Build VTK 5.0 using MinGW/MSys in Windows

Author: Kai-Lun Chung *

1. Build VTK 5.0 in MinGW/MSys in Windows

1.1. Build steps

1. Download and install **MinGW** and **MSys**.

   The link can be found in the download section in http://www.mingw.org/

   *(Note: Although not quite accurate, you can think **MinGW** is a thin abstraction layer which allows you to port softwares in Unix-like environment to Windows with the minimal changes of code. **MSys** provides a Unix-like shell environment, just like bash/csh).*

2. Download the latest source code of VTK in http://www.vtk.org/

   Throughout this tutorial, the latest stable release of VTK is 5.0.4

3. Unzip the source package to a directory in your file system

   Throughout this tutorial, I unpack the source code to C:\Downloads\firefox\vtk-5.0.4

4. Edit the source code

   To get compiled in MinGW/MSys, a minor modification on the source code is required. Don't be too panic on that, the changes are very minor and should not affect the architecture of VTK. It just some tricks to make it pass the gcc in MinGW. Two files are required to modified:

   C:\Downloads\firefox\vtk-5.0.4\CMakeLists.txt

   Append the following command to CMakeList.txt: SET(VTK_TYPE_USE__INT64 0)

   Reason: In MingGW, __int64 is actually equivalent to long long, but the CMake script seems not handle it quite well, and may generate compilation errors.

   C:\Downloads\firefox\vtk-5.0.4\Utilities\vtkjpeg\jconfig.h

   Locate the following source code:

```
#if defined(_WIN32) && !(defined(__CYGWIN__) || defined(__MINGW32__))
/* Define "boolean" as unsigned char, not int, per Windows custom */
/* don't conflict if rpcndr.h already read; Note that the w32api headers
   used by Cygwin and Mingw do not define "boolean", so jmorecfg.h
   handles it later. */
#ifndef __RPCNDR_H__
typedef unsigned char boolean;
#endif
#define HAVE_BOOLEAN            /* prevent jmorecfg.h from redefining it */
#endif
```

   Modify the block as follow:

```
#if defined(_WIN32) && !(defined(__CYGWIN__))
/* Define "boolean" as unsigned char, not int, per Windows custom */
/* don't conflict if rpcndr.h already read; Note that the w32api headers
   used by Cygwin and Mingw do not define "boolean", so jmorecfg.h
   handles it later. */
#ifndef __RPCNDR_H__
typedef unsigned char boolean;
#endif
#define HAVE_BOOLEAN            /* prevent jmorecfg.h from redefining it */
#endif
```

Reason: In MinGW, boolean should be already defined as unsigned char, so we are no need to redefine it again

(If you don't want to do that troublesome editing work....)

The modified two files can be downloaded here, just download and replace the old one:

http://code.hkpeterpeter.googlepages.com/CMakeLists.txt

http://code.hkpeterpeter.googlepages.com/jconfig.h

5. Download and install **CMake**

   The link can be found in the download section in http://www.cmake.org/

   Version 2 or above should be okay, version 2.6 is used in this tutorial

6. CMake configuration

   Set paths correctly (see figure 1). The option "where to build the binary" just means that where to generate the platform dependent codes to build VTK. You can choose any new directory. Then press configure, choose "MSys Makefiles" (see figure 2). For default installation, you only need to change one option (see figure 3 and 4). Then press configure again and wait. Then press "OK". The platform dependent code will be generated *(In my case, it is generated at C:\Downloads\firefox\vtk-5.0.4-msys)*

7. Build VTK using make in **MSys**

   Now, open **MSys**, go to the directory with platform dependent codes (see figure 5). Type "make". It takes quite a long time (see figure 6). After that, you can type "make install" to move the header files and the library files to the selected directory. *(In my case, I install VTK in C:\msys\1.0\VTK)*

8. Download the minimal VTK project in MinGW to test

   The minimal VTK project for MinGW can be grabbed here:

   http://code.hkpeterpeter.googlepages.com/vtk_minimal_mingw.zip

   Unzip to the directory you like. Run **MSys**. Locate to that directory. Edit the **Makefile** according to the following rule:

```
CXXFLAGS = -Wno-deprecated

# -I [include path]: The path to your include directory.
# The root directory means C:\msys\1.0
INC = -I /VTK/include/vtk-5.0

# -L [library path] -l[dep library] :
# The path to your library directory.
# More dependent libraries can be linked here..
LIB = -L /VTK/lib -mwin32 -mthreads   -mthreads -mwindows \
-lvtkRendering -lvtkGraphics -lvtkImaging \
-lvtkIO -lvtkFiltering -lvtkCommon \
```

```
-lvtkftgl -lvtkfreetype -lopengl32 -lvtkDICOMParser \
-lvtkpng -lvtktiff -lvtkzlib -lvtkjpeg -lvtkexpat \
-lvfw32 -lvtksys -lgdi32 -lstdc++

# Build rules
minimal: vtk_minimal.cpp
gcc $(CXXFLAGS) -c vtk_minimal.cpp $(INC)
gcc -o minimal vtk_minimal.o $(LIB)

clean:
rm -f *.o minimal.exe
```

After editing the **Makefile**, run **make**. The executable **minimal.exe** should be created. Then type in **./minimal** to run it. (see figure 7)

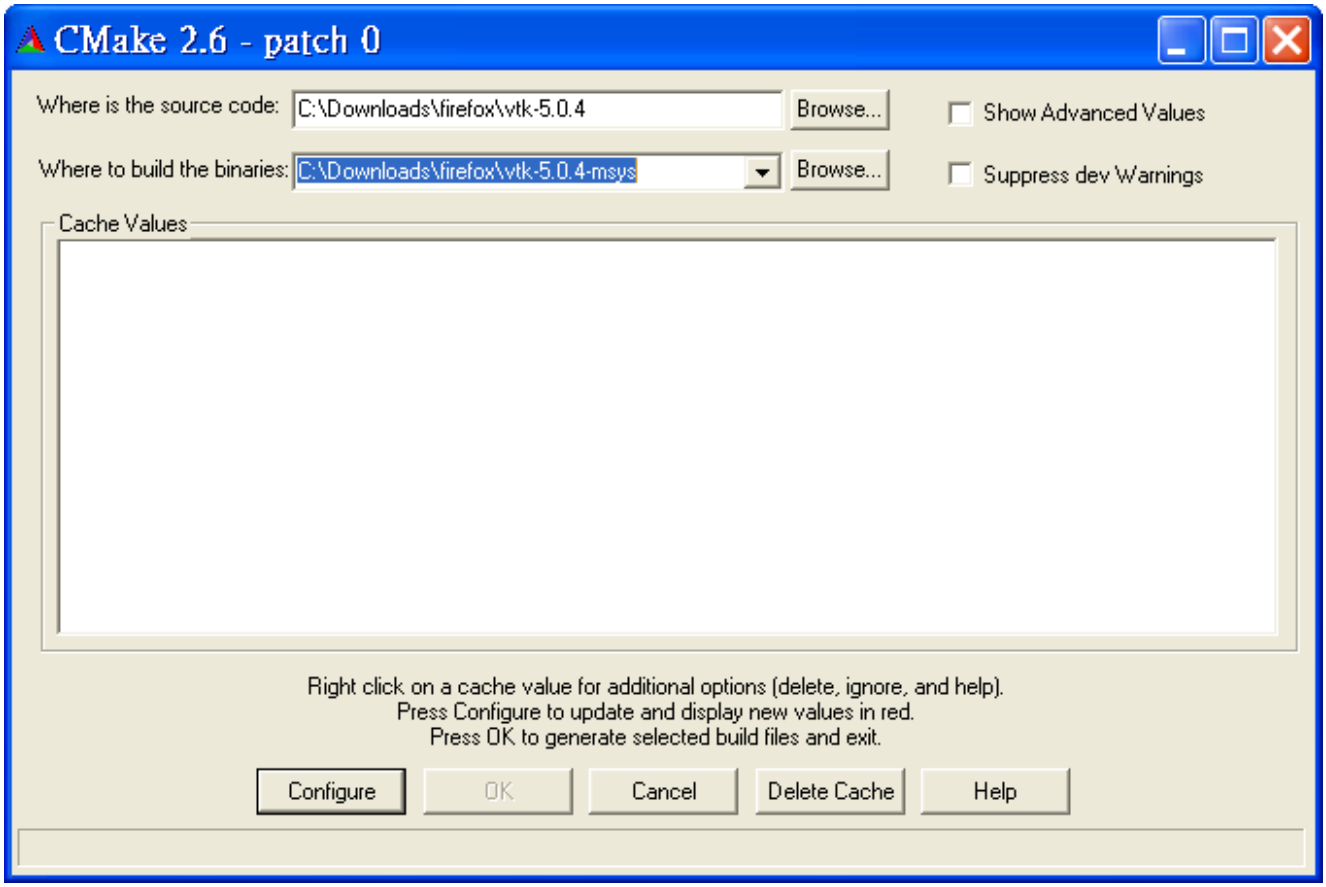

Figure 1: CMake, set configuration pathes

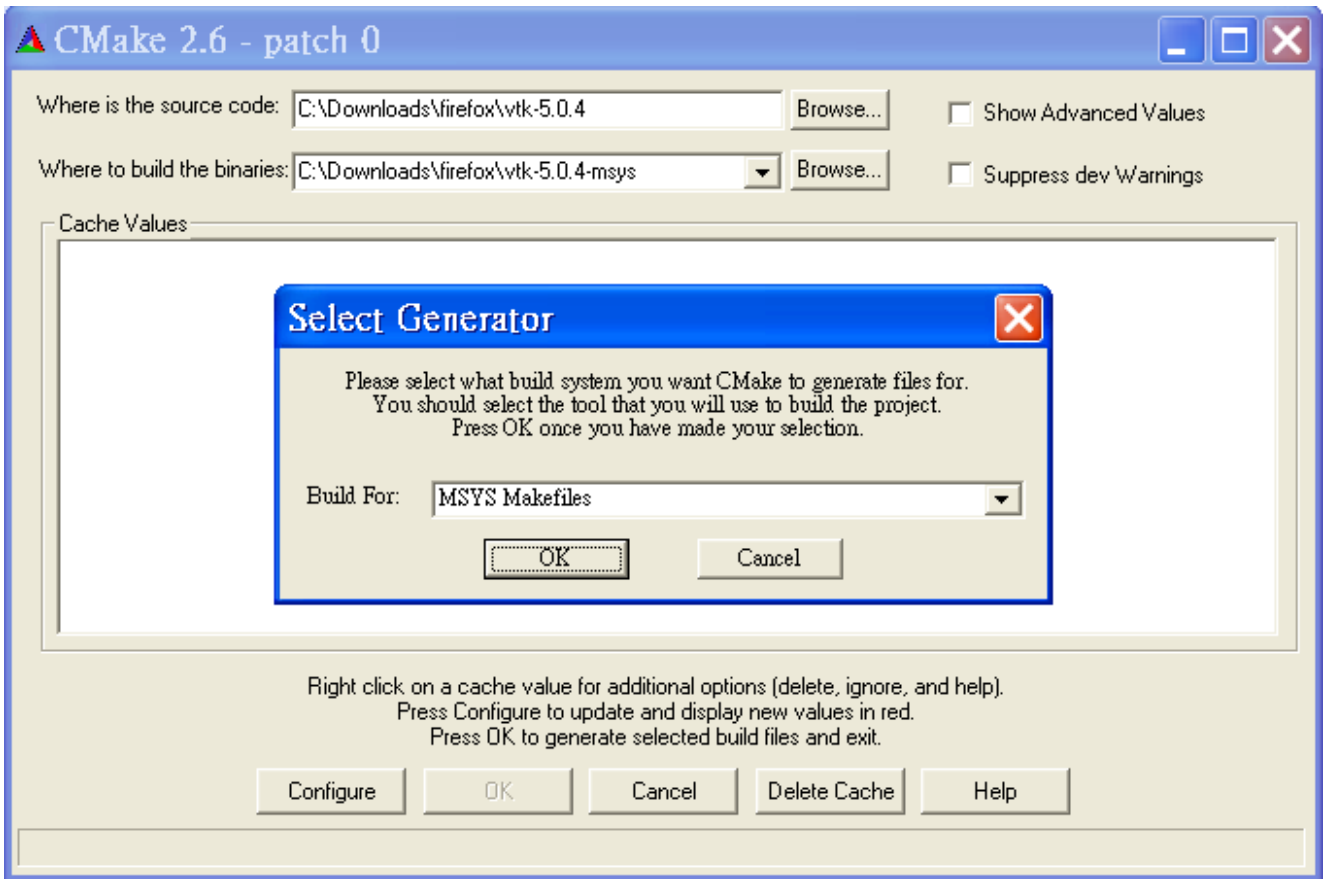

Figure 2: CMake, set the method to build VTK (We use MSys Makefiles)

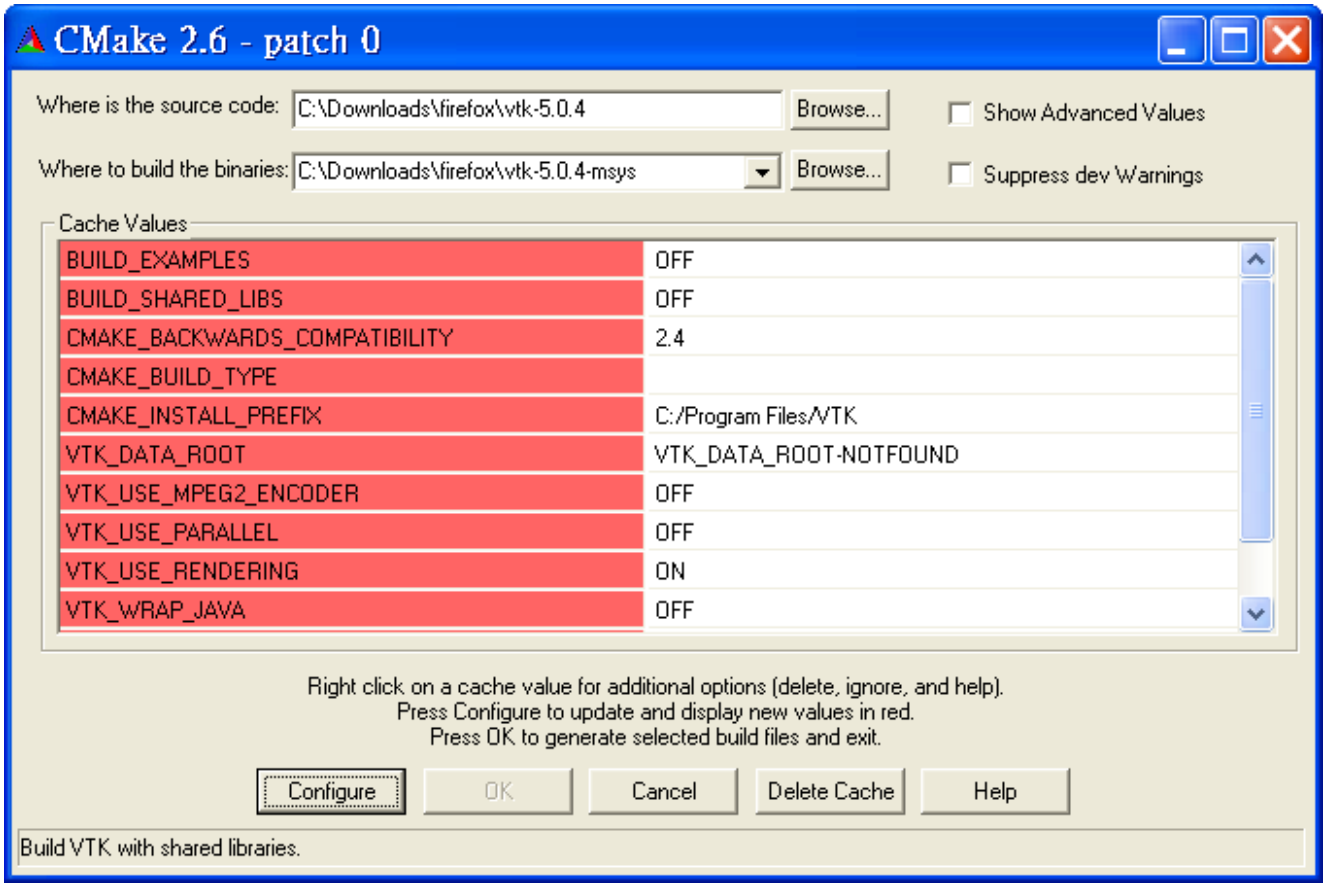

Figure 3: CMake, after the first time of configure

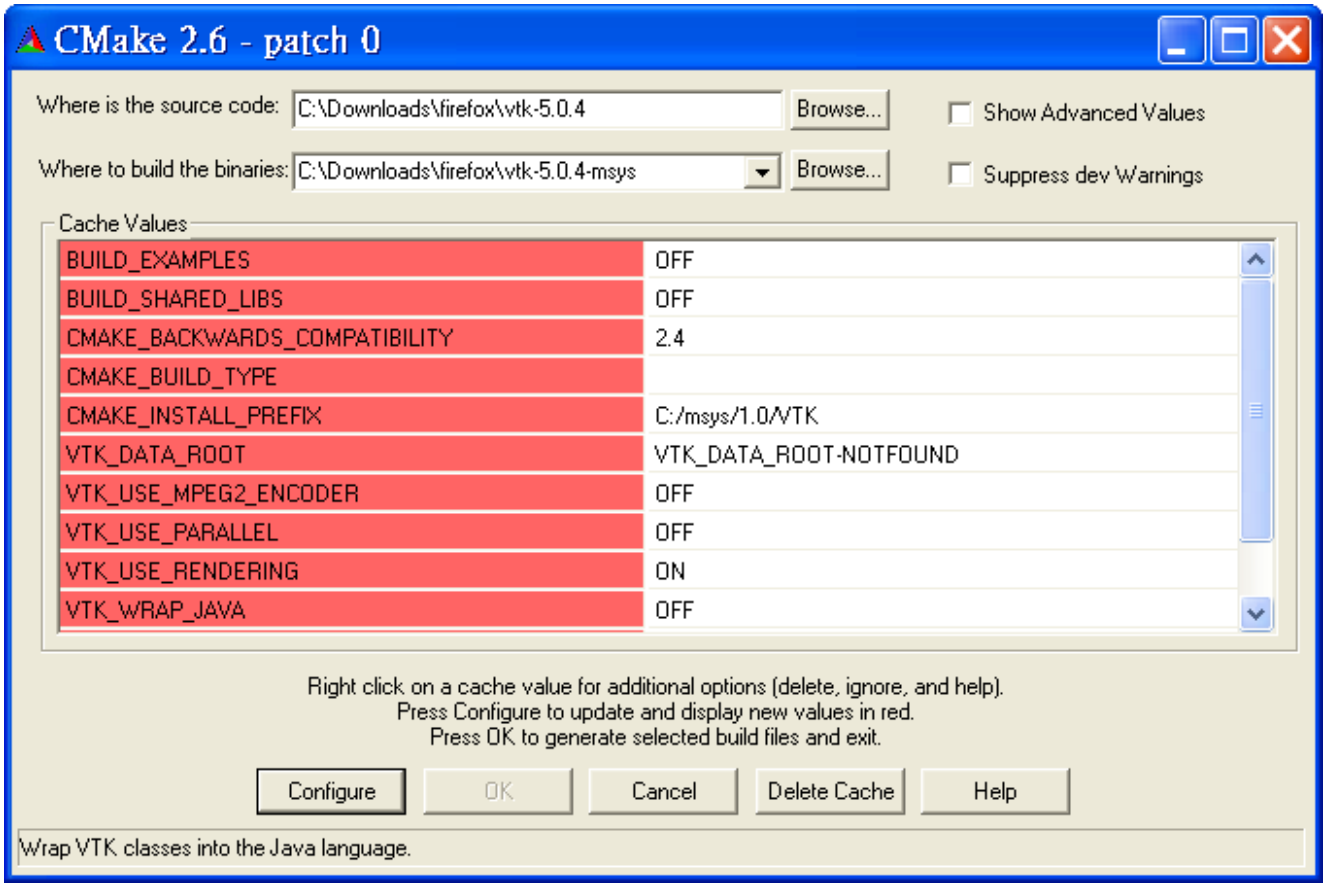

Figure 4: CMake, after editing a parameter in the list

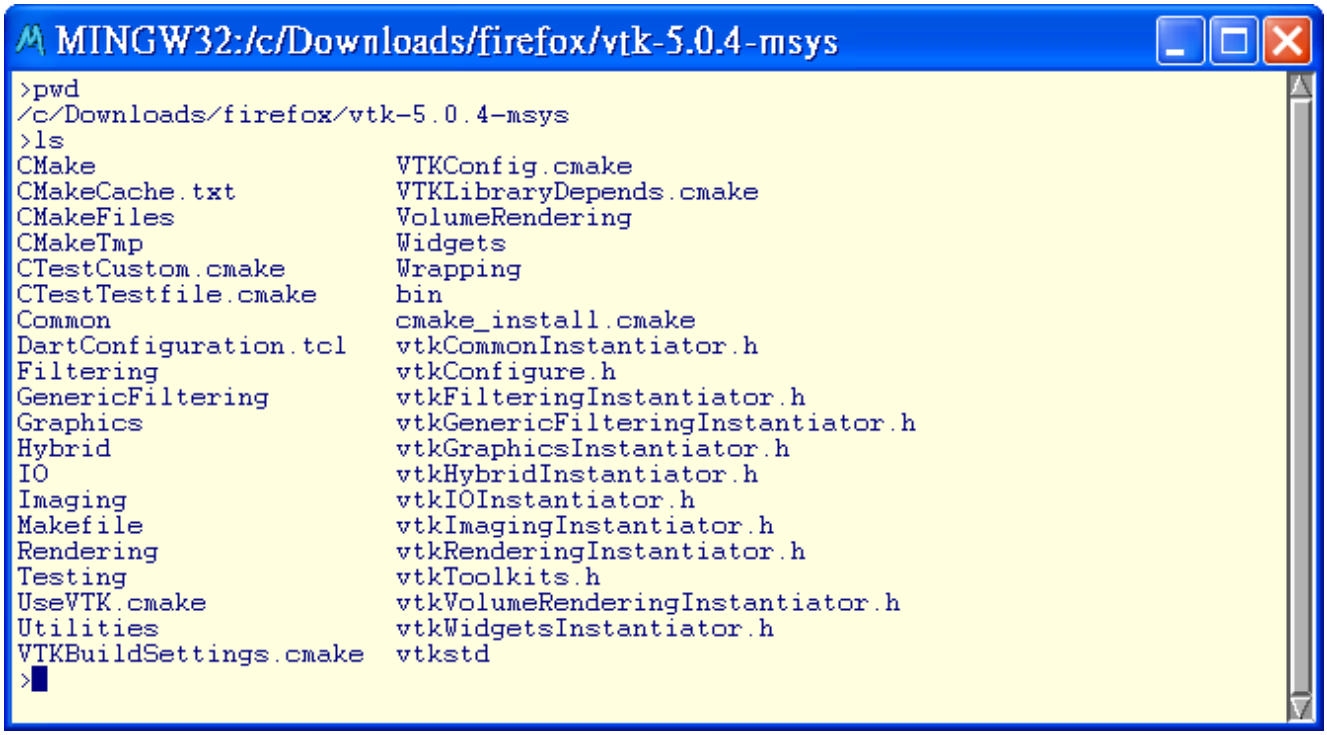

Figure 5: MSys, just before building VTK

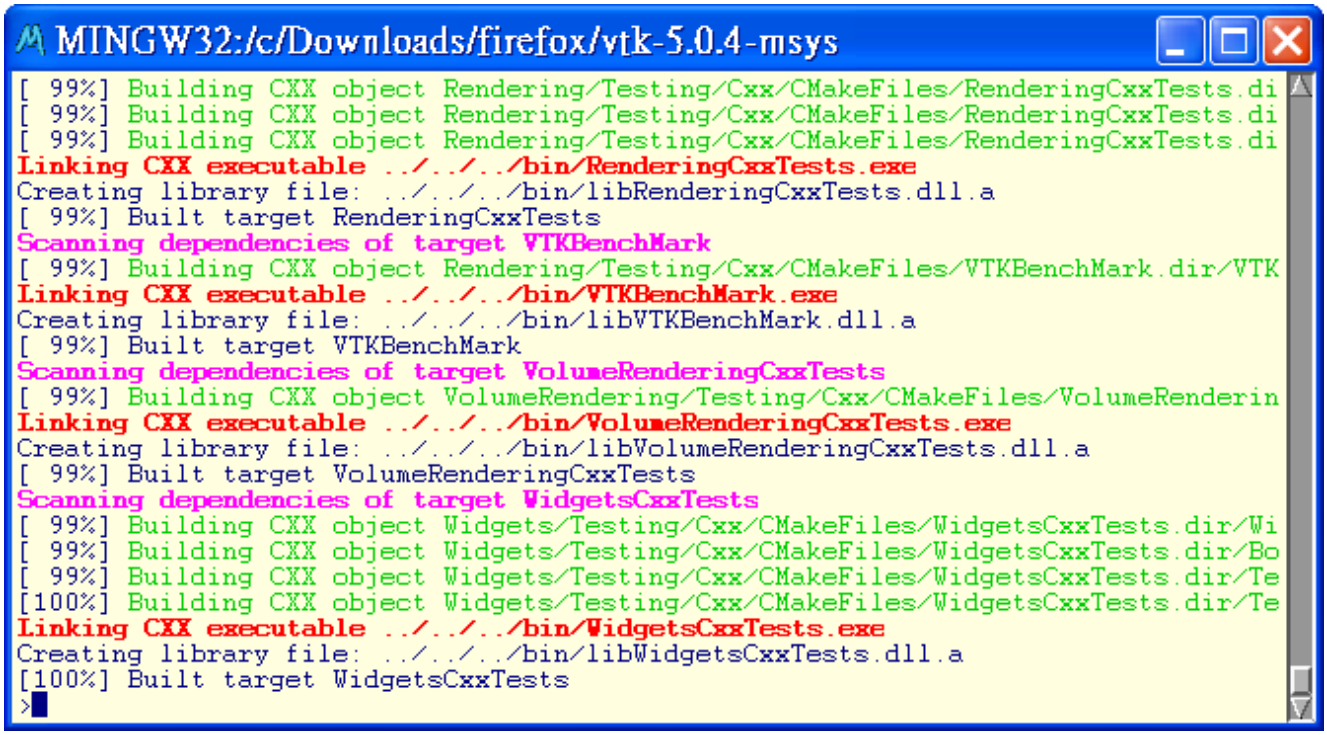

Figure 6: MSys, after building VTK. Now, you need to type ”make install” to install VTK

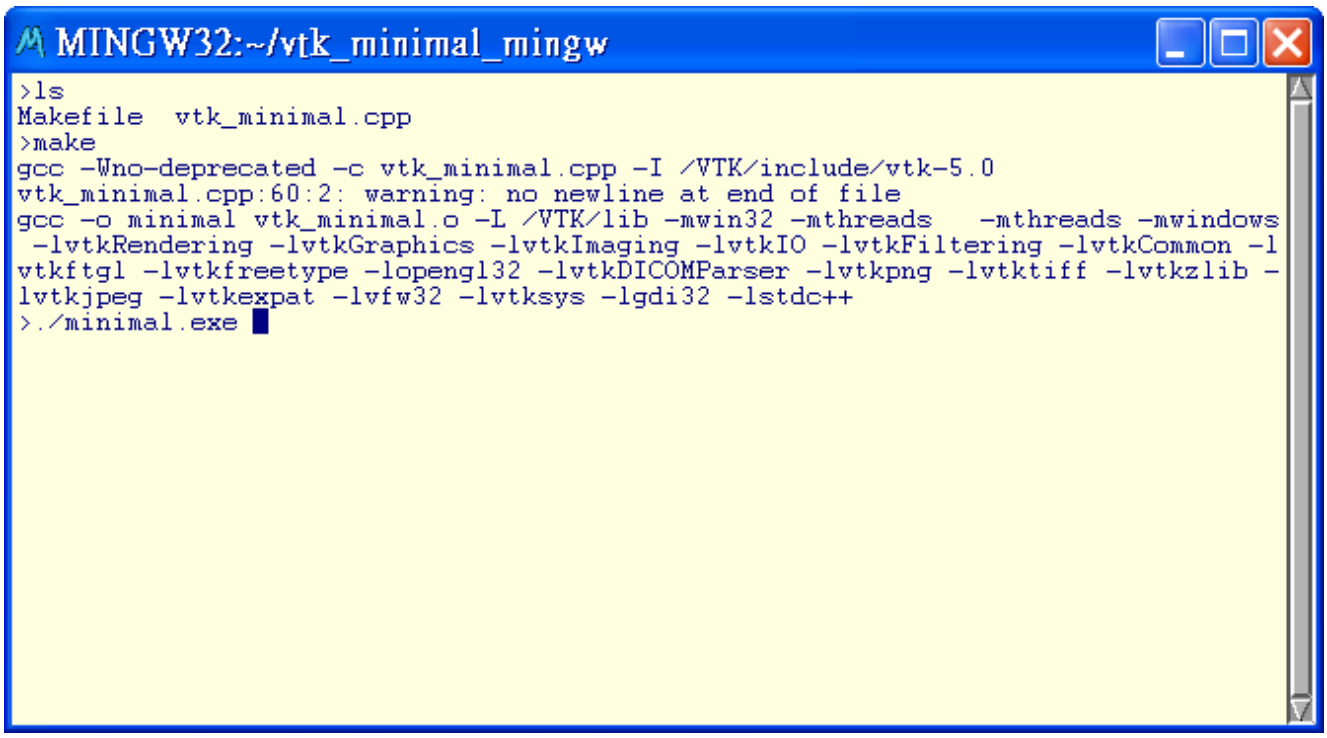

Figure 7: MSys, screen shot of building a minimal project in VTK using MinGW/MSys