*Created by:* Andrei I Volkov

*Jan 21, 2019*

# Instructions for building cone-detection and cone-segmentation projects

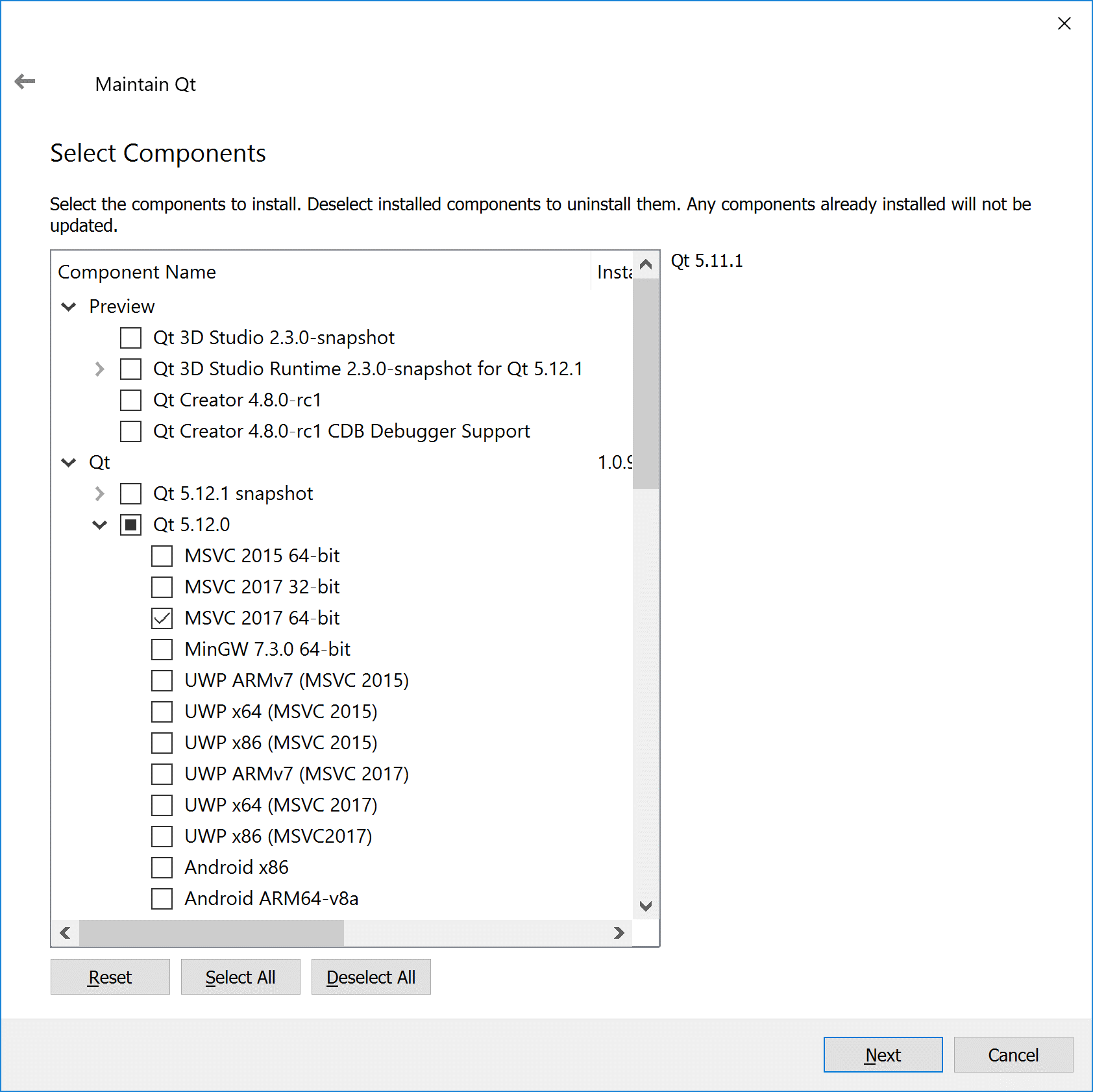
*Skip Windows instructions and jump to:* **Mac OS X**

## **Windows**

*Pre-requisites:* Microsoft Visual Studio 2017, NSIS, CMake 3.x, Qt 5.x, VTK 8.x, ITK 4.x.

1. **Download and install Microsoft Visual Studio 2017** Community or Professional (version 15.x), make sure it contains Visual C++ 2017. Download link: <https://visualstudio.microsoft.com/downloads/>
2. **Download and install the latest version of NSIS**: <https://sourceforge.net/projects/nsis/>
3. **Download and install the latest version of CMake**: <https://cmake.org/download/> for Win64.
4. **Download and install Qt 5**: <https://www.qt.io/download-qt-installer>

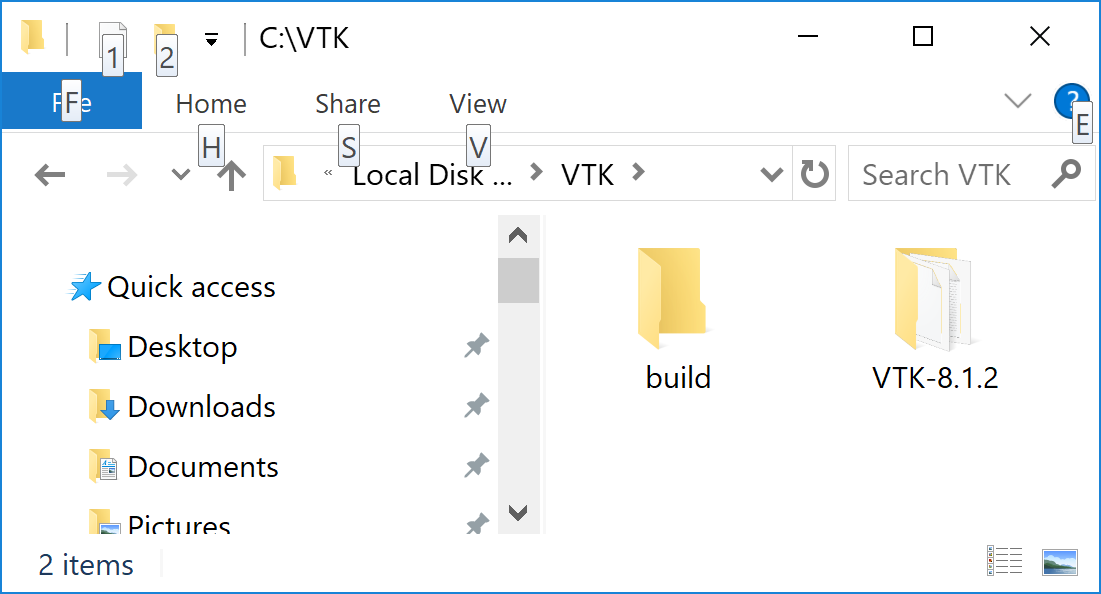
You will need to set up an account (username, email and password) with Qt. If asked, select “Open Source Download” (free). You can use the “recommended” online installer. Select a readily accessible “Installation Folder”, such as **C:\Qt**. You only need to check one of the latest “Qt 5.x, MSVC 2017 64-bit” options:



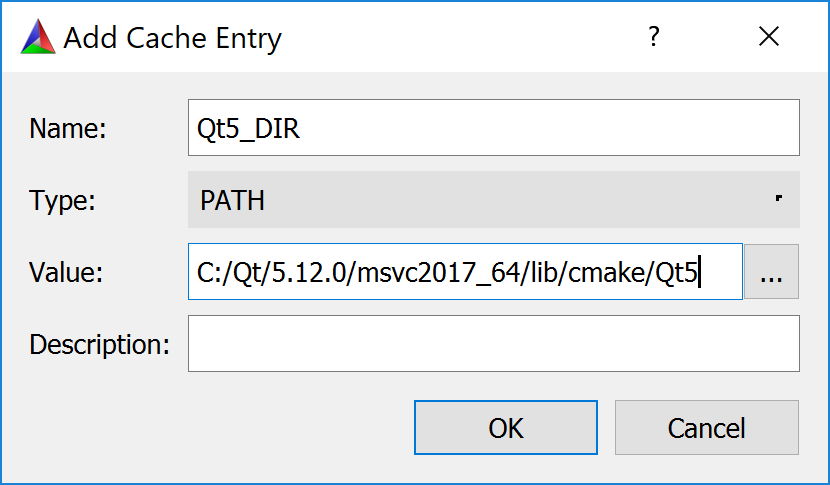
You can run the Qt MaintenanceTool.exe (found in the Qt installation folder) later to install more options or the source code (if you need them for something else).

1. **Download and build VTK 8.x**: <https://vtk.org/download/>

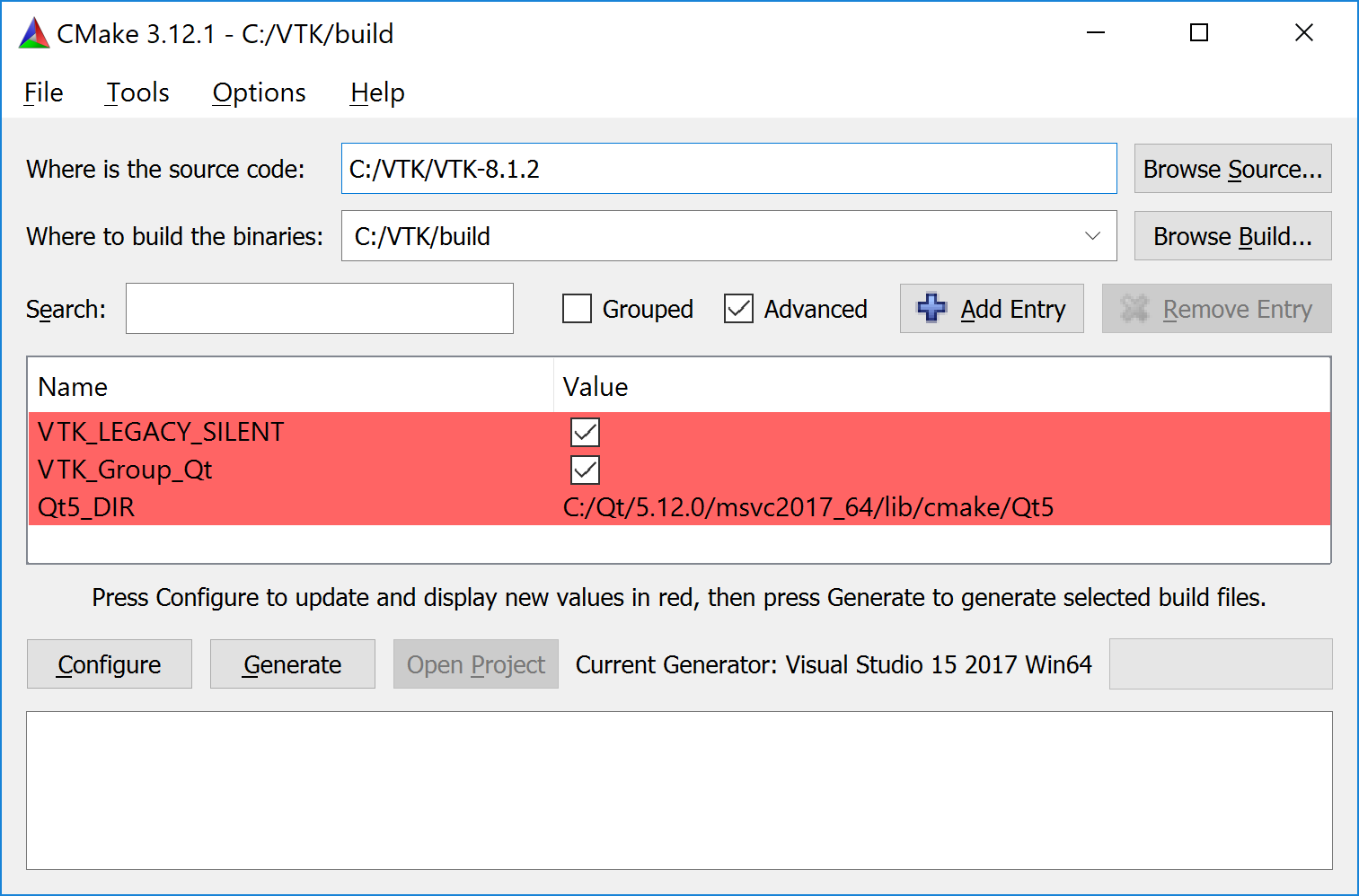
Create a readily accessible directory, such as C:\VTK, and unzip the downloaded VTK archive into it, then create an empty directory build in the same directory:



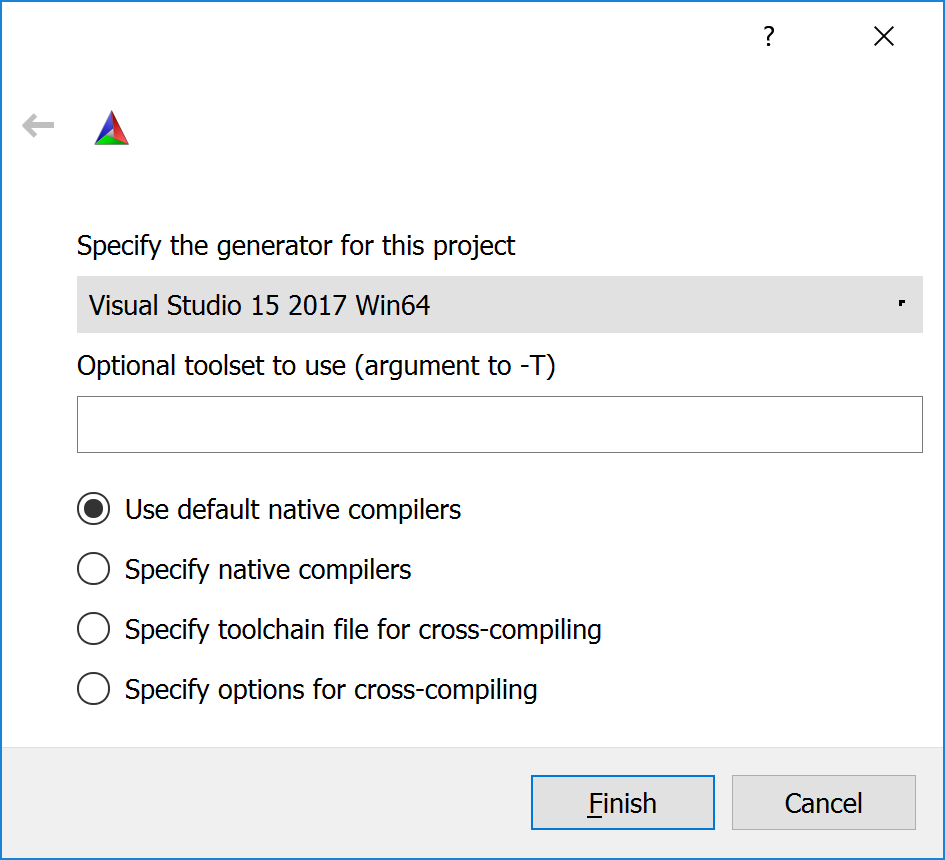
Run CMake, set the source directory to where you unpacked the VTK zip (C:\VTK\VTK-8.1.2 in the above example), and the build directory to the directory you just created (C:\VTK\build). Click “Add Entry” and enter the following:



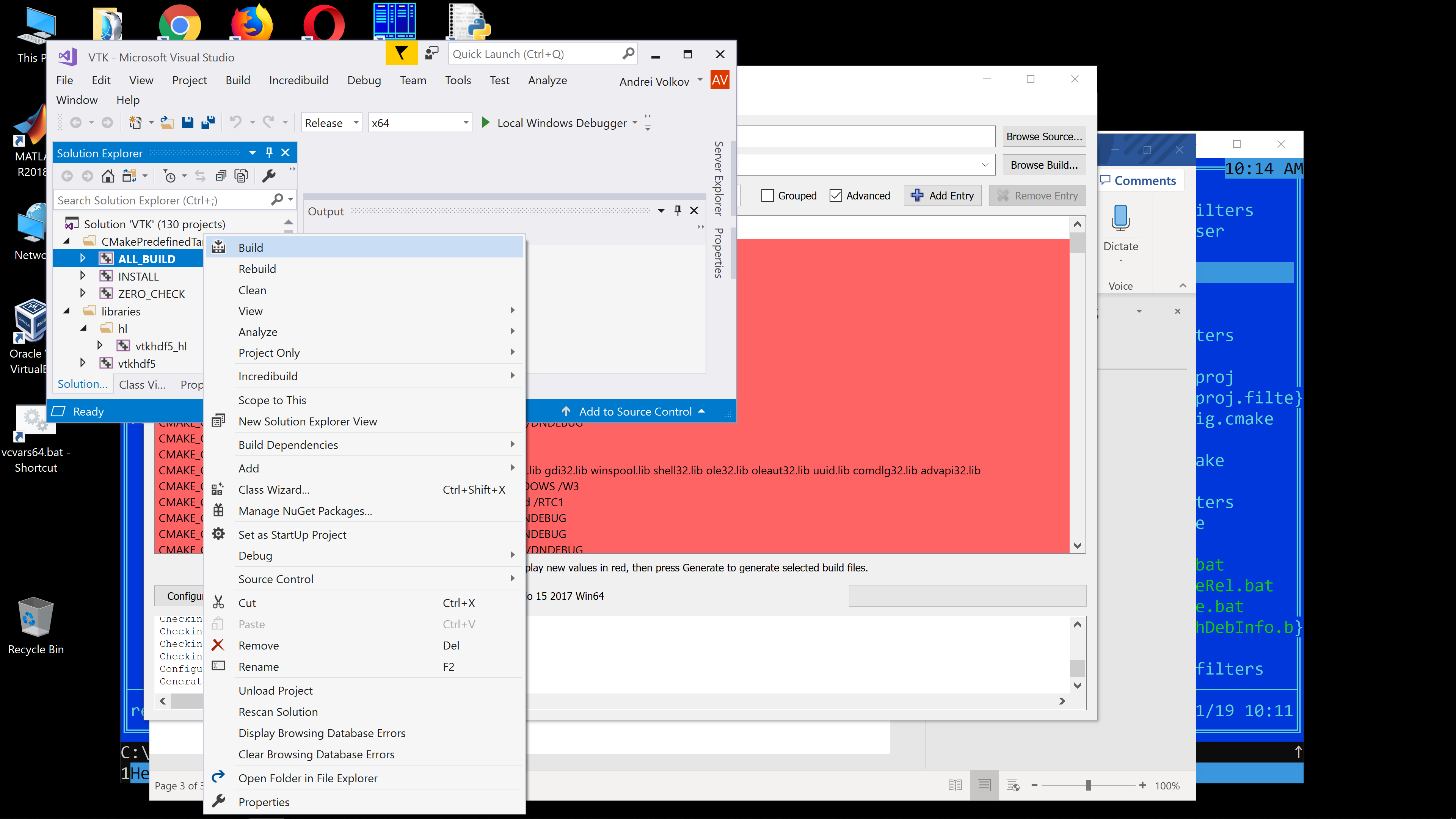
The “Value” box must contain the path to a directory inside Qt installation that contains the file Qt5Config.cmake. You can click on the “…” button and browse for this directory. Repeat “Add Entry” for two more variables, “VTK\_Group\_Qt”, and “VTK\_LEGACY\_SILENT”, set the type to “BOOL” and check the box for both. You should now see something like this:



Click “Configure”. Select “Visual Studio 2017 64 bit” for the compiler:



Then click “Finish”. Configuring will take a while. Once it is done, click “Generate”, then “Open Project”. This will open Visual Studio. Change “Solution Configurations” to “Release”, make sure “Solution Platforms” is set to “x64”, then right-click on “ALL\_BUILD” and select “Build”:



(The build process will take quite some time.)

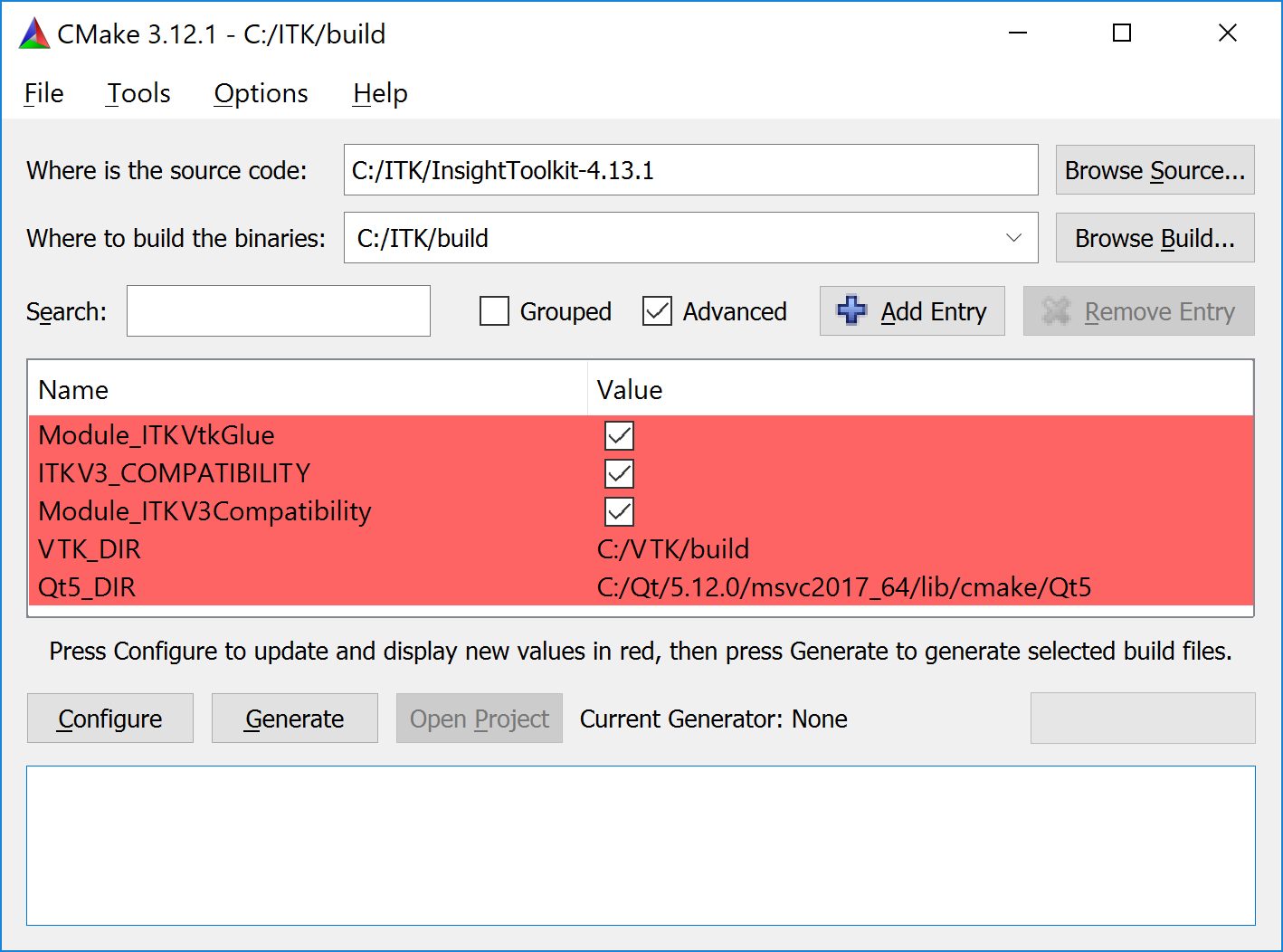
If you are only planning to build the release version of the applications, not to modify/debug them, then you are all set with VTK. Otherwise you may want to build the debug version as well, just switch “Solution Configurations” to “Debug” and build the “ALL\_BUILD” target again. You can also do that later, just open the build\VTK.sln file with Visual Studio, switch to “Debug” and build “ALL\_BUILD”.

1. **Download and build ITK 4.x**: <https://itk.org/ITK/resources/software.html>

This is very similar to VTK. Create a directory (e.g. C:\ITK), unpack the ITK zip into it, and create an empty directory “build”, then run CMake. Set the source directory to the unpacked ITK and the build directory, to your “build”. Add the following variables:

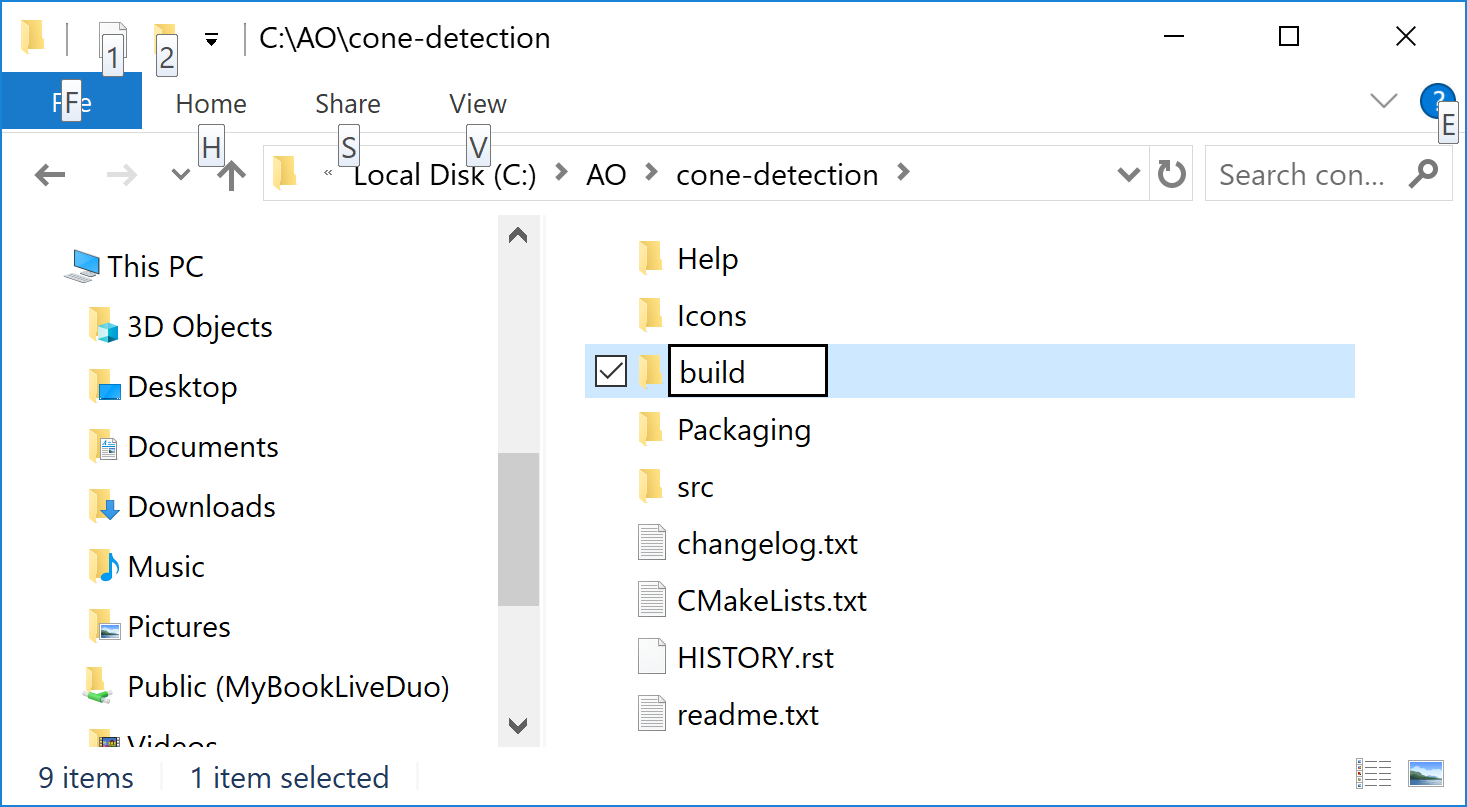
* Qt5\_DIR [PATH] = <path\_to\_qt5>\msvc2017\_64\lib\cmake\Qt5
* VTK\_DIR [PATH] = <path\_to\_vtk\_build>
* Module\_ITKV3Compatibility [BOOL] = On
* ITKV3\_COMPATIBILITY [BOOL] = On
* Module\_ITKVtkGlue [BOOL] = On

You should see something like this:



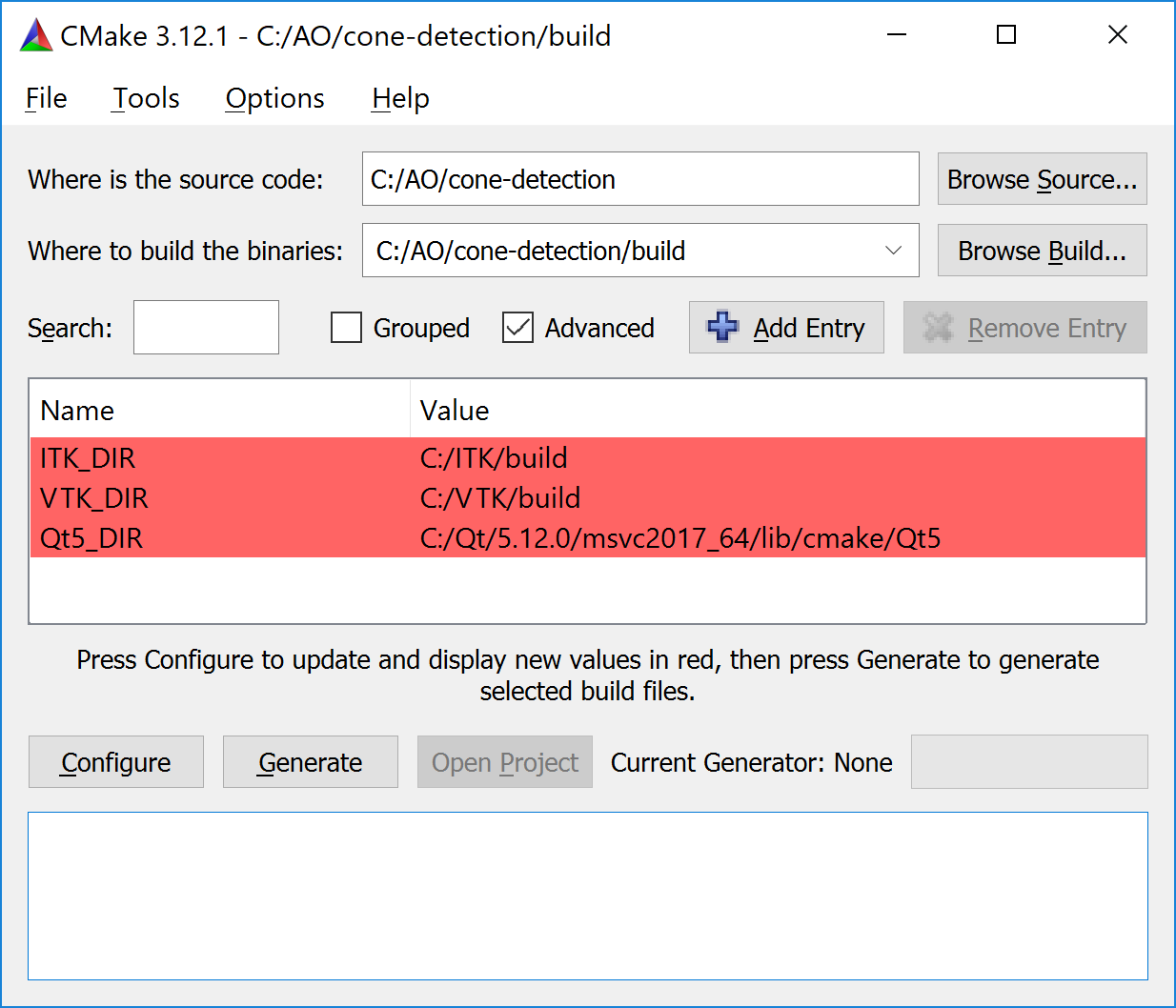
Click “Configure” (select “Visual Studio Win64”; configuring will take a while), then “Generate”, then “Open project”. In Visual Studio, select “Release”, “x64”, then right-click on “ALL\_BUILD” and select “Build”. Like with VTK, if you want to modify/debug your applications, repeat the build process for the “Debug” configuration. You can also do that later, just open the build\ITK.sln file with Visual Studio, switch to “Debug” and build “ALL\_BUILD”.

1. **Build cone-detection**. Suppose you have the source for cone-detection in the directory C:\AO\cone-detection. Open it and create an empty “build” directory inside:



Run CMake, set the source directory to C:\AO\cone-detection, the build directory, to C:\AO\cone-detection\build, and add the following variables:

* Qt5\_DIR [PATH] = <path\_to\_qt5>\msvc2017\_64\lib\cmake\Qt5
* VTK\_DIR [PATH] = <path\_to\_vtk\_build>
* ITK\_DIR [PATH] = <path\_to\_itk\_build>



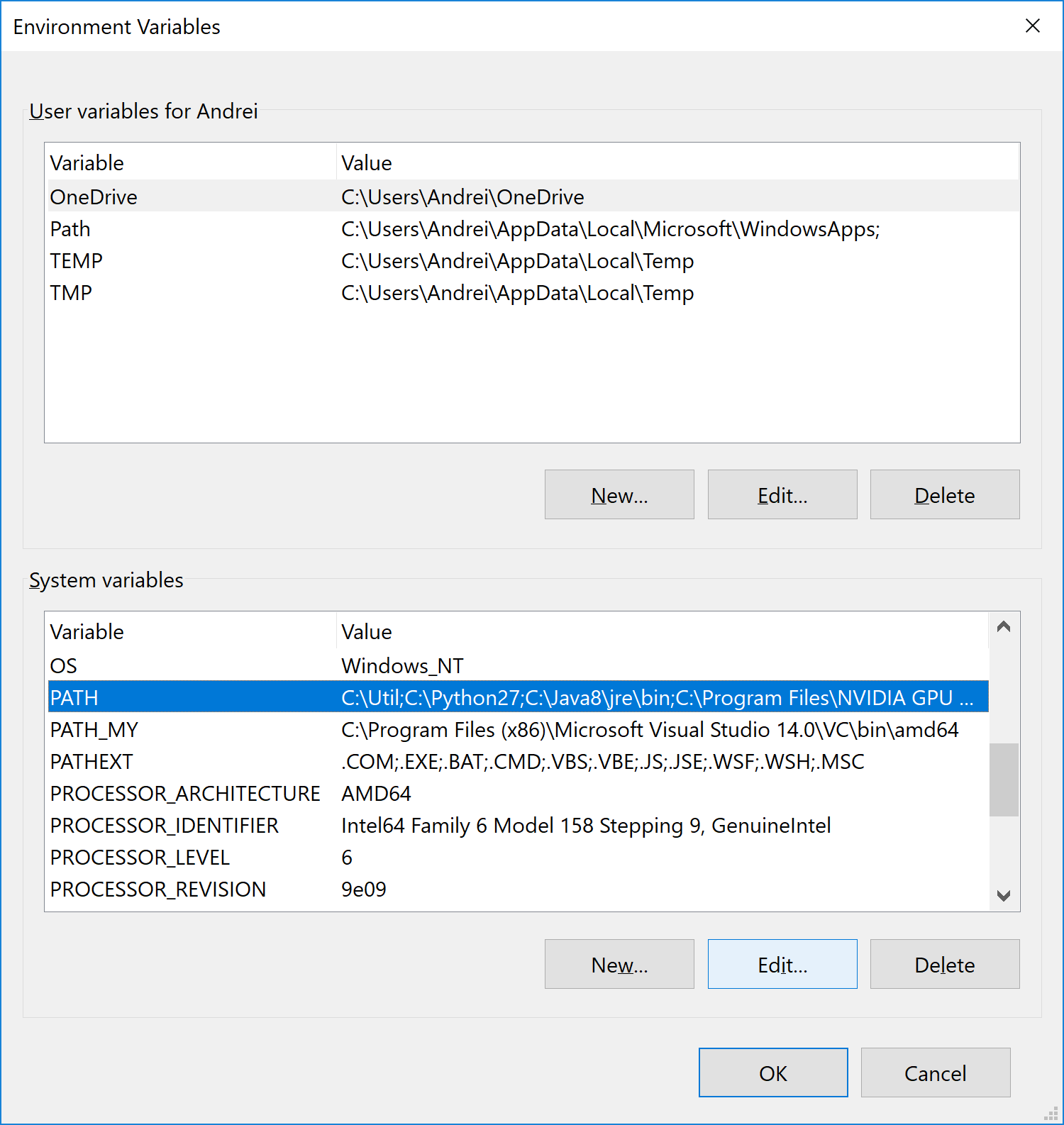
Click “Configure” (select “Visual Studio Win64”), then “Generate”, then “Open Project”. Switch to “Release”, then build “ALL\_BUILD”. The executable will be C:\AO\cone-detection\build\Release\ConeDetection.exe. If you want to create a distribution package, build the target “PACKAGE” (will require NSIS installed in your system). The resulting distribution file will be C:\AO\cone-detection\build\ConeDetection-1.0.0-win64.exe. To install ConeDetection on another computer, transfer this distribution file there, execute it, and follow the installer prompts.

1. **Build cone-segmentation**. Repeat the exact same procedure as for cone-detection but use the source directory for cone-segmentation (C:\AO\cone-segmentation). The executable will be C:\AO\cone-segmentation\build\Release\ConeSegmentation.exe, and the distribution package, C:\AO\cone-segmentation\build\ConeSegmentation-1.0.0-win64.exe.

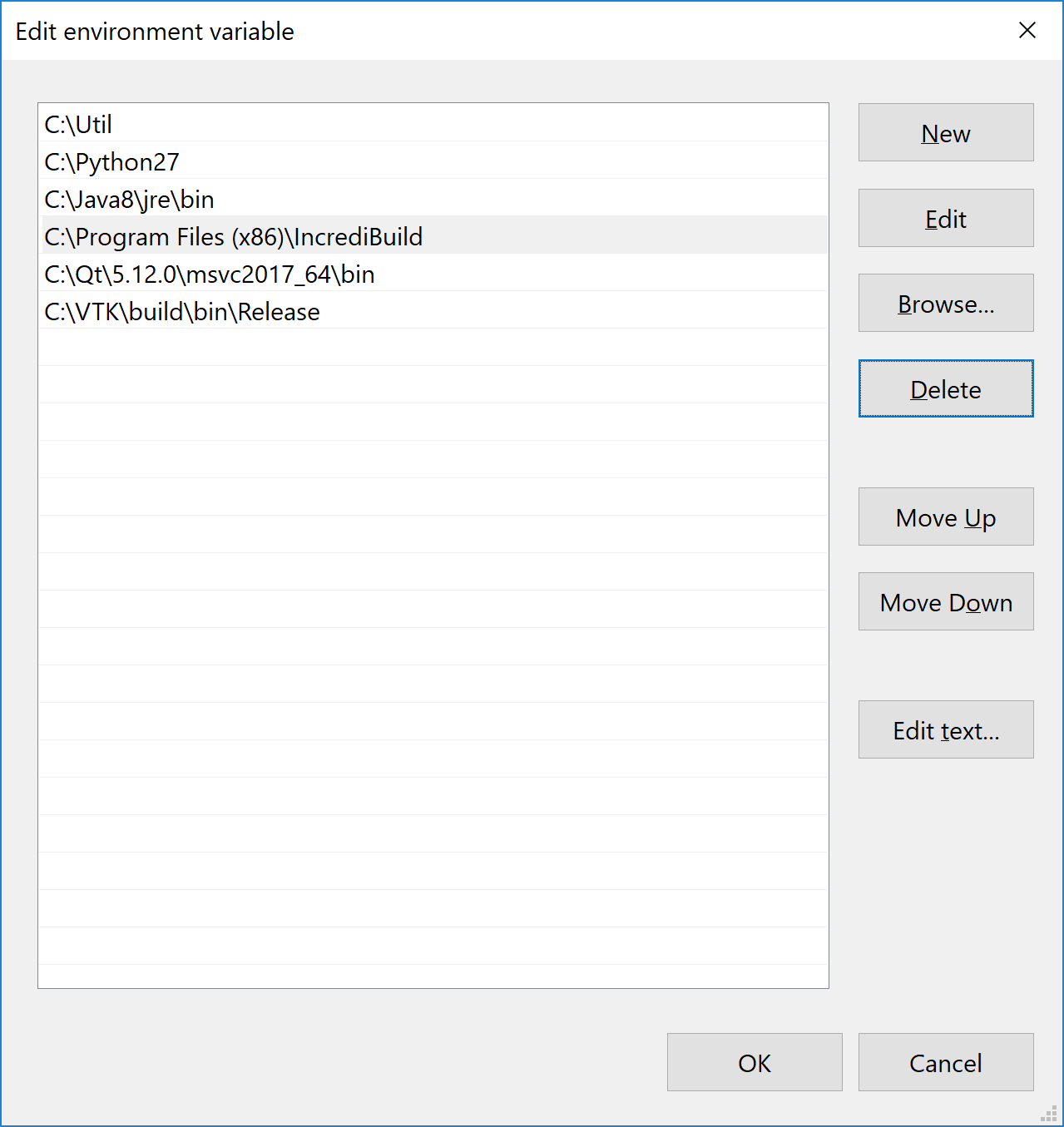
**Notes.**

* If you want to modify/debug the code, make sure you build the debug versions of VTK and ITK libraries. The “PACKAGE” target can be built only in the Release configuration.
* If you plan on developing your own applications using Qt and VTK, it makes sense to add their bin directories (containing DLLs required for running the applications) to the system PATH. While cone-detection and cone-segmentation are configured to copy DLLs to the directory containing the executable, it may not be a good idea for new projects. If the DLLs can be found through system PATH, they don’t need to be copied to the project.

To make them available via PATH, go to Settings, type in “environment” in the search box and select “Edit the system environment variables”. This will open the “System Properties” box. Click the “Environment variables…” button at the bottom. Select “PATH” from the “System variables” box at the bottom, and click “Edit…”:



Click “New”, then “Browse…”. Select Qt “bin” directory. Click “New”, then “Browse…” again, and select VTK “bin/Release” directory. The result will look like this:



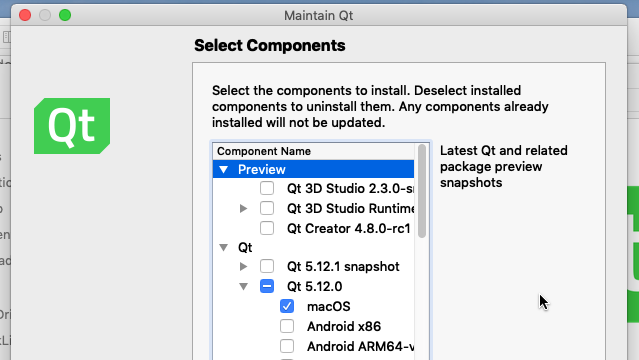
Click “OK”, then “OK” again.

## **Mac OS X**

*Pre-requisites:* Xcode, CMake 3.x, Qt 5.x, VTK 8.x, ITK 4.x.

1. **Download and install Xcode**: <https://developer.apple.com/xcode/>
2. **Download and install CMake**. Go to <https://cmake.org/download/> and select the “.dmg” installer for Mac OS X 10.7 or later, e.g. cmake-3.13.3-Darwin-x86\_64.dmg.
3. **Download and install Qt 5**: <https://www.qt.io/download-qt-installer>

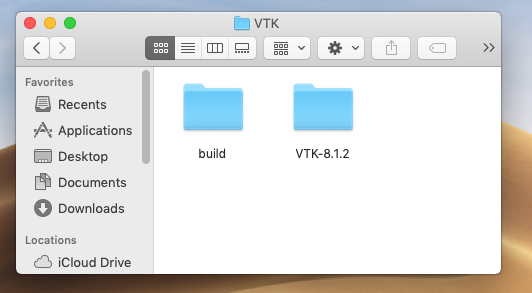
You will need to set up an account (username, email and password) with Qt. If asked, select “Open Source Download” (free). You can use the “recommended” online installer. Select a readily accessible “Installation Folder”, such as /Users/Shared/Qt. You only need to check one of the latest “Qt 5.x, macOS” options:



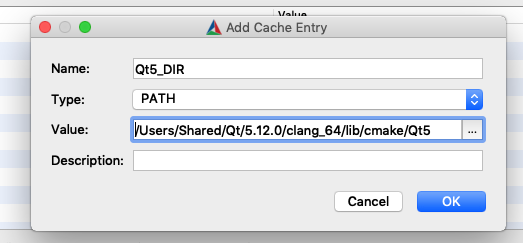
You can run the Qt MaintenanceTool.exe (found in the Qt installation folder) later to install more options or the source code (if you need them for something else).

1. **Download and build VTK 8.x**: <https://vtk.org/download/>

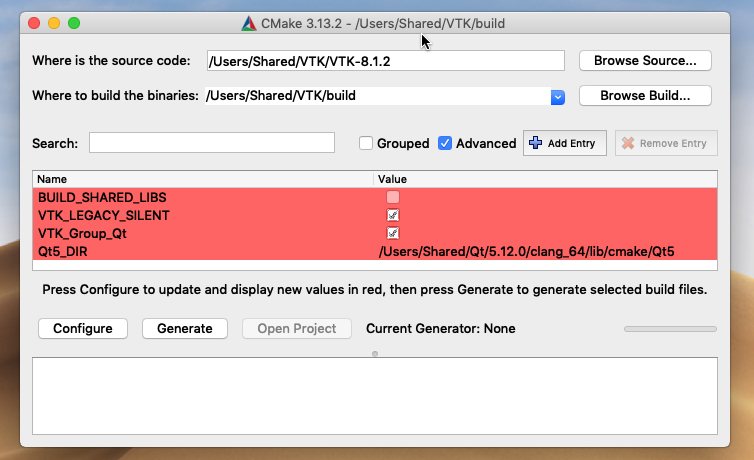
Create a readily accessible directory, such as /Users/Shared/VTK, and unzip the downloaded VTK archive into it, then create an empty directory build in the same directory:



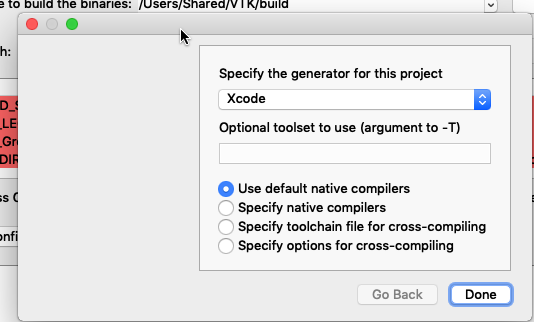
Run CMake, set the source directory to where you unpacked the VTK zip (/Users/Shared/VTK-8.1.2 in the above example), and the build directory to the directory you just created (/Users/Shared/VTK/build). Click “Add Entry” and enter the following:



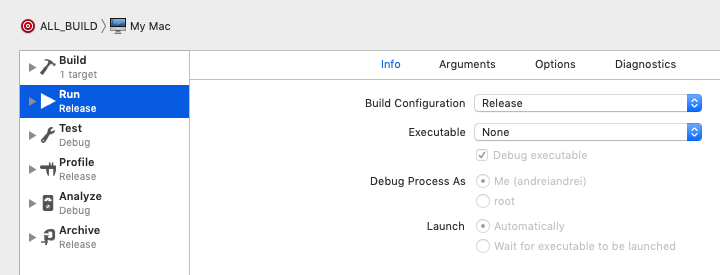
The “Value” box must contain the path to a directory inside Qt installation that contains the file Qt5Config.cmake. You can click on the “…” button and browse for this directory. Repeat “Add Entry” for two more variables, “VTK\_Group\_Qt”, and “VTK\_LEGACY\_SILENT”, set the type to “BOOL” and check the box for both. Add one more entry, “BUILD\_SHARED\_LIBS”, but leave the box unchecked. You should now see something like this:



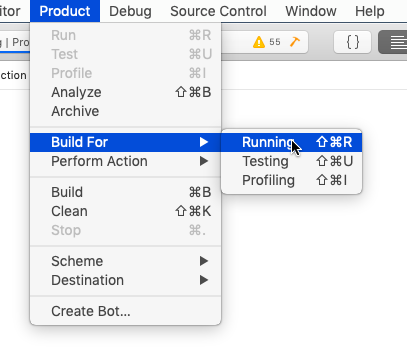
Click “Configure”. Select “Xcode” for the compiler:



Click “Done”. Configuring will take a while. Once it is done, click “Generate”, then “Open Project”. This will start Xcode. When asked about “Autocreate Schemes”, select “Automatically Create Schemes”. Make sure “ALL\_BUILD” is selected in the “current active scheme” box. Go to the menu “Product” -> “Scheme” -> “Edit Scheme…” and switch “Run” to “Release”:



Click “Close”, then start building the project via menu “Project” -> “Build For” -> “Running”:



(The build process will take quite some time.)

If you are only planning to build the release version of the applications, not to modify/debug them, then you are all set with VTK. Otherwise you may want to build the debug version as well, by switch A\_\_BUILD/Run back to “Debug” (via menu Product -> Scheme -> Edit Scheme…) and building it again. You can do that later, just open /Users/Shared/VTK/build/VTK.xcodeproj with Xcode and perform the above manipulations.

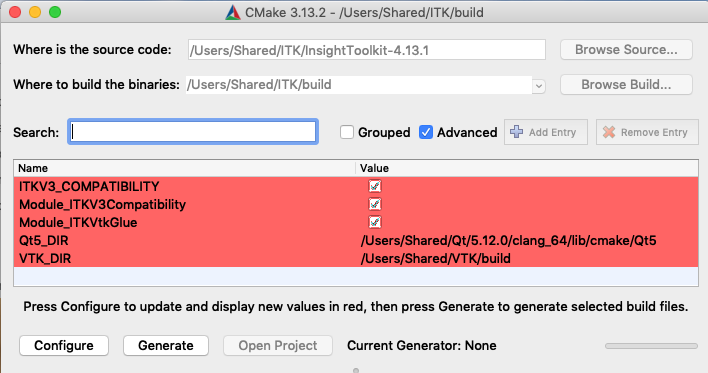
Another thing to note is that unchecked “BUILD\_SHARED\_LIBS” in CMake configuration causes CMake to build statically linked VTK libraries, which is required for cone-detection and cone-segmentation projects (unlike in the Windows environment, where they use dynamic VTK libraries). If you want dynamic VTK libraries in MacOS for other projects, you may want to build VTK into a separate folder (e.g. /Users/Shared/VTK/dynbuild) following the same steps, but with the “BUILD\_SHARED\_LIBS” box in CMake checked.

1. **Download and build ITK 4.x**: <https://itk.org/ITK/resources/software.html>

This is very similar to VTK. Create a directory (e.g. /Users/Shared/ITK), unpack the ITK zip into it, and create an empty directory “build”, then run CMake. Set the source directory to the unpacked ITK and the build directory, to your “build”. Add the following variables:

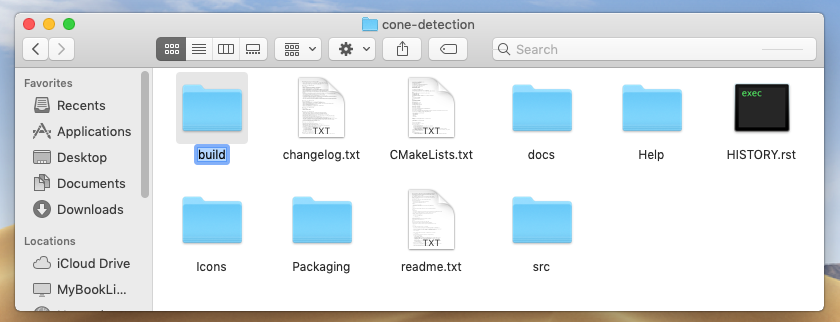
* Qt5\_DIR [PATH] = <path\_to\_qt5>/clang\_64/lib/cmake/Qt5
* VTK\_DIR [PATH] = <path\_to\_vtk\_build>
* Module\_ITKV3Compatibility [BOOL] = On
* ITKV3\_COMPATIBILITY [BOOL] = On
* Module\_ITKVtkGlue [BOOL] = On

You should see something like this:



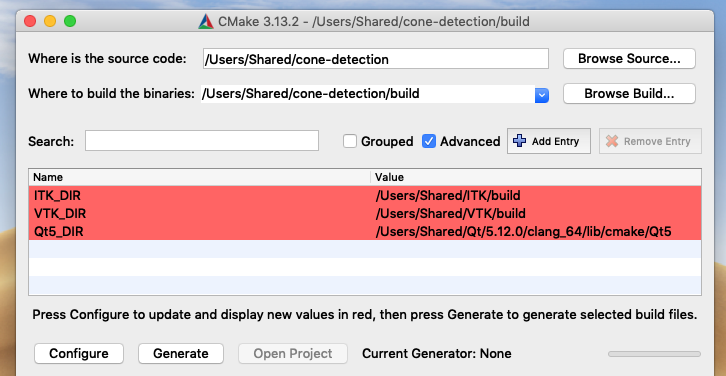
Click “Configure” (select “Xcode”; configuring will take a while), then “Generate”, then “Open project”. In Xcode, make sure “ALL\_BUILD” is selected in the “current active scheme” box. Go to the menu “Product” -> “Scheme” -> “Edit Scheme…” and switch “Run” to “Release”. Click “Close”, then start building the project via menu “Project” -> “Build For” -> “Running”. Like with VTK, if you want to modify/debug your applications, repeat the build process for the “Debug” configuration. You can also do that later, just open the /Users/Shared/ITK/build/ITK.xcodeproj file with Xcode, switch ALL\_BUILD/Run to “Debug”, then build it via Product -> Build For -> Running.

1. **Build cone-detection**. Suppose you have the source for cone-detection in the directory /Users/Shared/cone-detection. Open it and create an empty “build” directory inside:

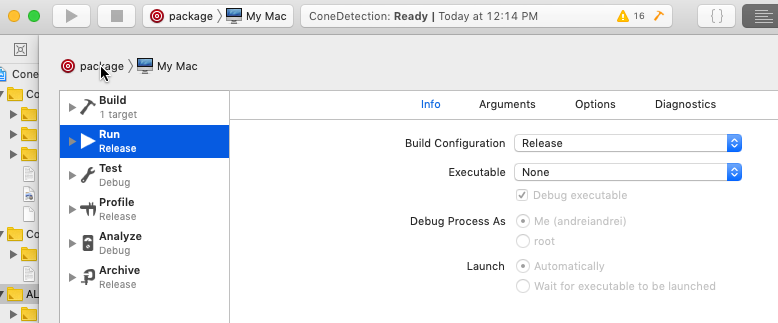


Run CMake, set the source directory to /Users/Shared/cone-detection, the build directory, to /Users/Shared/cone-detection/build, and add the following variables:

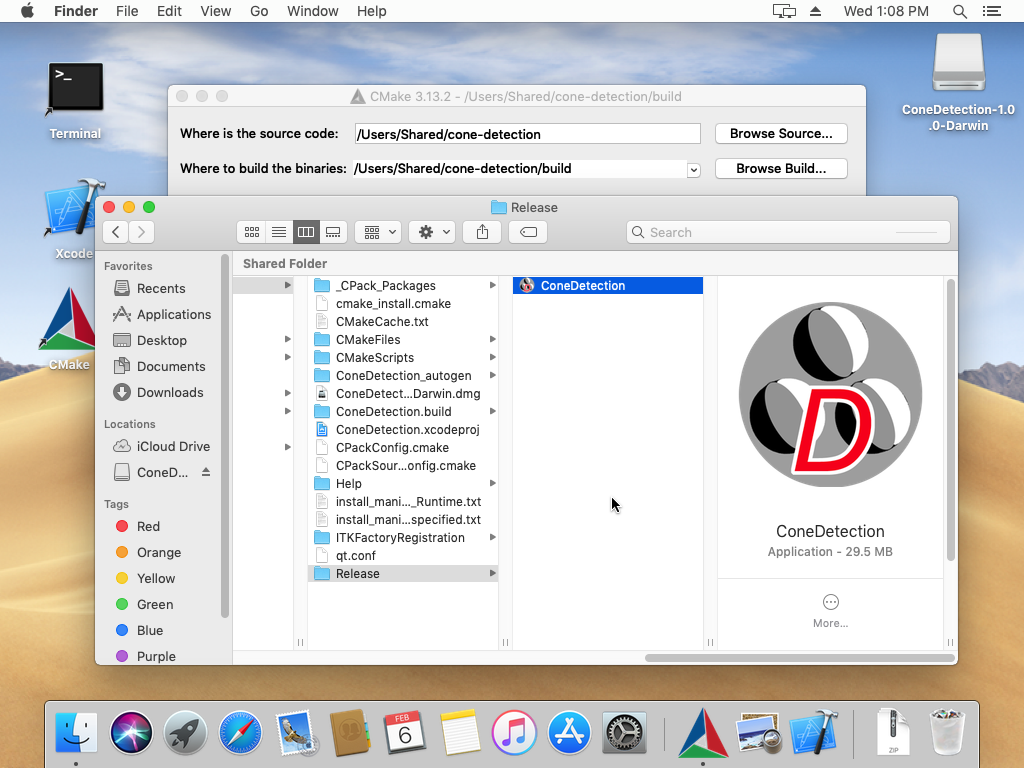
* Qt5\_DIR [PATH] = <path\_to\_qt5>/clang\_64/lib/cmake/Qt5
* VTK\_DIR [PATH] = <path\_to\_vtk\_build>
* ITK\_DIR [PATH] = <path\_to\_itk\_build>



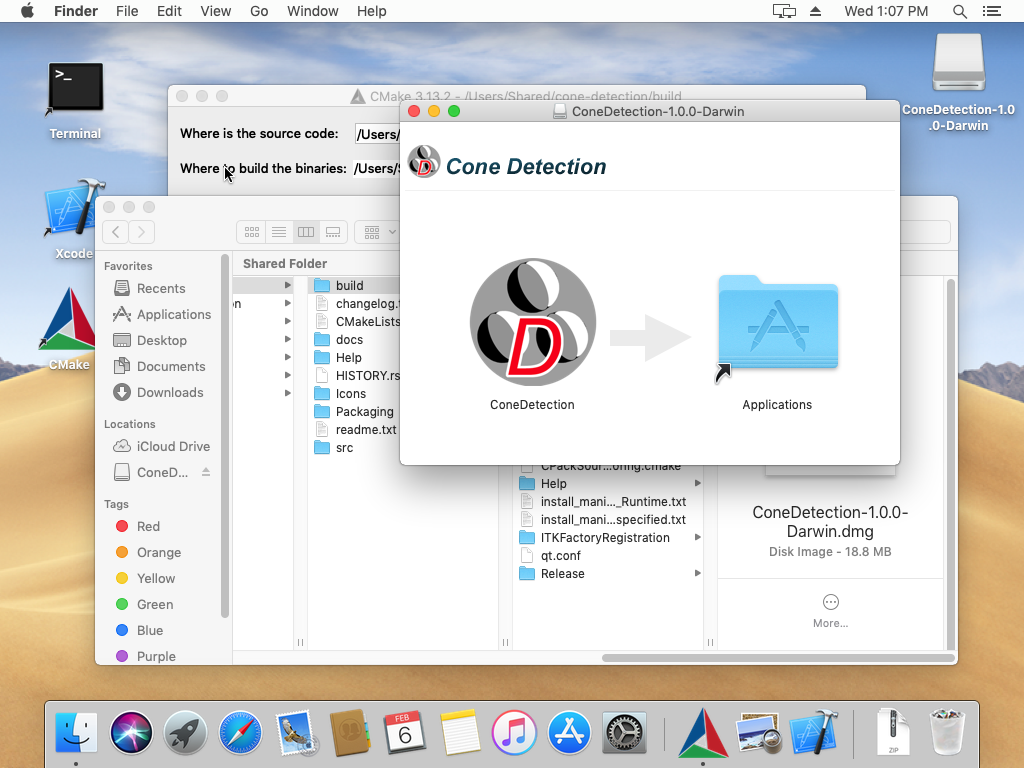
Click “Configure” (select “Xcode”), then “Generate”, then “Open Project”. In Xcode, go to menu “Product” -> “Scheme” -> “Edit Scheme…”, switch the “Run” mode for schemes “ALL\_BUILD”, “install” and “package” to the “Release” configuration. Use the scheme selector in the top left corner of the dialog to switch between schemes:



Click “Close”, switch scheme to “ALL\_BUILD”, and build the project via menu Product -> Build For -> Running. This should generate an executable app /Users/Shared/cone-detection/build/Release/ConeDetection.app



You can test the app, and if everything is OK, go back to Xcode, switch scheme to “package” and build again. While building “package” Xcode may ask you to grant permission to access “Finder” – say “Yes”, since a part of the building process is configuring the appearance of the package using Finder. You will see opening and closing Finder windows with the ConeDetection icon inside during the process. Once it is done, you can find the final distribution package /Users/Shared/cone-detection/build/ConeDetection-1.0.0-Darwin.dmg. This file can be distributed to other computers. If you double-click on it, a device drive will be mounted, and a customized Finder window will show up like this:



To install ConeDetection on this computer, drag and drop its icon over the “Applications” folder (you can drag and drop it on your desktop instead if you want, but don’t spread it to multiple locations, since it will make multiple copies of the app). After that you can “eject” the “ConeDetection-1.0.0-Darwin” drive and delete the .dmg file.

If you want to build a Debug version (open /Users/Shared/cone-detection/build/ConeDetection.xcodeproj in Xcode if it is not already open), switch the ALL\_BUILD/Run scheme back to Debug, and build it again. *Don’t build “install” or “package” in the Debug configuration!*

1. **Build cone-segmentation**. Repeat exact same procedure as for cone-detection but use the source directory for cone-segmentation (e.g. /Users/Shared/cone-segmentation). The executable app will be /Users/Shared/cone-segmentation/build/Release/ConeSegmentation.app, and the distribution package, /Users/Shared/cone-segmentation/build/ConeSegmentation-1.0.0-Darwin.dmg.