OpenVSP compile guidelines for Windows 10 and Python 3.5

Trevor Laughlin
Laughlin Research
September 29th, 2016

SUMMARY

This document is intended to provide useful guidelines for building OpenVSP along with the Python API. It is mostly based on the instructions from the OpenVSP GitHub site and a response from Rob McDonald on the Google Group. They are referenced below.

OpenVSP 3.5.2 CMake Build Errors on WIndows7

OpenVSP GitHub

This is most likely not a solution for everyone, but it may provide enough hints to help others on their specific platform. The information and instructions below are relative to my own system so please make adjustments where necessary.

OS INFORMATION

- Microsoft Windows 10 Pro 64-bit
- Version 10.0.14393 Build 14393

PREREQUISITES

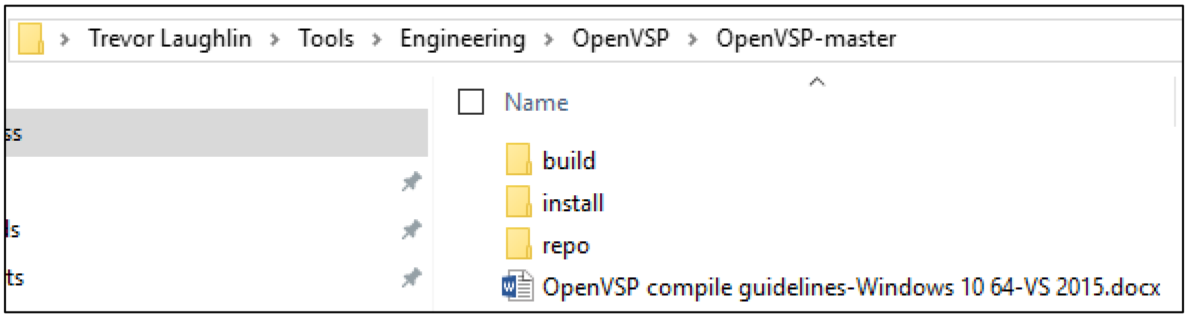
- Using CMake 3.6.2
- Using Visual Studio Community 2015
    - Version 14.0.25425.01 Update 3
- Using SWIG 3.0.10
    - Downloaded from http://swig.org/download.html
    - Used Windows version with pre-built executable
- Using Anaconda Python 3.5 64-bit as root environment
    - Download here https://www.continuum.io/downloads
    - If you would prefer Python 2.7 to be your root environment you can install that instead
    - Do not install both, use environments after installing your root Python
    - The root environment will be the one that gets put in your system Path variable and where it looks when you type "python" from a command prompt
    - Using Anaconda, you can create a Python 2.7 environment inside your Python 3.5 root environment and switch between them easily (or the other way around if you want)
- Not using any Doxygen options
- Using git version 2.9.2.windows.1

DIRECTORY STRUCTURE

Three main directories were used:

- repo (for the source code)
- build
- install

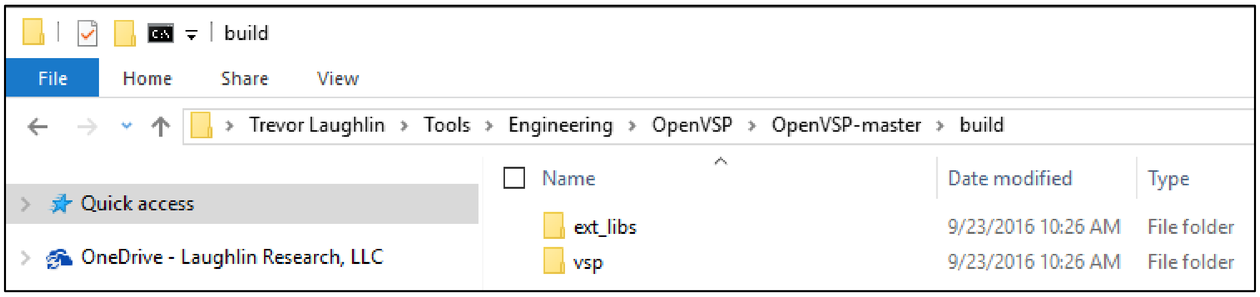
These three folders are in a root directory referred to as BASEDIR and may look something like this:


Trevor Laughlin › Tools › Engineering › OpenVSP › OpenVSP-master

Name
build
install
repo
OpenVSP compile guidelines-Windows 10 64-VS 2015.docx

where BASEDIR = "C:\Users\Trevor\Tools\Engineering\OpenVSP\OpenVSP-master".

Inside the "build" folder two more folders were created to hold the libraries and OpenVSP build files:


build
File
Home
Share
View
Trevor Laughlin › Tools › Engineering › OpenVSP › OpenVSP-master › build
Quick access
OneDrive - Laughlin Research, LLC

Name	Date modified	Type
ext_libs	9/23/2016 10:26 AM	File folder
vsp	9/23/2016 10:26 AM	File folder

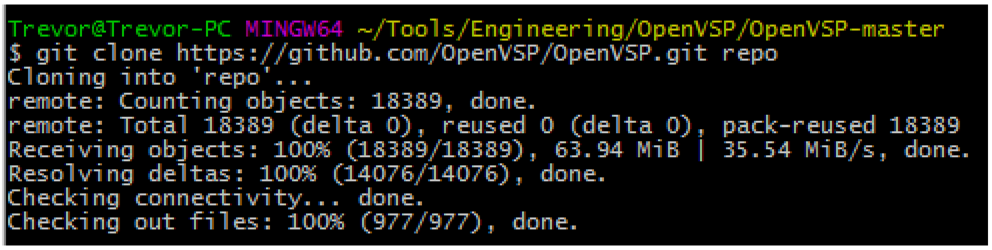
"ext_libs" are for the various libraries bundled with OpenVSP and "vsp" is for OpenVSP itself.

Important: Do not use spaces in your directories. One of the libraries fails to build in my experience (LIBXML2).

GETTING OPENVSP SOURCE

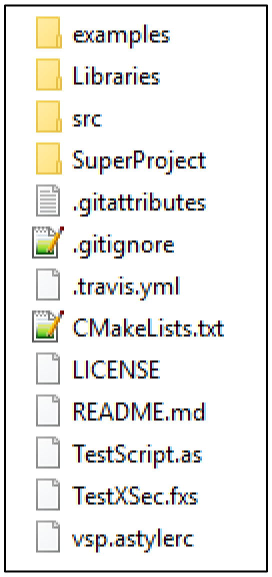
Use a Git clone command to get the source code into the repo folder:

```
$ git clone https://github.com/OpenVSP/OpenVSP.git repo
```



```
Trevor@Trevor-PC MINGW64 ~/Tools/Engineering/OpenVSP/OpenVSP-master
$ git clone https://github.com/OpenVSP/OpenVSP.git repo
Cloning into 'repo'...
remote: Counting objects: 18389, done.
remote: Total 18389 (delta 0), reused 0 (delta 0), pack-reused 18389
Receiving objects: 100% (18389/18389), 63.94 MiB | 35.54 MiB/s, done.
Resolving deltas: 100% (14076/14076), done.
Checking connectivity... done.
Checking out files: 100% (977/977), done.
```

The repo folder now contains the OpenVSP source:

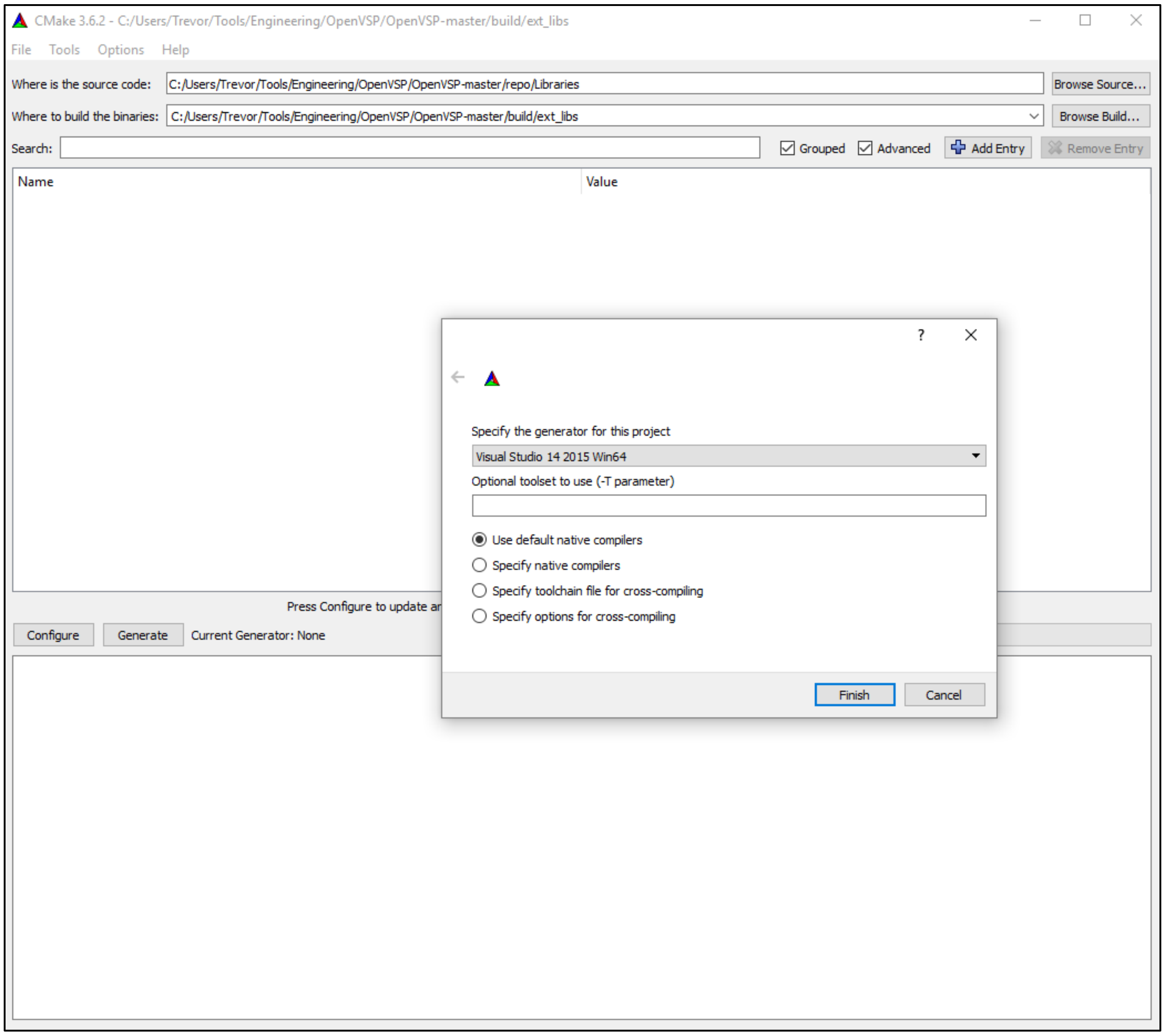


examples
Libraries
src
SuperProject
.gitattributes
.gitignore
.travis.yml
CMakeLists.txt
LICENSE
README.md
TestScript.as
TestXSec.fxs
vsp.astylerc

BUILDING BUNDLED LIBRARIES

The bundled libraries were first built using CMake and Visual Studio. The CMake GUI was used to set project variables. In CMake, the source directory was set to BASEDIR\repo\Libraries and the build directory set to BASEDIR\build\ext_libs. At this point the Configure button can be pushed and a generator selected. For my system, I selected “Visual Studio 14 2015 Win64”.

Note: For some reason the Analysis tools in OpenVSP were causing fatal errors when targeting a 64-bit machine. I was able to target 32-bit and they worked fine. This is probably a local issue, but if anyone else experiences this or knows a solution please share. The rest of the program seems to function normally.



Where is the source code: C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/repo/Libraries

Where to build the binaries: C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/build/ext_libs

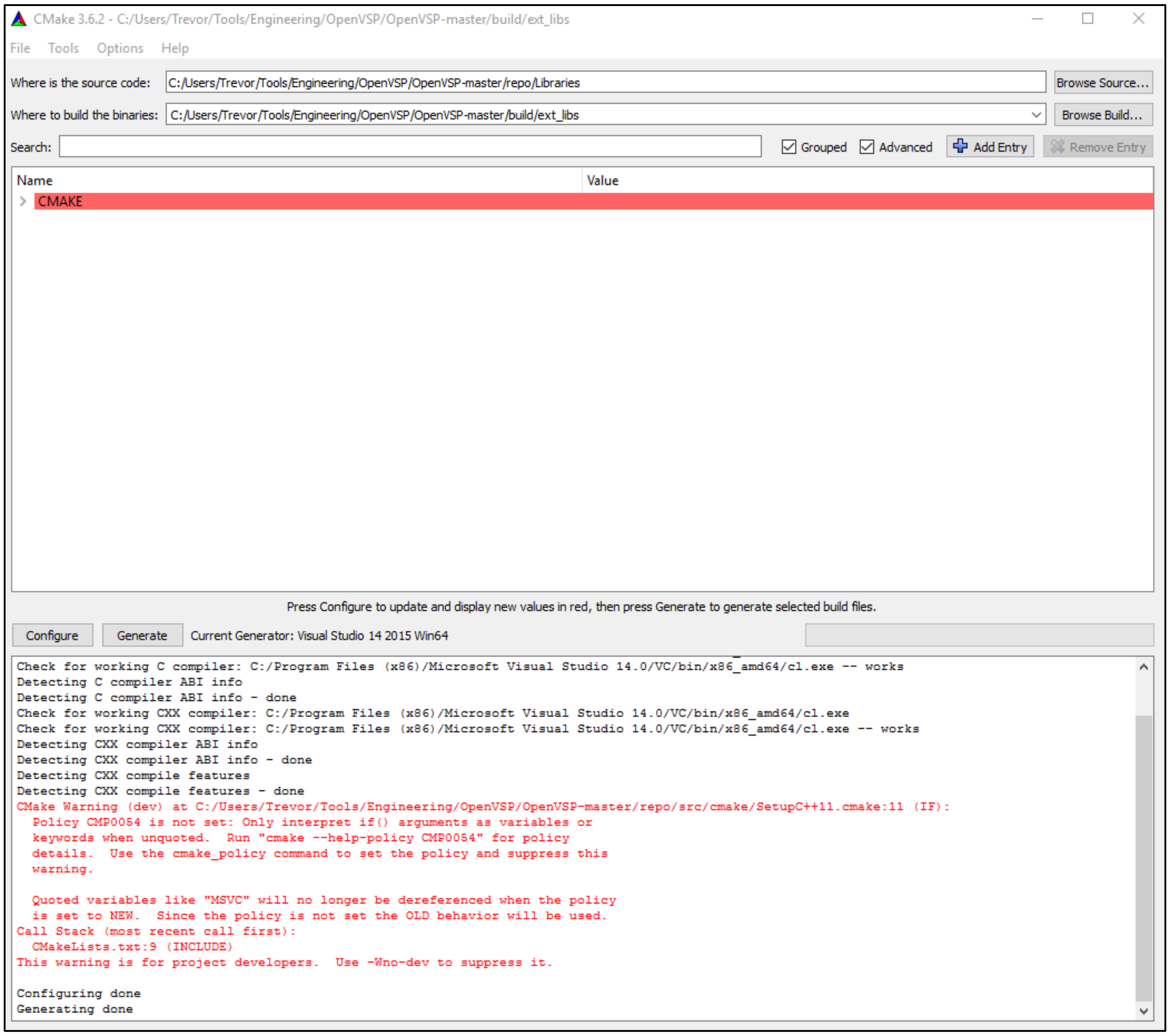
Search:

Specify the generator for this project: Visual Studio 14 2015 Win64

Optional toolset to use (-T parameter):

(●) Use default native compilers
( ) Specify native compilers
( ) Specify toolchain file for cross-compiling
( ) Specify options for cross-compiling

Some initial warnings may appear but “Configuring done” at the bottom appears. The Generate button generates the files to build in Visual Studio.



Where is the source code: C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/repo/Libraries

Where to build the binaries: C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/build/ext_libs

Search:

Name	Value
CMAKE	

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

Current Generator: Visual Studio 14 2015 Win64

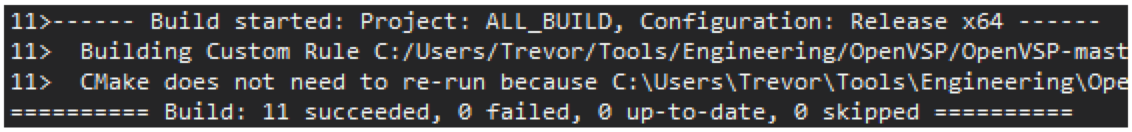
```
Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio 14.0/VC/bin/x86_amd64/cl.exe -- works
Detecting C compiler ABI info
Detecting C compiler ABI info - done
Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio 14.0/VC/bin/x86_amd64/cl.exe
Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio 14.0/VC/bin/x86_amd64/cl.exe -- works
Detecting CXX compiler ABI info
Detecting CXX compiler ABI info - done
Detecting CXX compile features
Detecting CXX compile features - done
CMake Warning (dev) at C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/repo/src/cmake/SetupC++11.cmake:11 (IF):
  Policy CMP0054 is not set: Only interpret if() arguments as variables or
  keywords when unquoted.  Run "cmake --help-policy CMP0054" for policy
  details.  Use the cmake_policy command to set the policy and suppress this
  warning.

  Quoted variables like "MSVC" will no longer be dereferenced when the policy
  is set to NEW.  Since the policy is not set the OLD behavior will be used.
Call Stack (most recent call first):
  CMakeLists.txt:9 (INCLUDE)
This warning is for project developers.  Use -Wno-dev to suppress it.

Configuring done
Generating done
```

Launch the file "VSP_LIBRARIES.sln" in Visual Studio, **set to "Release" mode**, and build the "ALL_BUILD" solution.

Hopefully you see this message at the end of the process:



```
11>------ Build started: Project: ALL_BUILD, Configuration: Release x64 ------
11>  Building Custom Rule C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-mast
11>  CMake does not need to re-run because C:\Users\Trevor\Tools\Engineering\Ope
========== Build: 11 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

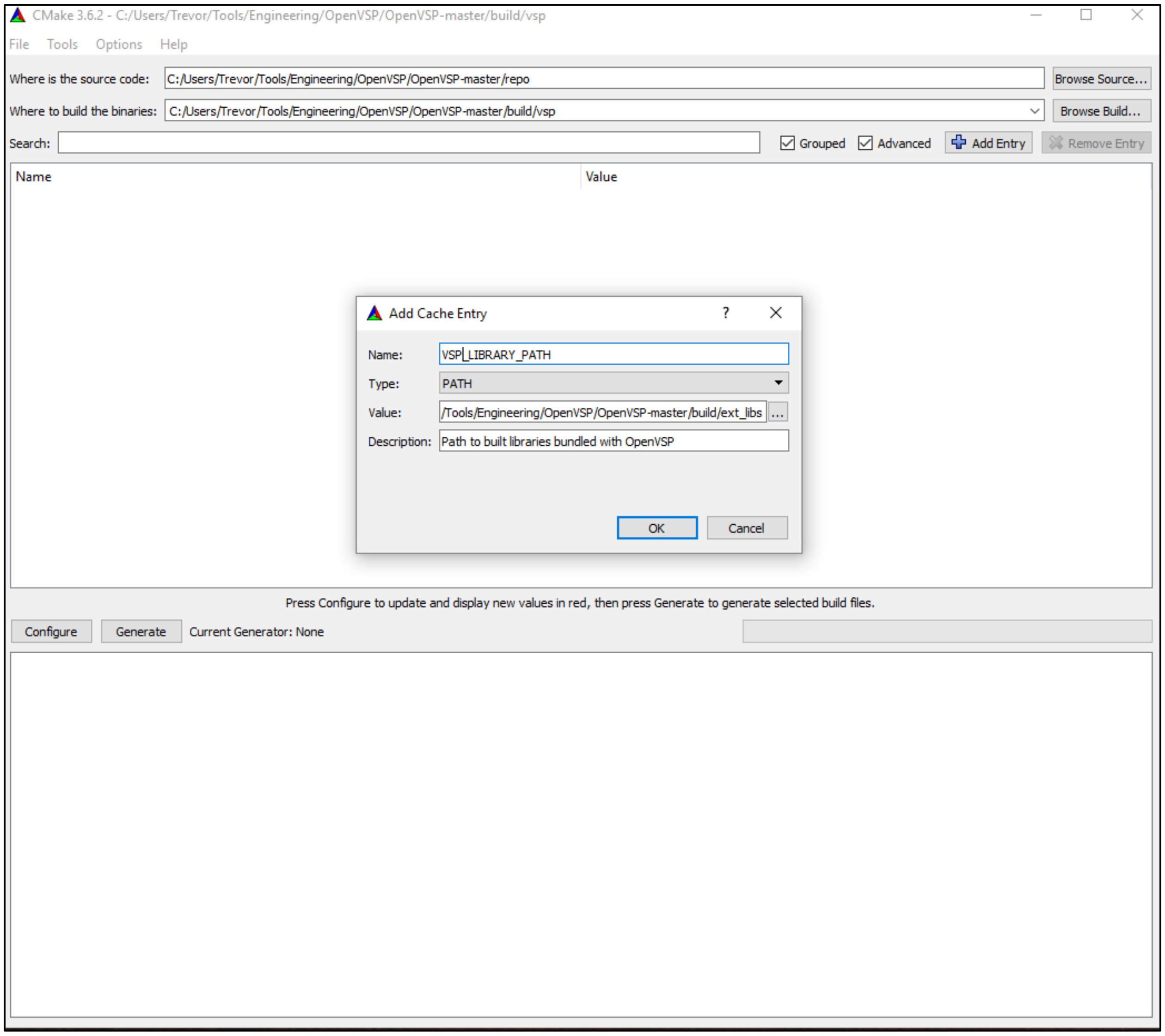
BUILDING OPENVSP

A similar process is used for building OpenVSP except some of the CMake variables may need adjusted and/or created. The source directory was set to BASEDIR\repo and the build directory to BASEDIR\build\vsp. Before hitting Configure, a few CMake variables were added.

- VSP_LIBRARY_PATH:PATH= BASEDIR\build\ext_libs
- CMAKE_BUILD_TYPE:STRING= Release

- CMAKE_INSTALL_PREFIX:PATH= BASEDIR\install

An example of adding these variables in the CMake GUI:



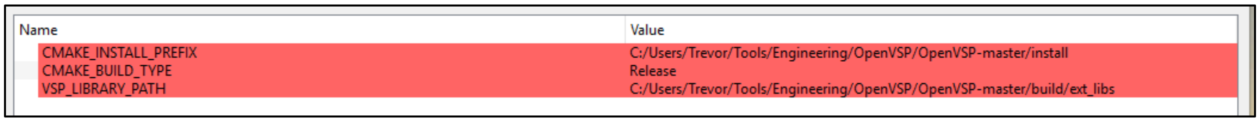
Where is the source code: C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/repo
Where to build the binaries: C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/build/vsp
Search:

Name: VSP_LIBRARY_PATH
Type: PATH
Value: /Tools/Engineering/OpenVSP/OpenVSP-master/build/ext_libs
Description: Path to built libraries bundled with OpenVSP

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

Current Generator: None

The variables in the CMake GUI before pressing Configure:



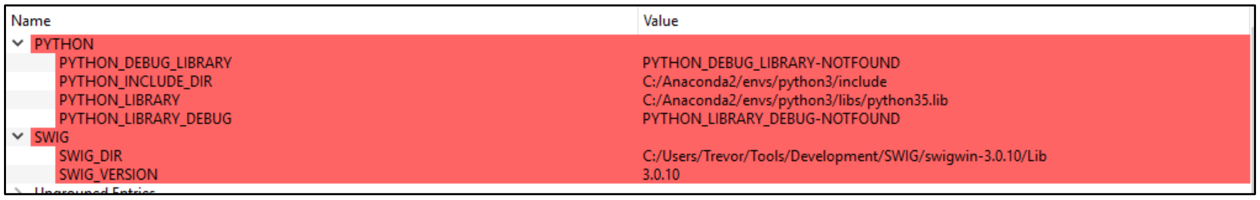
Name	Value
CMAKE_INSTALL_PREFIX	C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/install
CMAKE_BUILD_TYPE	Release
VSP_LIBRARY_PATH	C:/Users/Trevor/Tools/Engineering/OpenVSP/OpenVSP-master/build/ext_libs

Clicking Configure should generate many more variables with hopefully almost all of them being set (i.e., they found the paths to the bundled libraries). There are some changes that were made:

- FLTK_DIR
  - Did not find initially, so manually pointed to the directory that contains "FLTKConfig.cmake"
  - BASEDIR\build\ext_libs\FLTK-prefix\CMake

- SWIG_EXECUTABLE (**for the Python API**)
    - Pointed to local SWIG executable which for me was at "C:/Users/Trevor/Tools/Development/SWIG/swigwin-3.0.10/swig.exe"

After these adjustments the Configure button was pressed again and now new variables are shown related to SWIG and Python. The SWIG variables SWIG_DIR and SWIG_VERSION were set automatically. The Python variables were also found automatically, but verify they point to the correct Python libs if you have more than one installation.

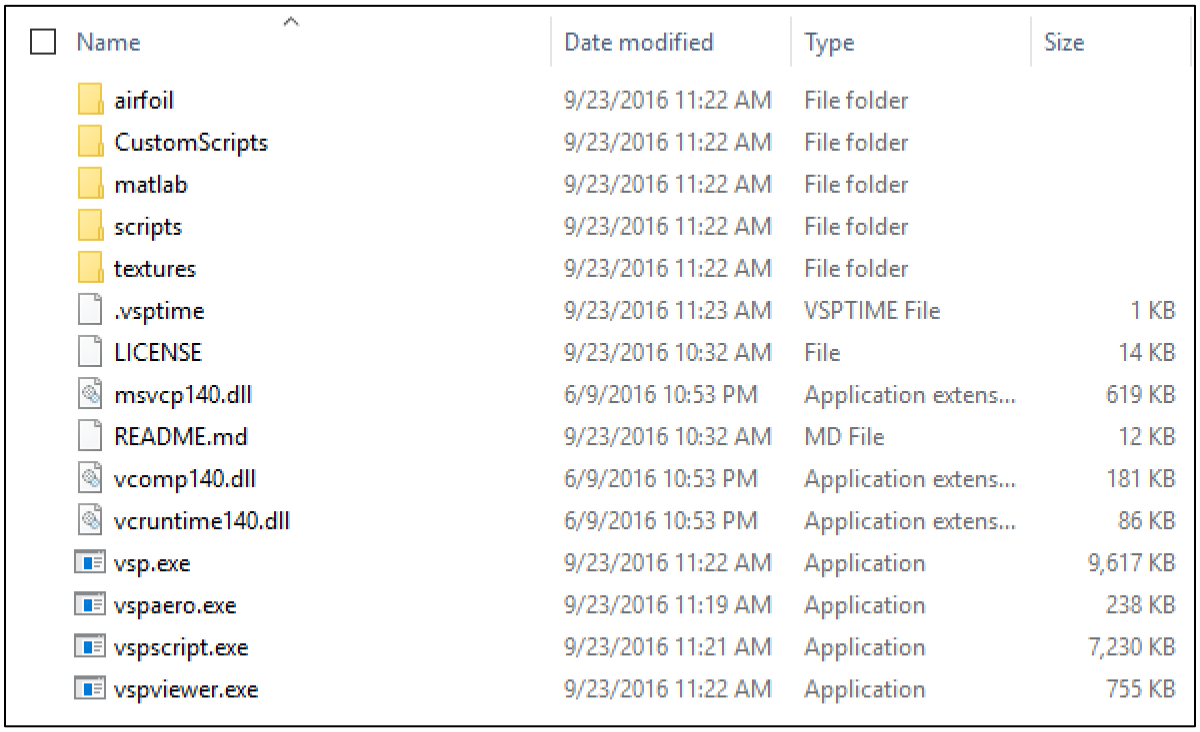


Name	Value
PYTHON	
PYTHON_DEBUG_LIBRARY	PYTHON_DEBUG_LIBRARY-NOTFOUND
PYTHON_INCLUDE_DIR	C:/Anaconda2/envs/python3/include
PYTHON_LIBRARY	C:/Anaconda2/envs/python3/libs/python35.lib
PYTHON_LIBRARY_DEBUG	PYTHON_LIBRARY_DEBUG-NOTFOUND
SWIG	
SWIG_DIR	C:/Users/Trevor/Tools/Development/SWIG/swigwin-3.0.10/Lib
SWIG_VERSION	3.0.10

Hit Configure one more time and hopefully nothing is in red in the CMake GUI. Then hit Generate.

Launch the file "VSP_TOP.sln" in Visual Studio, **set to "Release" mode**, and build the "INSTALL" solution.

Hopefully the build succeeded and your BASEDIR\install directory now looks like this:



Name	Date modified	Type	Size
airfoil	9/23/2016 11:22 AM	File folder	
CustomScripts	9/23/2016 11:22 AM	File folder	
matlab	9/23/2016 11:22 AM	File folder	
scripts	9/23/2016 11:22 AM	File folder	
textures	9/23/2016 11:22 AM	File folder	
.vsptime	9/23/2016 11:23 AM	VSPTIME File	1 KB
LICENSE	9/23/2016 10:32 AM	File	14 KB
msvcp140.dll	6/9/2016 10:53 PM	Application extens...	619 KB
README.md	9/23/2016 10:32 AM	MD File	12 KB
vcomp140.dll	6/9/2016 10:53 PM	Application extens...	181 KB
vcruntime140.dll	6/9/2016 10:53 PM	Application extens...	86 KB
vsp.exe	9/23/2016 11:22 AM	Application	9,617 KB
vspaero.exe	9/23/2016 11:19 AM	Application	238 KB
vspscript.exe	9/23/2016 11:21 AM	Application	7,230 KB
vspviewer.exe	9/23/2016 11:22 AM	Application	755 KB

INSTALLING PYTHON API

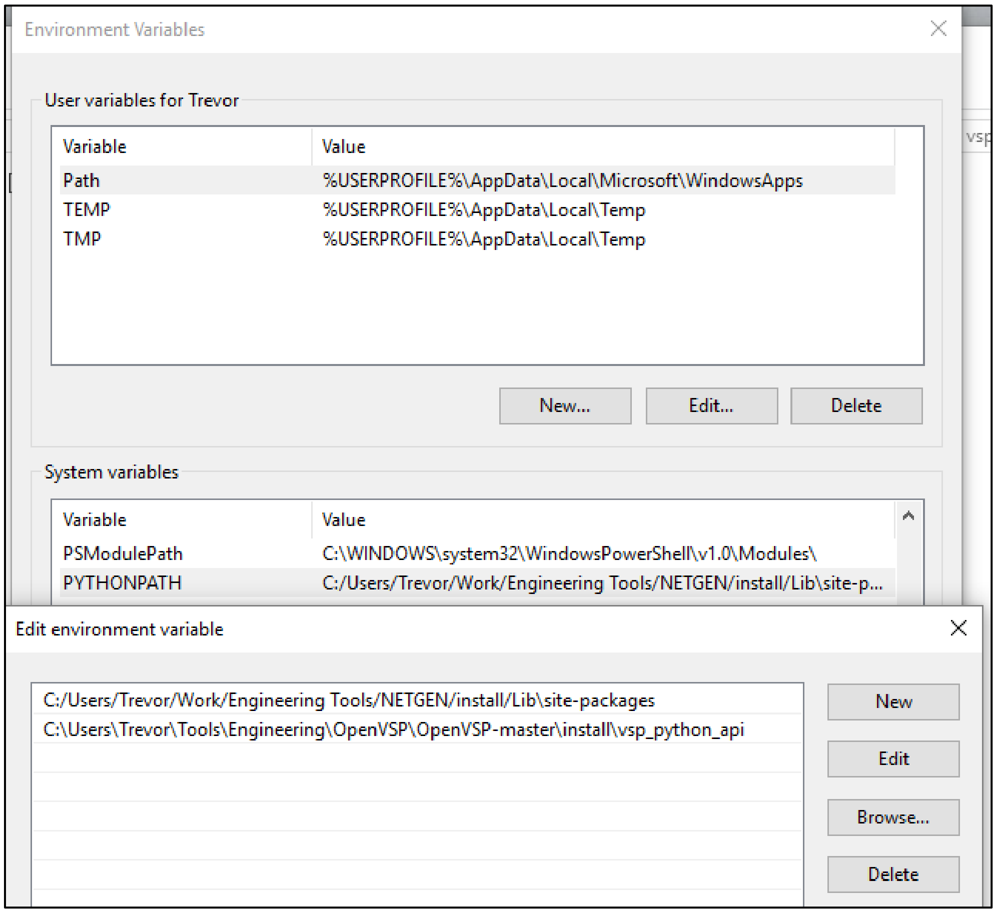
This section describes a few more manual steps to get the Python installed.

Look for the files

- vsp.py
    - BASEDIR\build\vsp\src\python_api

- _vsp.pyd
    - BASEDIR\build\vsp\src\python_api\Release

Copy these files in to a folder similar to BASEDIR\install\vsp_python_api. Create a new system environment variable called PYTHONPATH if it's not there already and add this directory to it:



Environment Variables

User variables for Trevor

Variable	Value
Path	%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

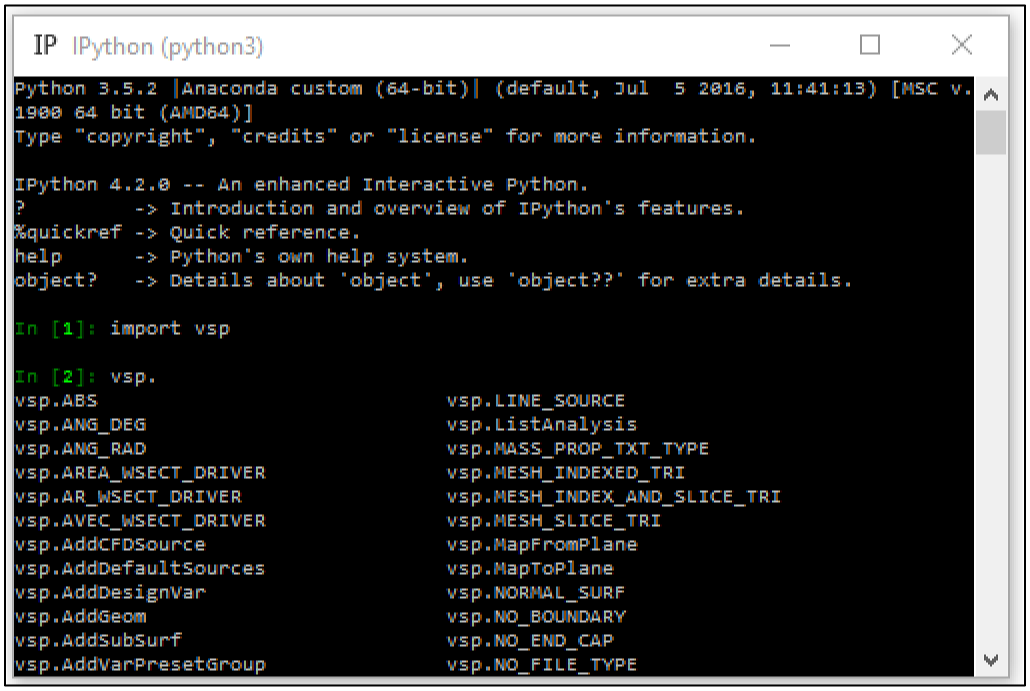
Variable	Value
PSModulePath	C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules\
PYTHONPATH	C:/Users/Trevor/Work/Engineering Tools/NETGEN/install/Lib\site-p...

Edit environment variable

```
C:/Users/Trevor/Work/Engineering Tools/NETGEN/install/Lib\site-packages
C:\Users\Trevor\Tools\Engineering\OpenVSP\OpenVSP-master\install\vsp_python_api
```

In this path create a filed called "__init__.py" and put "import vsp" in the first line. Note the double underscores before and after "init".

If this has been added to your PYTHONPATH successfully, you should be able to launch a Python console and do "import vsp" without any errors.



```
Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import vsp

In [2]: vsp.
vsp.ABS                                 vsp.LINE_SOURCE
vsp.ANG_DEG                             vsp.ListAnalysis
vsp.ANG_RAD                             vsp.MASS_PROP_TXT_TYPE
vsp.AREA_WSECT_DRIVER                   vsp.MESH_INDEXED_TRI
vsp.AR_WSECT_DRIVER                     vsp.MESH_INDEX_AND_SLICE_TRI
vsp.AVEC_WSECT_DRIVER                   vsp.MESH_SLICE_TRI
vsp.AddCFDSource                        vsp.MapFromPlane
vsp.AddDefaultSources                   vsp.MapToPlane
vsp.AddDesignVar                        vsp.NORMAL_SURF
vsp.AddGeom                             vsp.NO_BOUNDARY
vsp.AddSubSurf                          vsp.NO_END_CAP
vsp.AddVarPresetGroup                   vsp.NO_FILE_TYPE
```

In the repo\src\python_api folder there are some test files to try.

Note: The variables vsp.SLICE_PLANAR and vsp.SLICE_AWAVE don’t seem to be available in the test python script. Comment those lines out to run the complete script.