with 31” (see Figure 5) which might be possible bug of CMake, just type “nmake” again and the building process should be able to continue.



```
[ 29%] Building CXX object tml/message/CMakeFiles/TmlMessage.dir/packed_multi_message.cpp.obj
packed_multi_message.cpp
Linking CXX shared library ..\..\output\TmlMessage.dll
   Creating library ..\..\lib\TmlMessage.lib and object ..\..\lib\TmlMessage.exp
[ 29%] Built target TmlMessage
Scanning dependencies of target TmlComm
[ 30%] Building CXX object tml/comm/CMakeFiles/TmlComm.dir/cart_comm.cpp.obj
cart_comm.cpp
[ 30%] Building CXX object tml/comm/CMakeFiles/TmlComm.dir/comm.cpp.obj
comm.cpp
[ 30%] Building CXX object tml/comm/CMakeFiles/TmlComm.dir/comm_world.cpp.obj
comm_world.cpp
Linking CXX shared library ..\..\output\TmlComm.dll
   Creating library ..\..\lib\TmlComm.lib and object ..\..\lib\TmlComm.exp
[ 30%] Built target TmlComm
Scanning dependencies of target ntable
[ 30%] Building CXX object ntable/src/CMakeFiles/ntable.dir/dslice.cpp.obj
dslice.cpp
Linking CXX shared library ..\..\output\ntable.dll
MT failed. with 31
NMAKE : fatal error U1077: '"C:\Program Files (x86)\CMake 2.8\bin\cmake.exe"' : return code '0xffffffff'
Stop.
NMAKE : fatal error U1077: '"c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\BIN\nmake.exe"' : return code '0x2'
Stop.
NMAKE : fatal error U1077: '"c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\BIN\nmake.exe"' : return code '0x2'
Stop.

C:\ESyS-Particle-2.1-Win\buildVS2008>
```

Figure 5 Possible stop of CMake

You should see "[100%] Built target strainextract" at the end of the building process which indicates a successful build (Figure 6). The built binary files, including executables and DLLs, are placed under "buildVS200X/output" directory, where the "X" is your Visual Studio version.



```
[ 98%] Building CXX object Tools/ForceChains/CMakeFiles/fcconv.dir/main.cpp.obj
main.cpp
[ 98%] Building CXX object Tools/ForceChains/CMakeFiles/fcconv.dir/vvf.cpp.obj
vvf.cpp
[ 98%] Building CXX object Tools/ForceChains/CMakeFiles/fcconv.dir/vtk.cpp.obj
vtk.cpp
Linking CXX executable ..\..\output\fcconv.exe
[ 98%] Built target fcconv
Scanning dependencies of target slip2vtk
[ 98%] Building CXX object Tools/slip2vtk/CMakeFiles/slip2vtk.dir/main.cpp.obj
main.cpp
[ 98%] Building CXX object Tools/slip2vtk/CMakeFiles/slip2vtk.dir/slip2vtk2d.cpp.obj
slip2vtk2d.cpp
Linking CXX executable ..\..\output\slip2vtk.exe
[ 98%] Built target slip2vtk
Scanning dependencies of target strainextract
[ 99%] Building CXX object Tools/ExtractStrain/CMakeFiles/strainextract.dir/main.cpp.obj
main.cpp
[ 99%] Building CXX object Tools/ExtractStrain/CMakeFiles/strainextract.dir/DataParticle.cpp.obj
DataParticle.cpp
[ 99%] Building CXX object Tools/ExtractStrain/CMakeFiles/strainextract.dir/DataExtractor.cpp.obj
DataExtractor.cpp
[100%] Building CXX object Tools/ExtractStrain/CMakeFiles/strainextract.dir/SnapFileHelp.cpp.obj
SnapFileHelp.cpp
Linking CXX executable ..\..\output\strainextract.exe
[100%] Built target strainextract

C:\ESyS-Particle-2.1-Win\buildVS2008>
```

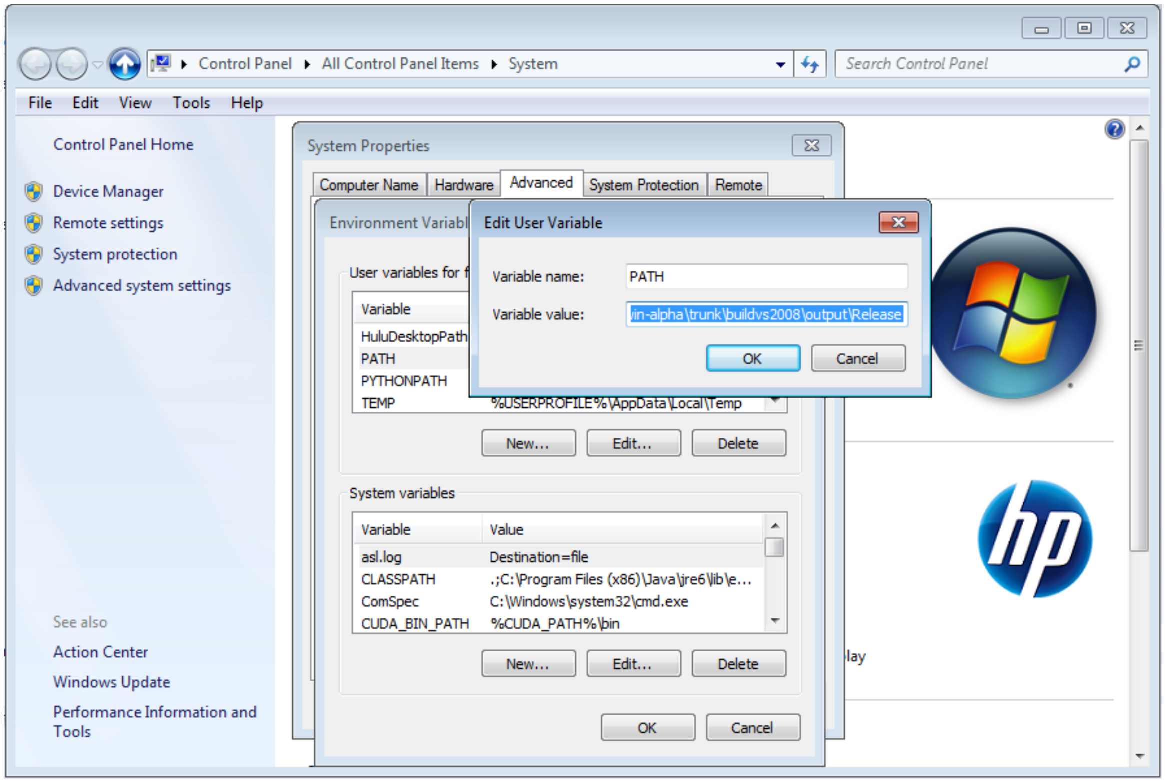
Figure 6 A successful build of ESyS-Particle-win

2.3.3 Add user environment variables

In order to use ESyS-Particle-win, the path of the compiled binaries need to be added to %PATH% and %PYTHONPATH% environment variables as shown in Figure 7 (on Windows 7 it is through: Control Panel\All Control Panel Items\System), note that you might need to log off and then log on for these variable changes to take effect. It is also essential to add the boost library path to both the %PATH% and %PYTHONPATH% variables. An example of changing the environment variables should look like:

```
Variable Name: PATH
Variable Value: %PATH%;C:\Program Files (x86)\boost\boost_1_47\lib;C:\esys-
particle-win\buildvs2008\output\Release
```

```
Variable Name: PYTHONPATH
Variable Value: %PYTHONPATH%;C:\Program Files
(x86)\boost\boost_1_47\lib;C:\esys-particle-win\buildvs2008\output\Release
```



Variable name: PATH

Variable value: in-alpha\trunk\buildvs2008\output\Release

User variables for f

Variable	
HuluDesktopPath	
PATH	
PYTHONPATH	
TEMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
asl.log	Destination=file
CLASSPATH	.;C:\Program Files (x86)\Java\jre6\lib\e...
ComSpec	C:\Windows\system32\cmd.exe
CUDA_BIN_PATH	%CUDA_PATH%\bin

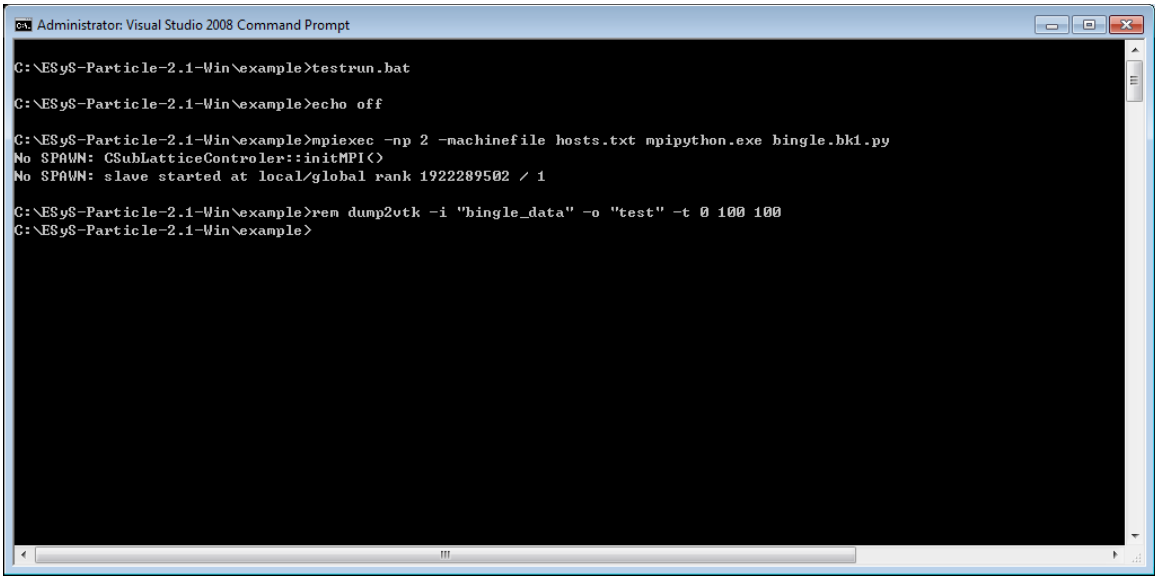
Figure 7 Edit %PATH% and %PYTHONPATH% Environment Variables

2.3.4 Run test Python script

From the command prompt, go to “examples” directory of the source folder, type “testrun.bat”, if you see the output like (see Figure 8):

```
C:\esys-particle-win\example>mpiexec -np 2 -machinefile hosts.txt
mpipython.exe bingle.py
No SPAWN: CSubLatticeControler::initMPI()
No SPAWN: slave started at local/global rank 1922289502 / 1
```

This means that the ESyS-Particle-win has been successfully installed.



```
C:\ESyS-Particle-2.1-Win\example>testrun.bat

C:\ESyS-Particle-2.1-Win\example>echo off

C:\ESyS-Particle-2.1-Win\example>mpiexec -np 2 -machinefile hosts.txt mpipython.exe bingle.bk1.py
No SPAWN: CSubLatticeControler::initMPI()
No SPAWN: slave started at local/global rank 1922289502 / 1

C:\ESyS-Particle-2.1-Win\example>rem dump2vtk -i "bingle_data" -o "test" -t 0 100 100
C:\ESyS-Particle-2.1-Win\example>
```

Figure 8 Test script of ESyS-Particle-win

3 Pre-built binaries

Pre-built binaries will also be provided for download, you can perform a DEM analysis using these binaries without going through the entire compilation process, however, Python is still required to run simulation scripts.

4 Known problems

4.1 Domain decomposition (Parallel)

Currently domain decomposition does not work, which means when initializing the neighbor search algorithm like:

sim = LsmMpi(numWorkerProcesses=nwp, mpiDimList=[nx,ny,nz])

You must specify nwp=nx=ny=nz=1 which means the simulation could run using only one domain.

4.2 Povray

Currently Povray does not work as expected, however the dump2pov is provided and interested users are welcome to modify and debug this module.

4.3 Release/Debug version

Due to the complication of the Python debug version which will add a “_d” to the compiled DLLs, the debug version of the compiled binaries does not work. Interested users can refer to http://docs.python.org/extending/windows.html to find a possible solution about how to get around this problem.

5 Build GenGeo-win

The particle generation module GenGeo is easier to compile and use than ESyS-Particle itself because it does not require OpenMPI, assuming you have the above ESyS-Particle prerequisites installed (see Section 1), follow the instructions below to download and install GenGeo-win:

5.1 Checkout gengeo-win from Launchpad repository

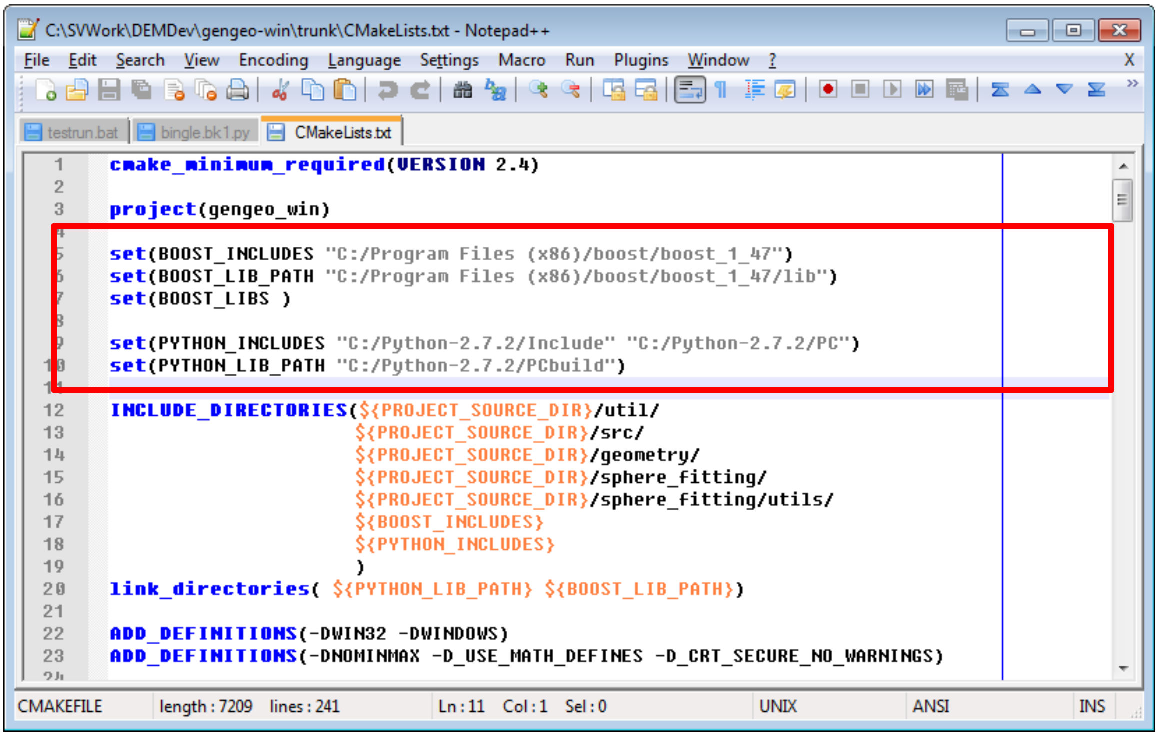
Similar to Section 2.1, checkout the gengeo-win branch from Launchpad repository lp:esys-particle/gengeo-win to your local folder:

```
bzr branch lp:esys-particle/gengeo-win
```

5.2 Check and modify CMake variables

Under the ROOT folder of checked GenGeo-win source, open CMakeLists.txt (i.e. C:\GenGeo-win\CMakeLists.txt), check and modify CMake variables similar to Section 2.2. Compared with ESyS-Particle-win compilation, you only need to modify the Boost and Python CMake variables.

An example of correctly modified CMakeLists.txt file should look like



C:\SVWork\DEMDev\gengeo-win\trunk\CMakeLists.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
testrun.bat
bingle.bk1.py
CMakeLists.txt
CMAKEFILE
length : 7209 lines : 241
Ln : 11 Col : 1 Sel : 0
UNIX
ANSI
INS

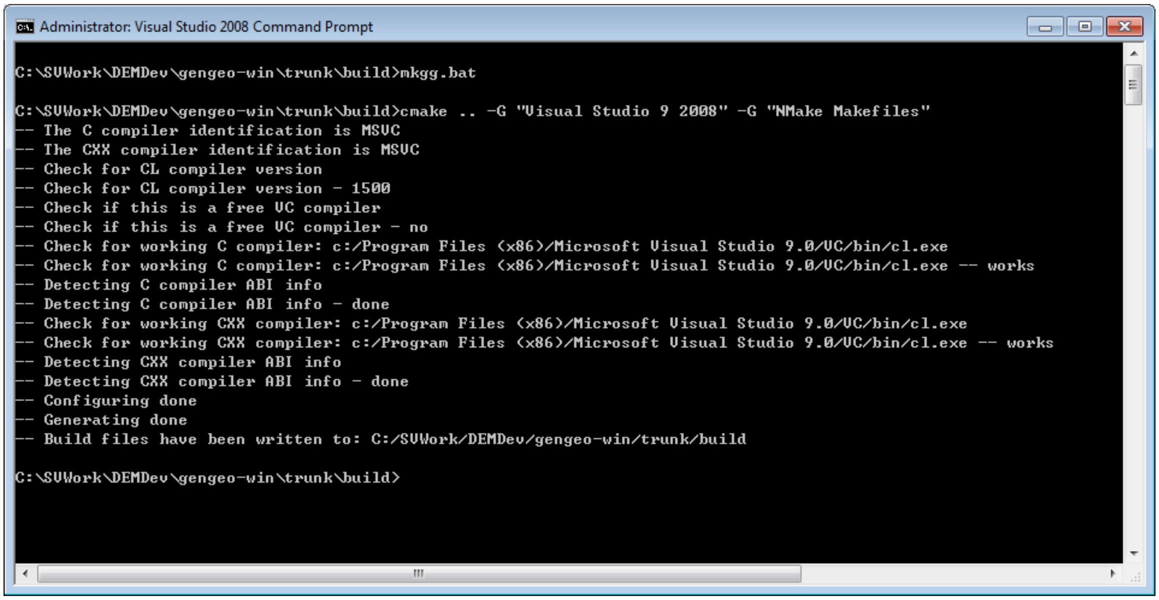
```
cmake_minimum_required(VERSION 2.4)

project(gengeo_win)

set(BOOST_INCLUDES "C:/Program Files (x86)/boost/boost_1_47")
set(BOOST_LIB_PATH "C:/Program Files (x86)/boost/boost_1_47/lib")
set(BOOST_LIBS )

set(PYTHON_INCLUDES "C:/Python-2.7.2/Include" "C:/Python-2.7.2/PC")
set(PYTHON_LIB_PATH "C:/Python-2.7.2/PCbuild")

INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/util/
                    ${PROJECT_SOURCE_DIR}/src/
                    ${PROJECT_SOURCE_DIR}/geometry/
                    ${PROJECT_SOURCE_DIR}/sphere_fitting/
                    ${PROJECT_SOURCE_DIR}/sphere_fitting/utils/
                    ${BOOST_INCLUDES}
                    ${PYTHON_INCLUDES}
                    )
link_directories( ${PYTHON_LIB_PATH} ${BOOST_LIB_PATH})

ADD_DEFINITIONS(-DWIN32 -DWINDOWS)
ADD_DEFINITIONS(-DNOMINMAX -D_USE_MATH_DEFINES -D_CRT_SECURE_NO_WARNINGS)
```

Figure 9 CMakeLists.txt example file for GenGeo-win

5.3 Build GenGeo-win

5.3.1 Generate CMake Makefiles

Open a Visual Studio command prompt, suppose you are using Visual Studio 2008 (VC 9.0), go to the root of your GenGeo-win source folder and enter the “build” subfolder and then type “mkgg.bat”, if all your previous configurations are correct you should see output as shown in Figure 10.

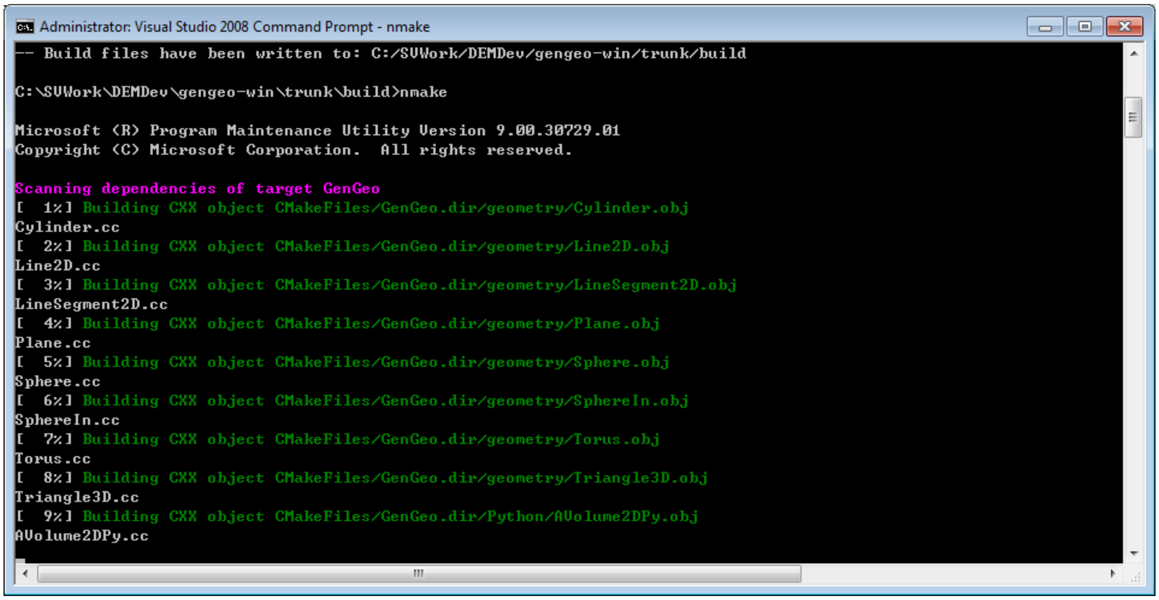


```
C:\SVWork\DEMDev\gengeo-win\trunk\build>mkgg.bat

C:\SVWork\DEMDev\gengeo-win\trunk\build>cmake .. -G "Visual Studio 9 2008" -G "NMake Makefiles"
-- The C compiler identification is MSVC
-- The CXX compiler identification is MSVC
-- Check for CL compiler version
-- Check for CL compiler version - 1500
-- Check if this is a free VC compiler
-- Check if this is a free VC compiler - no
-- Check for working C compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
-- Check for working C compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
-- Check for working CXX compiler: c:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/SVWork/DEMDev/gengeo-win/trunk/build

C:\SVWork\DEMDev\gengeo-win\trunk\build>
```

Figure 10 Correct output from GenGeo-win CMake generation

5.3.2 Build GenGeo-win

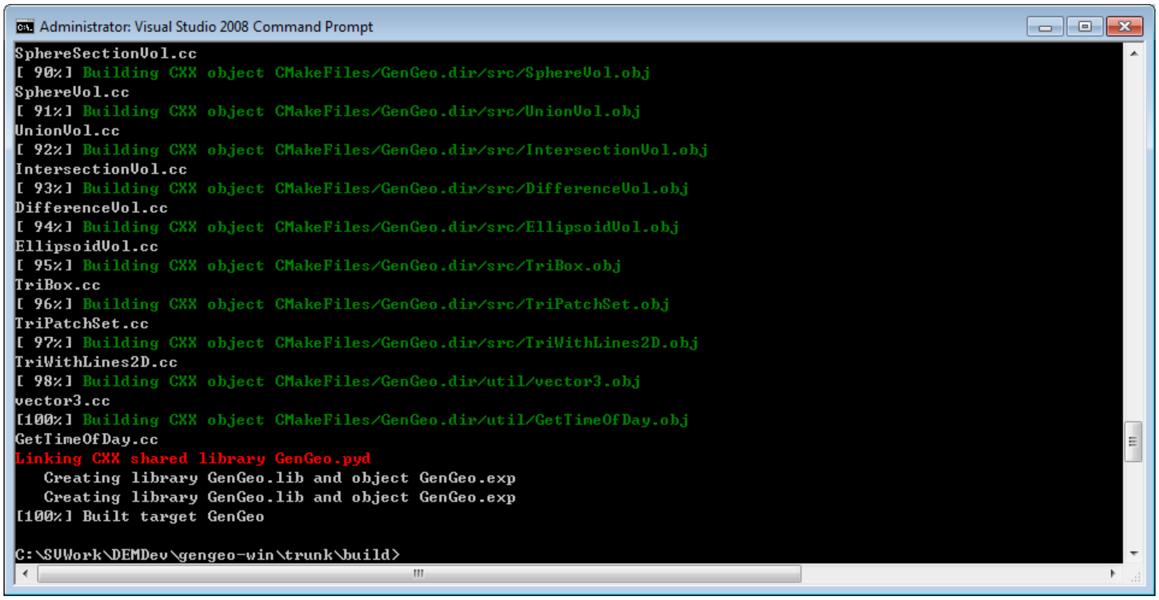
After the previous step, type “nmake” in the command window, the GenGeo-win building process will then start, similar to ESyS-Particle, it might take some time depending on the configuration of your hardware (Figure 11).



```
-- Build files have been written to: C:/SVWork/DEMDev/gengeo-win/trunk/build

C:\SVWork\DEMDev\gengeo-win\trunk\build>nmake

Microsoft (R) Program Maintenance Utility Version 9.00.30729.01
Copyright (C) Microsoft Corporation.  All rights reserved.

Scanning dependencies of target GenGeo
[  1%] Building CXX object CMakeFiles/GenGeo.dir/geometry/Cylinder.obj
Cylinder.cc
[  2%] Building CXX object CMakeFiles/GenGeo.dir/geometry/Line2D.obj
Line2D.cc
[  3%] Building CXX object CMakeFiles/GenGeo.dir/geometry/LineSegment2D.obj
LineSegment2D.cc
[  4%] Building CXX object CMakeFiles/GenGeo.dir/geometry/Plane.obj
Plane.cc
[  5%] Building CXX object CMakeFiles/GenGeo.dir/geometry/Sphere.obj
Sphere.cc
[  6%] Building CXX object CMakeFiles/GenGeo.dir/geometry/SphereIn.obj
SphereIn.cc
[  7%] Building CXX object CMakeFiles/GenGeo.dir/geometry/Torus.obj
Torus.cc
[  8%] Building CXX object CMakeFiles/GenGeo.dir/geometry/Triangle3D.obj
Triangle3D.cc
[  9%] Building CXX object CMakeFiles/GenGeo.dir/Python/AVolume2DPy.obj
AVolume2DPy.cc
```

Figure 11 Building process of GenGeo-win

You should see “[100%] Built target GenGeo” at the end of the building process which indicates a successful build (Figure 12). The built binary file, which is only the Python DLL “GenGeo.pyd”, is placed under “build” directory:



```
SphereSectionVol.cc
[ 90%] Building CXX object CMakeFiles/GenGeo.dir/src/SphereVol.obj
SphereVol.cc
[ 91%] Building CXX object CMakeFiles/GenGeo.dir/src/UnionVol.obj
UnionVol.cc
[ 92%] Building CXX object CMakeFiles/GenGeo.dir/src/IntersectionVol.obj
IntersectionVol.cc
[ 93%] Building CXX object CMakeFiles/GenGeo.dir/src/DifferenceVol.obj
DifferenceVol.cc
[ 94%] Building CXX object CMakeFiles/GenGeo.dir/src/EllipsoidVol.obj
EllipsoidVol.cc
[ 95%] Building CXX object CMakeFiles/GenGeo.dir/src/TriBox.obj
TriBox.cc
[ 96%] Building CXX object CMakeFiles/GenGeo.dir/src/TriPatchSet.obj
TriPatchSet.cc
[ 97%] Building CXX object CMakeFiles/GenGeo.dir/src/TriWithLines2D.obj
TriWithLines2D.cc
[ 98%] Building CXX object CMakeFiles/GenGeo.dir/util/vector3.obj
vector3.cc
[100%] Building CXX object CMakeFiles/GenGeo.dir/util/GetTimeOfDay.obj
GetTimeOfDay.cc
Linking CXX shared library GenGeo.pyd
   Creating library GenGeo.lib and object GenGeo.exp
   Creating library GenGeo.lib and object GenGeo.exp
[100%] Built target GenGeo

C:\SVWork\DEMDev\gengeo-win\trunk\build>
```

Figure 12 A successful build of GenGeo-win

5.3.3 Run test Python script

Similar to Section 2.3.3, you need to add the gengeo-win compiled binary output path (your_gengeo_source_directory\build) to the %PATH% and %PYTHONPATH% environment variables. An example of changing the gengeo-win environment variable should then look like (assuming you have already added esys-particle-win):

```
Variable Name: PATH
Variable Value: %PATH%;C:\esys-particle-win\buildvs2008\output\Release;
C:\gengeo-win\build
```

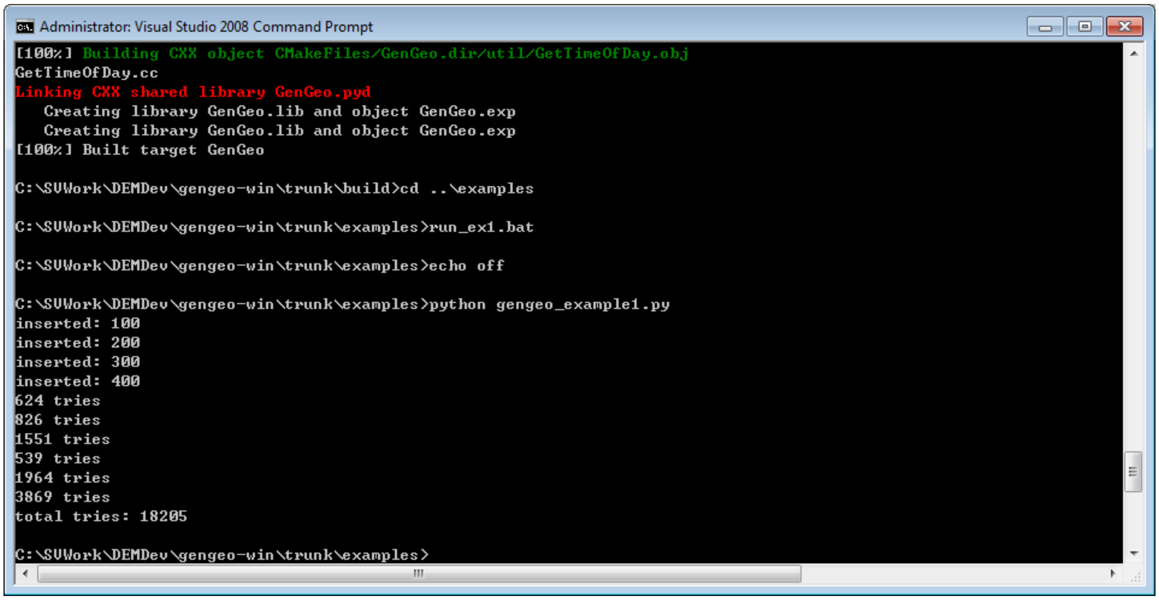
```
Variable Name: PYTHONPATH
Variable  Value:  %PYTHONPATH%;C:\esys-particle-win\buildvs2008\output\Release;
C:\gengeo-win\build
```

Type “run_ex1.bat” from command prompt, if you see the output like below:

```
inserted: 100
inserted: 200
inserted: 300
inserted: 400
624 tries
826 tries
1551 tries
539 tries
1964 tries
```

```
3869 tries
total tries: 18205
```

This indicates the gengeo-win has been successfully compiled and installed (Figure 13).



```
[100%] Building CXX object CMakeFiles/GenGeo.dir/util/GetTimeOfDay.obj
GetTimeOfDay.cc
Linking CXX shared library GenGeo.pyd
   Creating library GenGeo.lib and object GenGeo.exp
   Creating library GenGeo.lib and object GenGeo.exp
[100%] Built target GenGeo

C:\SVWork\DEMDev\gengeo-win\trunk\build>cd ..\examples

C:\SVWork\DEMDev\gengeo-win\trunk\examples>run_ex1.bat

C:\SVWork\DEMDev\gengeo-win\trunk\examples>echo off

C:\SVWork\DEMDev\gengeo-win\trunk\examples>python gengeo_example1.py
inserted: 100
inserted: 200
inserted: 300
inserted: 400
624 tries
826 tries
1551 tries
539 tries
1964 tries
3869 tries
total tries: 18205

C:\SVWork\DEMDev\gengeo-win\trunk\examples>
```

Figure 13 Test script of GenGeo-win

6 Point of contact

Problems related to esys-particle-win and gengeo-win can be contacted through Feng Chen (fchen3@gmail.com) or via Launchpad ESyS-Particle discussion board.

7 Disclaimer

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.