



University of  
**Nottingham**

UK | CHINA | MALAYSIA

A large, high-resolution image of the Earth as seen from space, showing the Western Hemisphere. The Earth is a vibrant blue and green sphere, with the Americas clearly visible. It is set against a deep black background filled with numerous small, distant stars. A thin white rectangular border is superimposed over the Earth, framing the title text.

# How To Build TexGen



# Contents

1. Introduction
2. Source Code
3. Prerequisites
4. Using Cmake
5. Building the TexGen Core
6. Building the Libraries – Python
7. Building the Libraries – VTK
8. Building the Libraries – wxWidgets
9. Building the Libraries – Open Cascade
10. The final build
11. Troubleshooting
12. Where to find help





# Introduction



# Introduction

TexGen is a geometry modelling pre-processor software written at the University of Nottingham. It can be used for modelling textile mechanics, permeability and composite mechanical behaviour. Based on modelling theory as set out in [1,2].

It is written in C++ with the functions accessible via a Python interface. It can be built on both Windows and Linux. If you do not wish to build the software, there are binaries available on either the releases section of the TexGen repo on Github:

<https://github.com/louisepb/TexGen/releases>

or on Sourceforge:

<https://sourceforge.net/projects/texgen/files/texgen/>

TexGen can be built in both debug and release modes. Debug mode enables use of the debugger which can be helpful when developing code. Debug builds of Python, VTK and wxWidgets are necessary if you want to build in this mode.



[1] M Sherburn, "Geometric and Mechanical Modelling of Textiles", PhD Thesis (2007)

[2] L P Brown and A C Long. "Modelling the geometry of textile reinforcements for composites: TexGen", Chapter 8 in "Composite reinforcements for optimum performance (Second Edition)", ed. P Boisse, Woodhead Publishing Ltd, 2021, ISBN: 978-0-12-819005-0.

<https://doi.org/10.1016/B978-0-12-819005-0.00008-3>



# Source Code

TexGen is an open source geometry modelling pre-processor software written at the University of Nottingham. It is distributed under a GPL licence.



The source code is hosted on Github: <https://github.com/louisepb/TexGen>



Git can be downloaded online at <https://git-scm.com/downloads>

Set up an account on Github if you don't already have one. With an account you can clone a copy of TexGen and create pull requests and raise issues to be fixed.



Clone/Download the repository to a folder on your computer.



Link to software

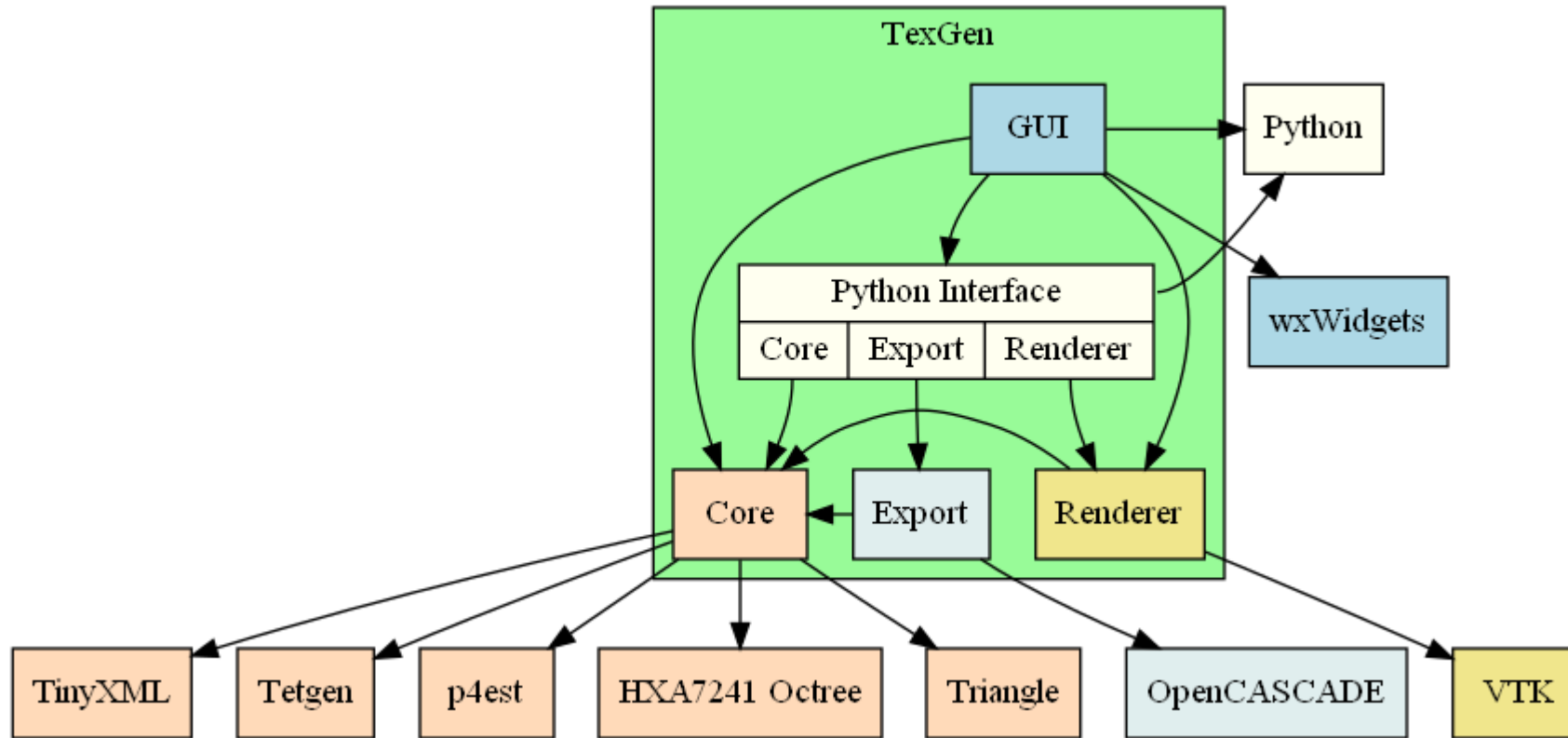


Action item





# TexGen Code Structure

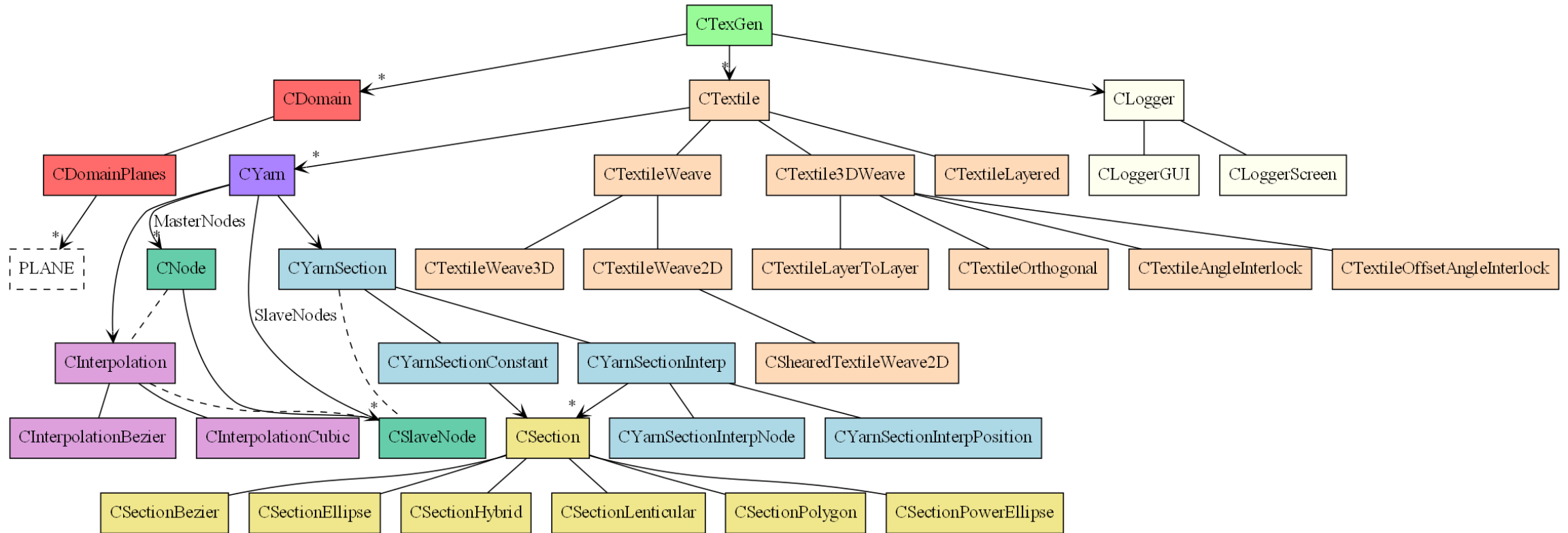


TexGen modules and how they interact with the external dependencies. For a working user interface the build requires Python for the scripting interface, wxWidgets for the user interface and VTK for the rendering.





# Core Class Hierarchy





# Building TexGen

Windows





# Prerequisites

To build TexGen and its dependencies on Windows requires MS Visual Studio. This provides both an IDE and the C++ compilers. The current build was completed using VS 2017. Make sure to check the option to install the Windows SDK.



Visual Studio 2017 can be obtained online with a free account. The install can take some time.



Cmake is used to configure the build environment, selecting 'Add CMake to the system PATH for all users' (select current user if this is not permissible on your system). <https://cmake.org/download/>



SWIG is used to wrap the C++ functions and make them available from the Python interface <https://www.swig.org/>



Download all the prerequisite software

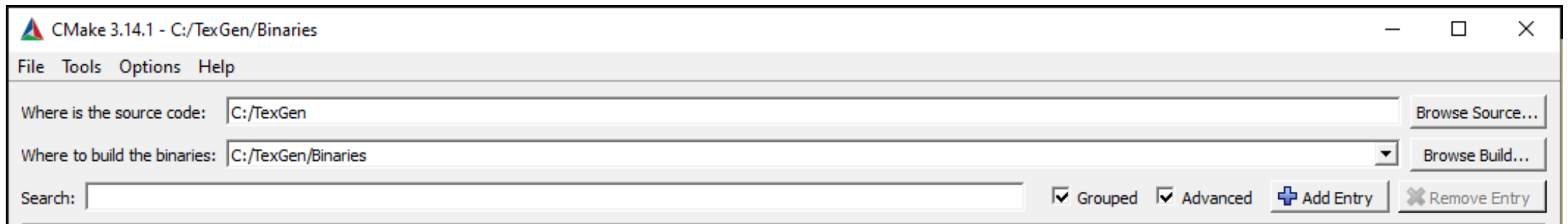


# The Initial Build

To start we will complete a build of the TexGen Core library to check everything is working before adding the external libraries to complete the build later.



Open CMake, the TexGen source code folder and choose a folder to place the binaries in, usually C:/<TexGen Folder Location>/TexGen/Binaries





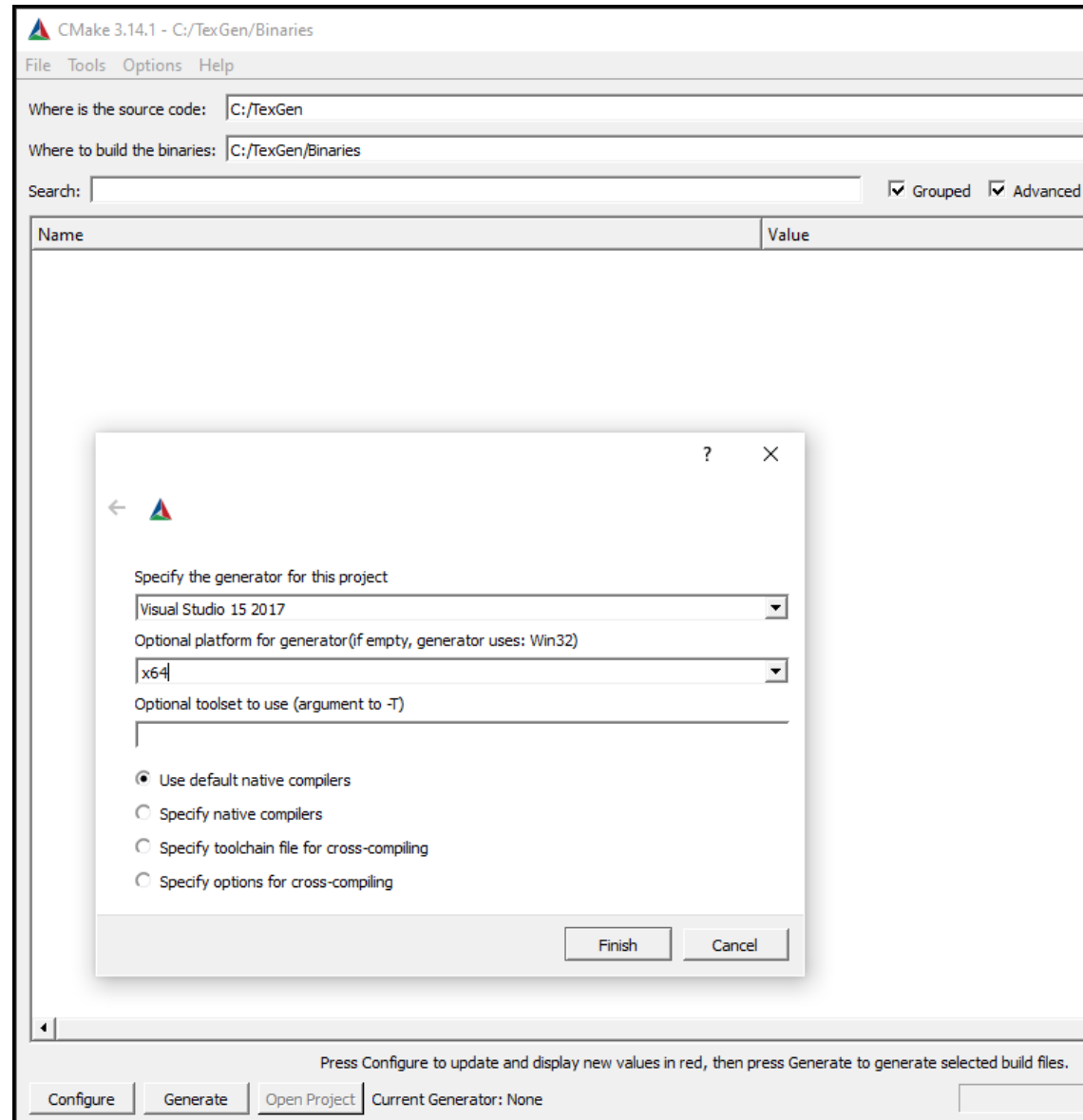
# The Initial Build

To start we will complete a build of the TexGen Core library to check everything is working before adding the external libraries to complete the build later.



Press the "Configure" button and choose the version of Visual Studio you have installed and select x64 if you have a 64 bit register computer.

You will probably get some errors.





# The Initial Build



To resolve the errors, change the Cmake variables so that only "BUILD\_SHARED" is on.

This stops Cmake from looking for the external libraries required for the user interface. We'll get to that later.



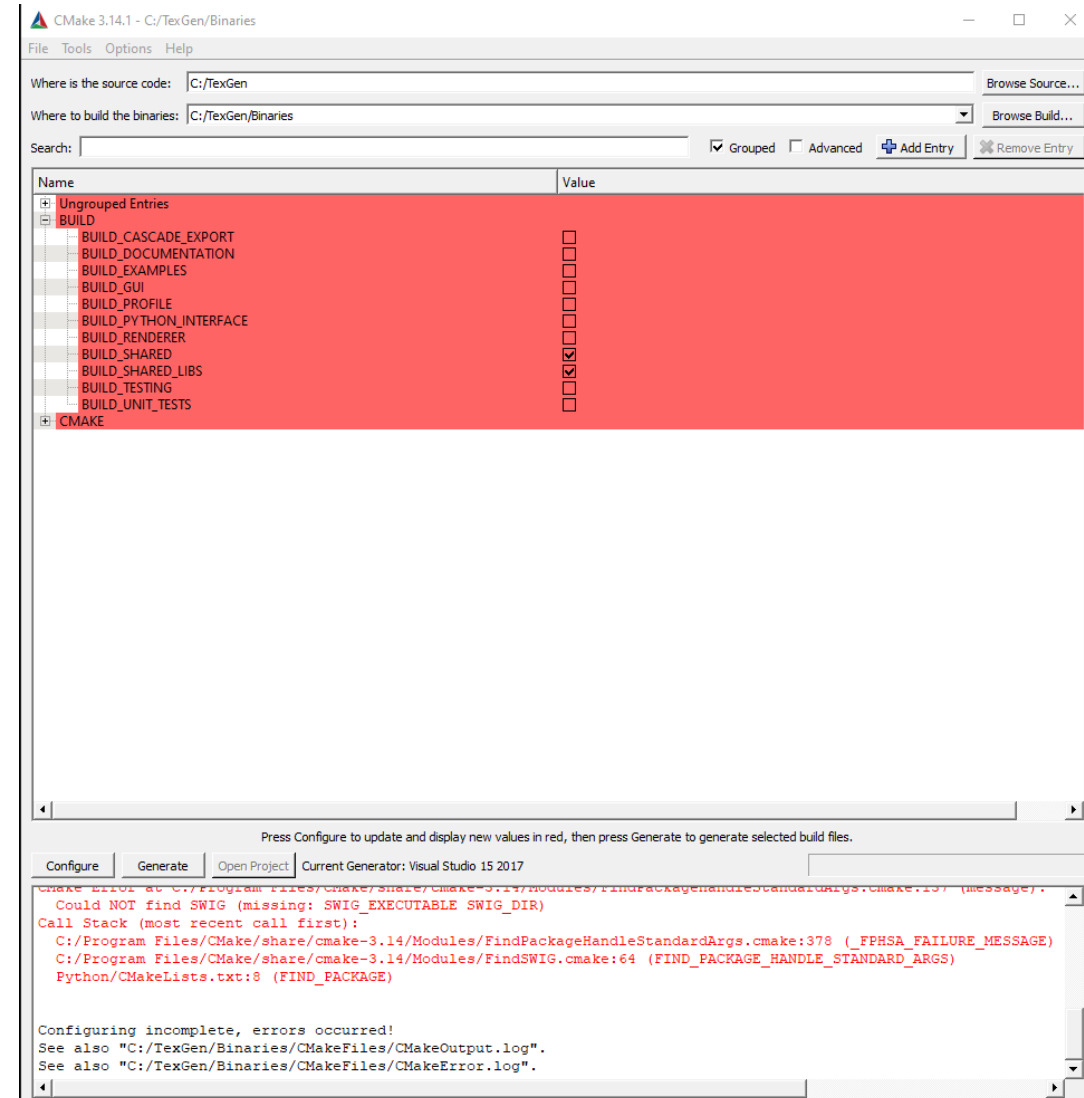
Press "Configure" again. The build errors should now be resolved.



Press "Generate" to generate the Visual Studio solution files.



"Open Project" to open the solution in Visual Studio



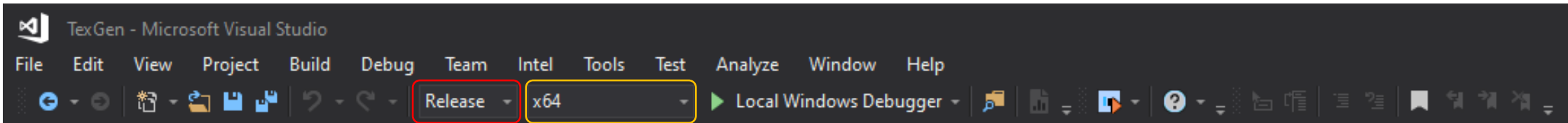


# The Initial Build

To install the software you may need to open visual studio in “Administrator mode”. You can right click on the Visual Studio icon and select run in administrator mode.



In Visual Studio, check that the project files have been configured to build in Release mode and that the platform toolset is correct for your machine.



Check that Release mode and x64 platform are selected.

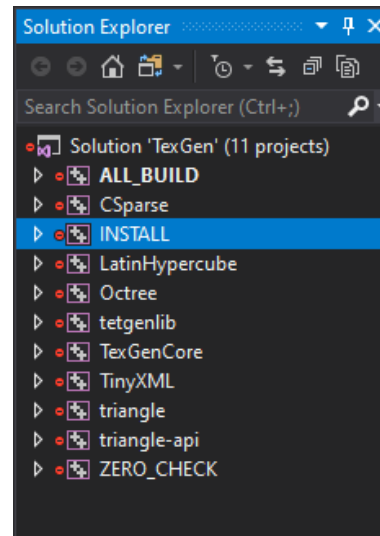
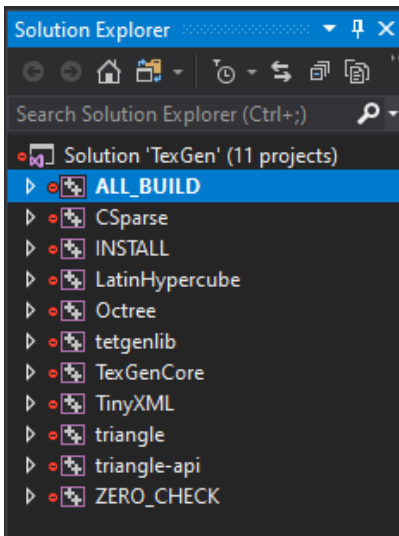


# The Initial Build



In the solution explorer, right click on "All Build" and in the menu click "Build", this will compile the TexGen source code.

If you get a "Set local" error, it's probably because you are not in administrator mode.





# Building the Libraries

Windows





# Python



The new release of TexGen is built with Python 3 and has been tested with version 3.9. To build the release version of TexGen download Python 3 from here: <https://www.python.org/downloads/> then skip to the “Adding Python to the Path” section.

If you want to build TexGen in debug mode and enable debug checks, you’ll need to compile a debug version of Python 3 from source code. Follow the instructions in the next few slides.



The Python source code is hosted on GitHub here: <https://github.com/python/cpython/tree/3.9>



# Compiling Python



Clone the CPython source code: <https://github.com/python/cpython/tree/3.11>

Open a terminal: run cmd.exe

Go the Python directory

To download source code of dependencies (OpenSSL, Tk, etc.): type PCbuild\get\_externals.bat



Type: PCbuild\build.bat -e -d -p x64

-e: download external dependencies (OpenSSL, Tkinter, ...)

-d: build in debug mode (Py\_DEBUG, enable assertions, ...)

-p x64: build in 64-bit mode



Once the compilation process has completed, run the command “PCbuild\python\_d.exe” if Python interpreter opens the process has succeeded.



# Python – Setting up the Directory

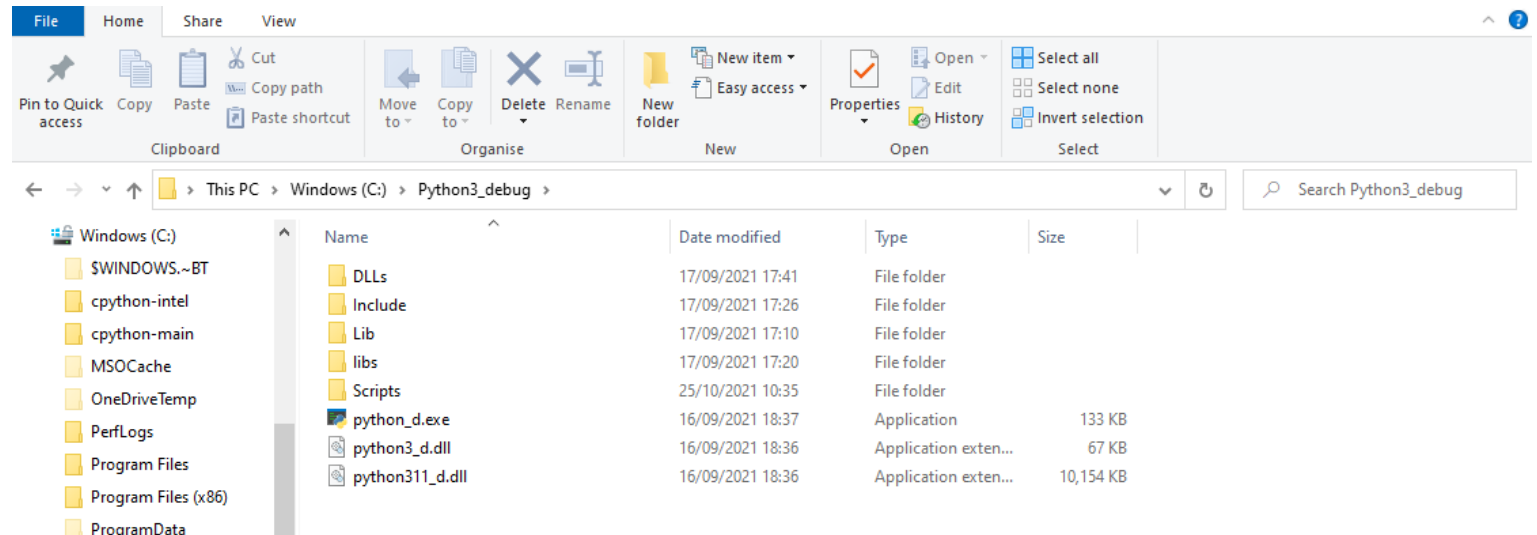


It's best to set up a separate Python folder to put the newly built .dll, .lib and .exe files in. Run the following commands to do so:

```
xcopy C:\cpython\PCBuild\amd64\python_d.exe \Python3_debug\  
xcopy /I C:\cpython\PCbuild\amd64\*.pyd \Python3_debug\DLLs\  
xcopy /I C:\cpython\PCBuild\amd64\*.dll \Python3_debug\DLLs\  
xcopy C:\cpython\PCbuild\amd64\python3_d.dll \Python3_debug\  
xcopy /V /S /E /H /I C:\cpython\Lib \Python3_debug\Lib\.
```

The Python directory should look something like this.

You may need to manually copy over the Include folder from the source directory.

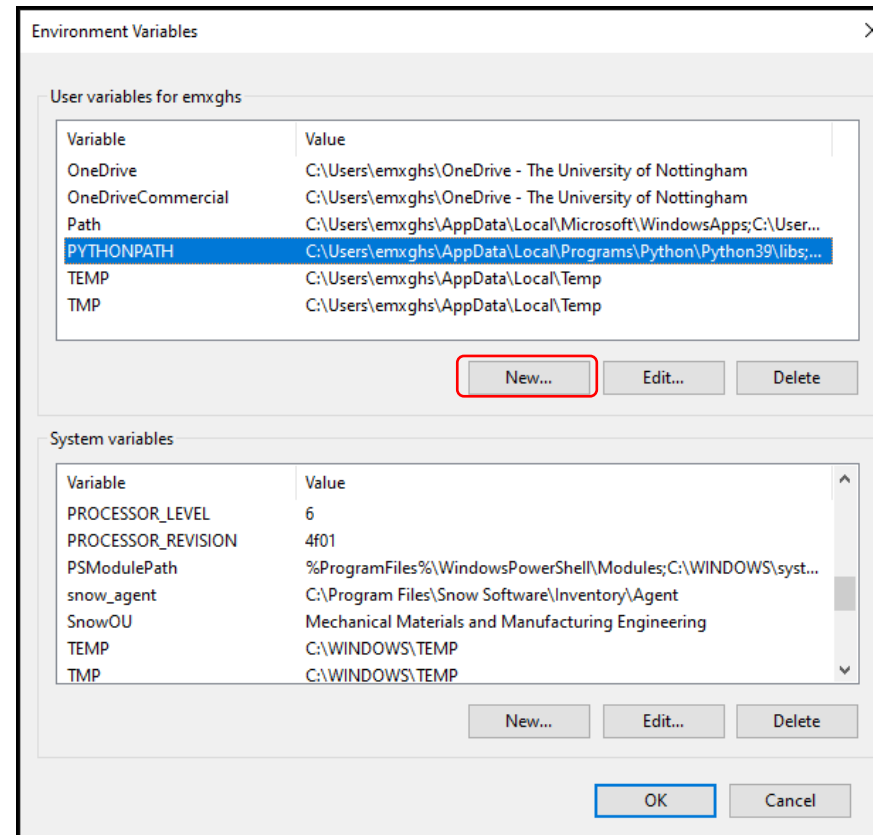
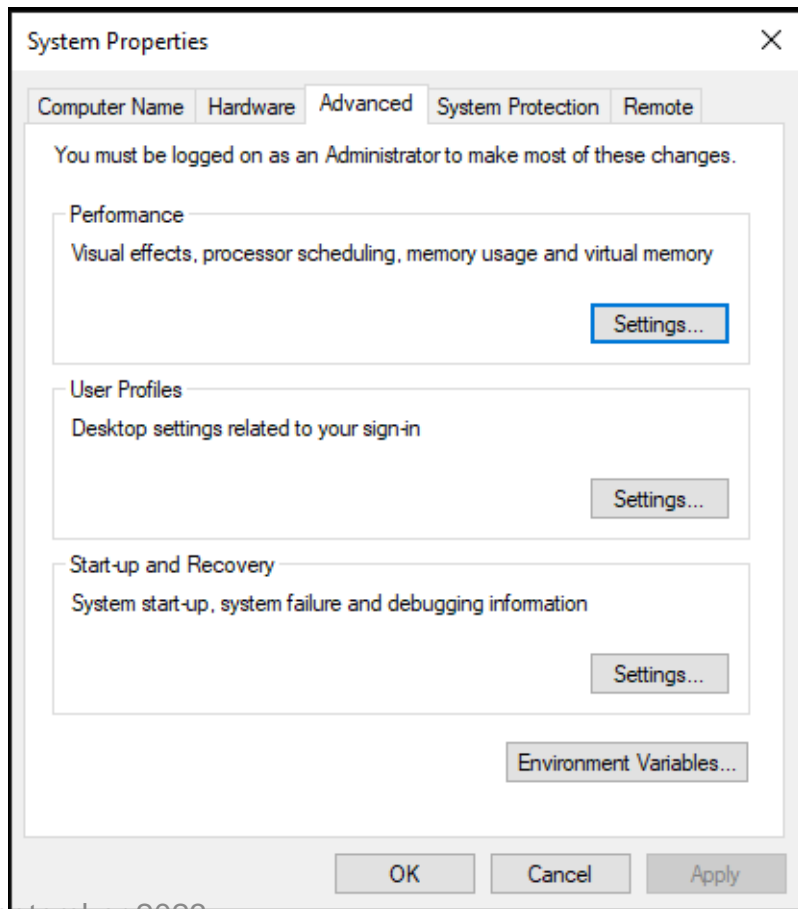




# Python – Adding Python to path



To be able to access the Python interpreter from the command line, add Python to the system path by searching for environment variables from the start menu. Add an additional user variable called PYTHONPATH by selecting New..





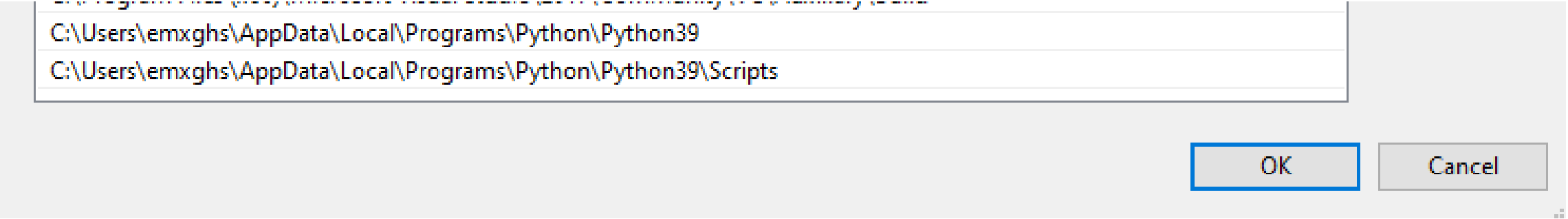
# Python – Adding Python to path



Add the Python interpreter onto the system path by changing the system PATH environment variable to point to the new Python folder. If you compiled Python as in the previous slides it will be C:\Python3\_debug and C:\Python3\_debug\Scripts etc.



If you are using TexGen in release mode having downloaded Python, it will be something like C:\users\<username>\AppData\Local\Programs\Python39\ and C:\ users\<username>\AppData\Local\Programs\Python39\Scripts



C:\Users\emxghs\AppData\Local\Programs\Python\Python39  
C:\Users\emxghs\AppData\Local\Programs\Python\Python39\Scripts

OK

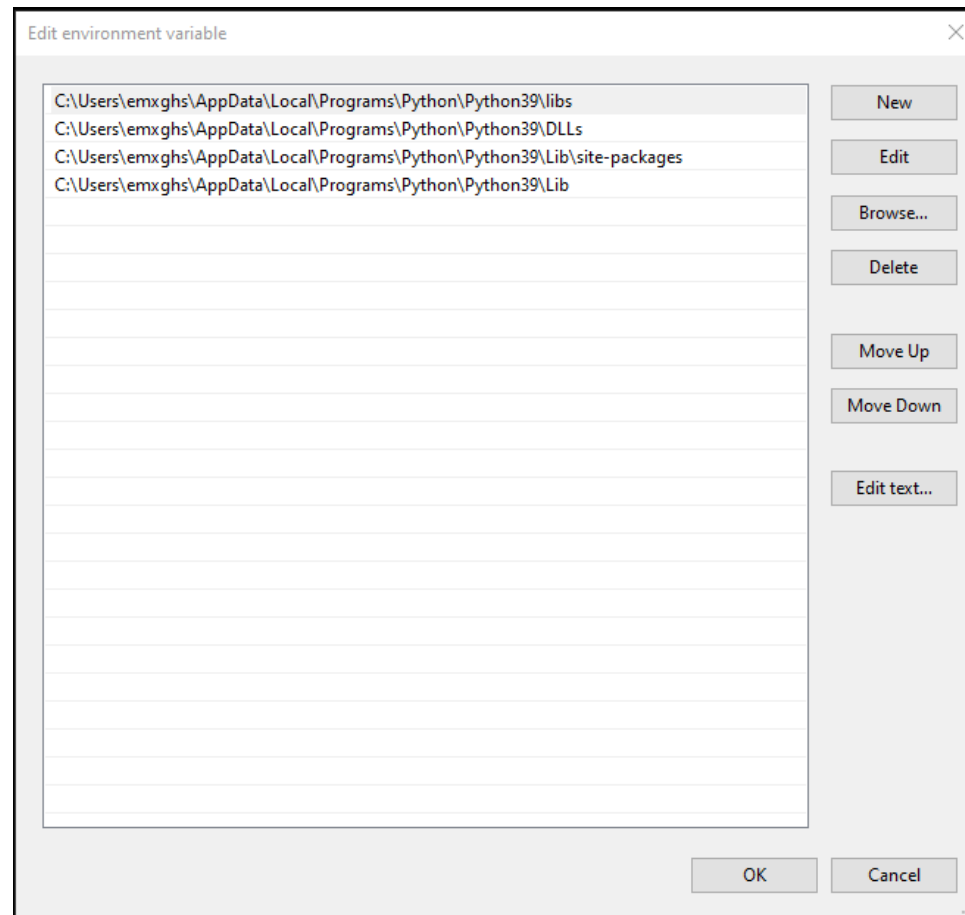
Cancel



# Python – Adding PYTHONPATH variables



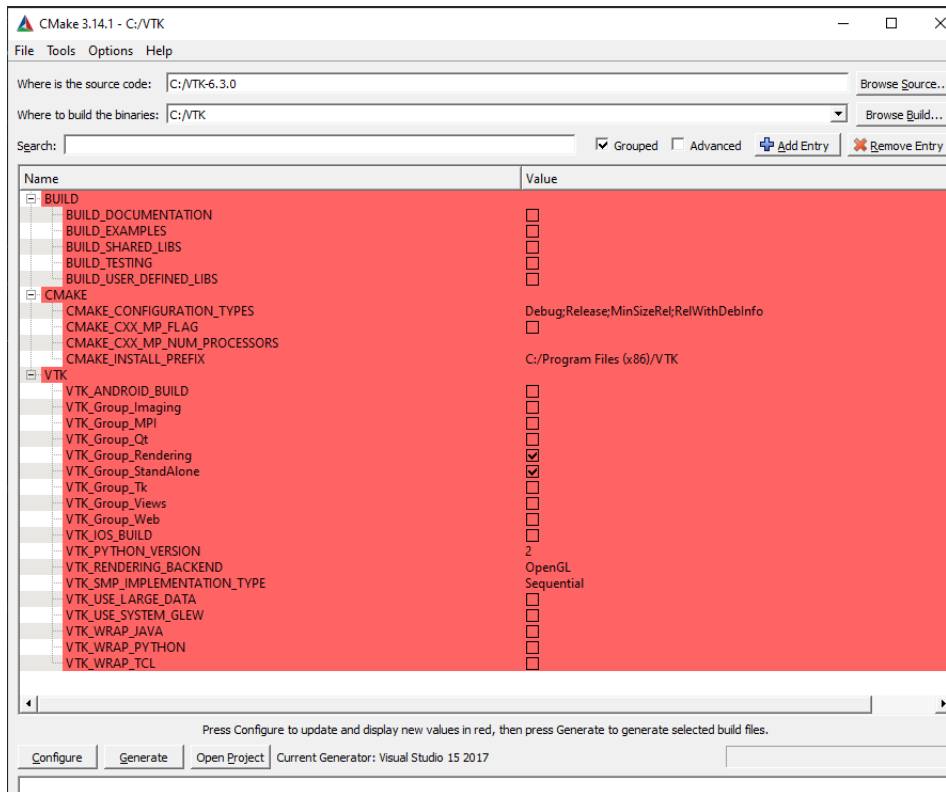
Add the Python library files onto the PYTHONPATH by changing the PYTHONPATH environment variable to point to the new Python folder, if you compiled Python as in the previous slides it will be C:\Python3\_debug and C:\Python3\_debug\Scripts etc.





VTK is used to render the TexGen models in the user interface. This version uses VTK-6.3 and can be found here: <https://github.com/louisepb/VTK-6.3> (This version is no longer available on the VTK download page and is saved here until TexGen is updated to use a more recent version). Download the files and extract them.

Run CMake again, click “Configure” and then make sure the following options are set.



Once completed, open Visual Studio and like before go to “All Build” and right click before selecting “Build”.





Download and install the windows version 3.0.4 of wxWidgets (wxMSW).

<https://www.wxwidgets.org/downloads/>



It is then necessary to compile wxWidgets from source. Detailed instructions on how to do this can be found in 'install.txt' located in the 'docs\msw' subfolder. The main steps are to open 'wx.dsw' with MSVC located in the 'build\msw' subfolder and compile both Release and Debug configurations (The configuration can be selected from the drop-down box on the main toolbar in Visual Studio. Universal and DLL versions need not be compiled).



# The final build

Windows



# Configuring optional modules

Once the additional libraries have been built, they can be used to build the renderer, Python interface and GUI.



Reopen Cmake, selecting the TexGen folders used in the first section. Select the additional modules you want to include. To build the renderer, Python interface and GUI select the renderer, GUI and Python in Cmake and enter the paths for the libraries as shown below (with the correct paths for your installations).

Name	Value
▼ Ungrouped Entries	
TRIANGLE_ENABLE_ACUTE	<input checked="" type="checkbox"/>
VTk_DIR	C:/Users/epzlpb/Documents/VTk-6.3.0-build
> BUILD	
> CMAKE	
> CPPUNIT	
> OPENCASCADE	
▼ PYTHON	
PYTHON_DEBUG_LIBRARY	C:/Users/epzlpb/Python-3.9.9/Python-3.9.9/PCbuild/amd64/python39_d.lib
PYTHON_INCLUDE_DIR	C:/Users/epzlpb/AppData/Local/Programs/Python/Python39/include
PYTHON_LIBRARY	C:/Users/epzlpb/AppData/Local/Programs/Python/Python39/libs/python39.lib
PYTHON_LIBRARY_DEBUG	C:/Users/epzlpb/Python-3.9.9/Python-3.9.9/PCbuild/amd64/python39_d.lib
PYTHON_SITEPACKAGES_DIR	C:/Users/epzlpb/AppData/Local/Programs/Python/Python39/Lib/site-packages
▼ SWIG	
SWIG_DIR	C:/Program Files/swigwin-4.0.2/Lib
SWIG_EXECUTABLE	C:/Program Files/swigwin-4.0.2/swig.exe
SWIG_VERSION	4.0.2
> WX	
▼ wxWidgets	
wxWidgets_CONFIGURATION	mswu
wxWidgets_LIB_DIR	C:/wxWidgets-3.0.5/lib/vc_x64_lib
wxWidgets_ROOT_DIR	C:/wxWidgets-3.0.5
wxWidgets_USE_REL_AND_DBG	<input type="checkbox"/>

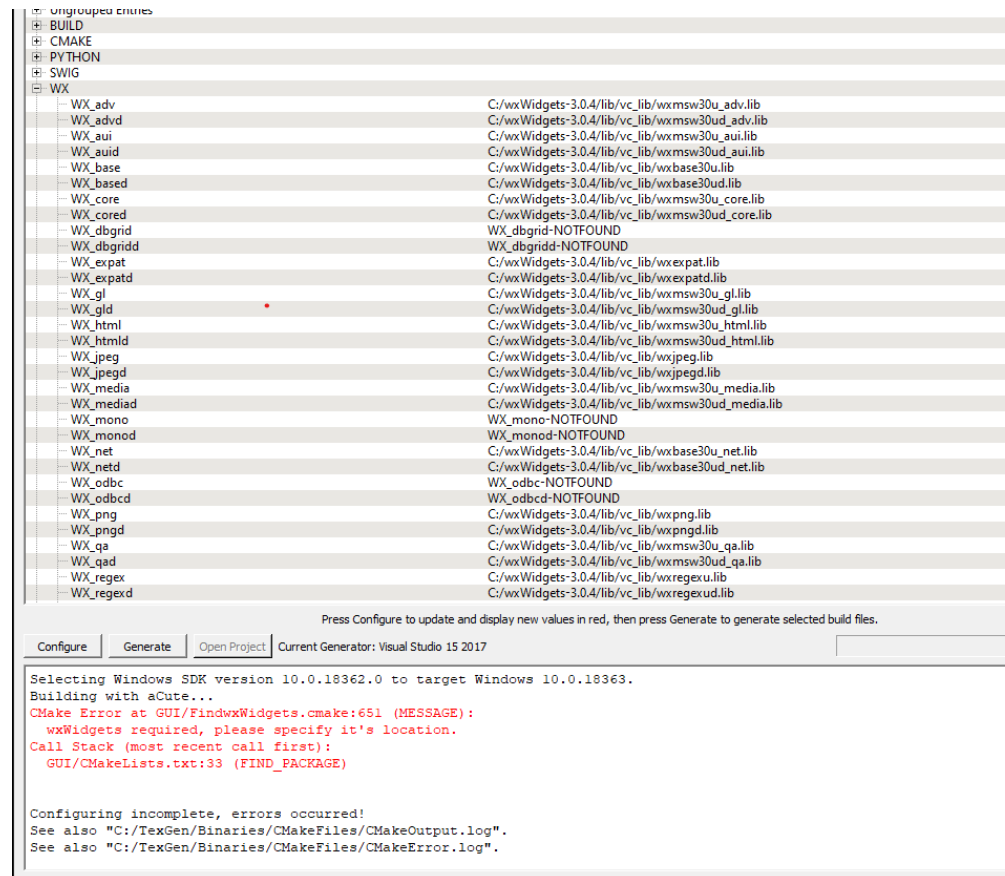
Change  
wxWidgets\_CONFIGURATION  
to mswud for debug builds



# Configuring optional modules



After pressing configure, you may see some errors as Cmake struggles to find the wxWidget libraries. You will have to put them in individually as below.





27