

How to build VTK on Windows with Java support

Bartłomiej Wilkowski – Wilku – wilkowskib@gmail.com

February 16, 2007

Abstract

I have decided to write this document to ease up to the other users the process of (sometimes painful and frustrating) building the VTK on Windows with Java support. The date of creation of this document: October 2, 2006. Of course, this is only mine experience, so unfortunately I can't assure and give any warranty to anybody that all of my remarks given here are correct, but for sure, such configuration and such steps resulted that the VTK is working correctly with Java on my computer.

The details refer to my own experience with building VTK-5.0.2 source on Windows XP Service Pack 2, in order to use it with Sun's Java Development Kit 1.5.0_08 (JDK 1.5.0_08) support on Eclipse SDK 3.2 environment.

1 Required downloads and software installation

1.1 VTK source download

The VTK source we can download from the VTK official site [5] . Then we proceed by entering the Download sub page and downloading the latest release of VTK for Windows (in my case it was vtk-5.0.2.zip). BE CAREFUL!!! If you want to use VTK with Java support you must download **the source**, not Windows installer. We can enable the Java wrapping only during the manual compilation of VTK - with Windows Installer it's impossible.

It is very important. You must unpack your vtk-[version].zip file on the NTFS partition. With FAT32 I had some serious errors during compilation process

1.2 CMake download and install

To compile VTK I was using CMake software. CMake controls the software compilation process using simple platform and compiler independent configuration files. For Microsoft and Borland compilers there are pre-compiled binary. You can download the software from the CMake's webpage [3]. In my case, I got the most recent version (2.4).

Install CMake soft on the NTFS partition also (I'd recommend the same partition as the VTK). I suppose it could not work on FAT32.

1.3 C++ compiler installation

I used the Microsoft Visual Studio 2005 compiler. I know that it can be also Borland but I didn't try it.

1.4 Java SDK download and installation

You can download it from the Sun's webpage: <http://java.sun.com/> . It MUST be **Java Development Kit (JDK)** not Java Runtime Environment (JRE)!! Then there are various options to install it. You can enter the Sun's webpage, follow the link Java SE in the Download area and then look for the JDK (the most recent) and download it. Check it out there You can download also the source files in order to use them later in Eclipse.

1.5 Eclipse download and installation

You can download the software from the <http://www.eclipse.org>. I downloaded Eclipse-SDK-3.2, the most recent stable version. You can need also the EMF and GEF plugins as well as the Visual Editor (VE) for creating advanced GUIs. All of these you will find on the same webpage. The installation of Eclipse is very easy as you don't need to install anything You need to unpack the downloaded zip file, then copy some additional plugins and features to the respective folders

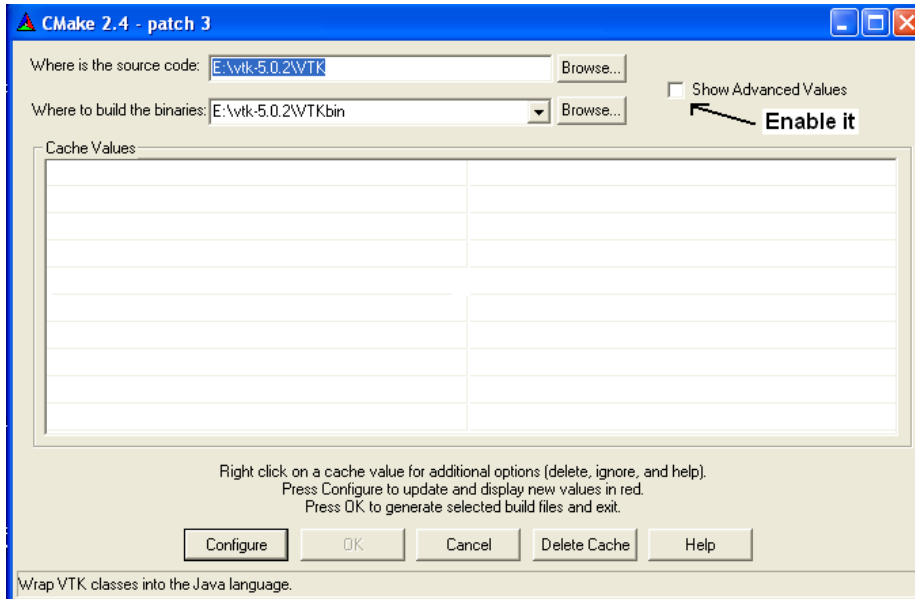


Figure 1: CMake window before configuration

in the Eclipse path and start the Eclipse with the Eclipse.exe file. For more info about the adding another features you can refer to the Eclipse webpage.

2 Compiling the VTK source with CMake

As we have the C++ compiler installed, CMake installed, the VTK source unpacked and Java SDK installed in the our system we are able to begin with the VTK source compiling. We start with executing the CMake executable file. There will appear window as shown in figure 1 on page 2.

Firstly, we must specify the path where the VTK source is placed (the our unpacked directory). In my case it was: E:\vtk-5.0.2\VTK. Then we must create the folder for the binaries of VTK. I have chosen the name for it as follows: VTKbin. Thus, the path was: E:\vtk-5.0.2\VTKbin. Enable the Show Advanced Values option - it is necessary for the later configuration. Finally, we press the Configure button, “praying” for the execution without any error - just joking, I am almost sure (if you do it on the NTFS partition) that there will be no errors. In the meantime, before the configuring, the CMake will ask you about the C++ compiler you are going to use later to build the configuration. In this case I selected in the combo-box the Visual Studio 2005 option. After accepting, the configuration starts and is going to take a while. Afterwards we will see the variables and values found in the CMake cache as depicted in figure 2 on page 3.

Now, we are customizing our build. In order to compile successfully you must

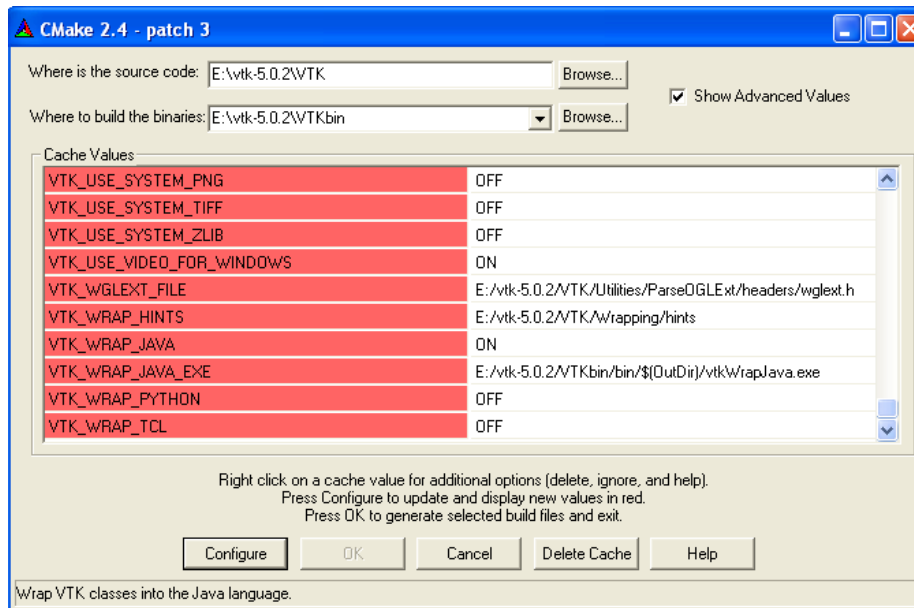


Figure 2: CMake window before customization

enable (change the value to ON) for the following options: VTK_WRAP_JAVA, BUILD_SHARED_LIBS, and VTK_USE_RENDERING. These are the options, which are almost always necessary. Another time, you can always open CMake and enable more options if you require.

After enabling and disabling all the options you consider as necessary, you must press Configure button one more time (or more times). You must do it until you reach that all the variables and values are not any longer in red. Then you can generate selected build files by pressing the OK button. This will cause CMake to write out the build files for the build selected. After that, the CMake will exit.

In case of any error appearance (at any moment), I recommend you not to carry on with the next step of this tutorial, but to repeat all the actions described here. It is easier to repeat it than to be disappointed afterwards (the next step takes a longer while).

3 Building the configuration in C++ compiler (Microsoft Visual Studio 2005)

Now, we must build the entire configuration in the C++ compiler. We do it in the following manner. We must find the location in Windows of our already created binaries. Open this folder in a normal Windows window. In my case the path was: E:\vtk-5.0.2\VTKbin. We are there looking for the project called

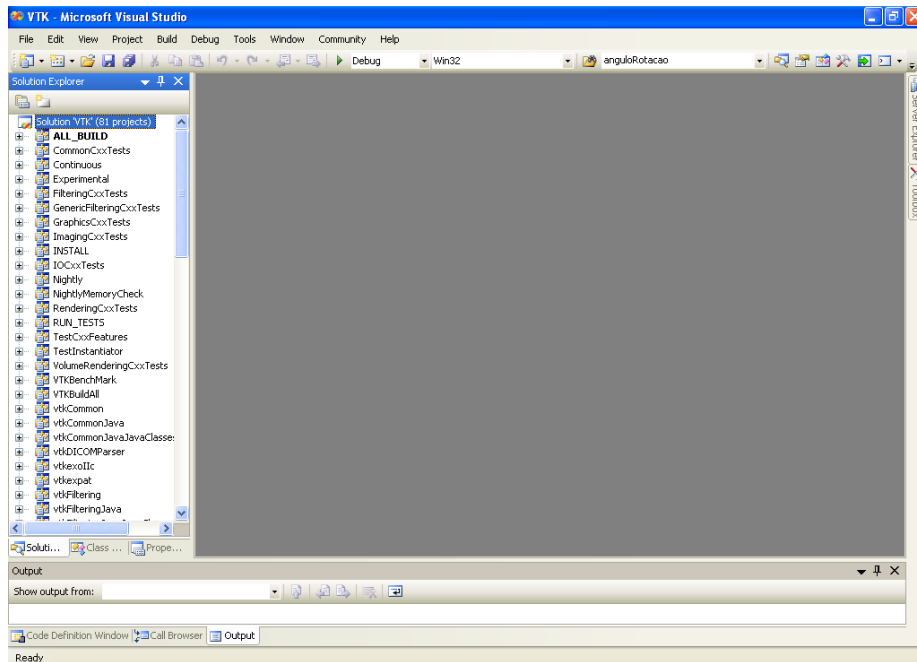


Figure 3: Visual Studio 2005 screenshot

ALL_BUILD. In my case it was the file ALL_BUILD.vcproj. Double-click then and you will simultaneously enter to your's earlier chosen C++ compiler environment (in my case - the Visual Studio 2005) as depicted on the figure 3 on page 4.

Single-click on the Solution VTK (look at the screenshot above) and then choose the option Build→Build Solution. We can define before the Active Configuration of the build. I used the default one (Debug) but there are also Release, MinsizeRel and RelWithDebInfo modes. The build is the long-time process. At the end of it we expect to not to have any error and afterwards usually all the libraries and executables are located in the VTK binary directory specified before (E:\vtk-5.0.2\VTKbin). Unfortunately, you will encounter some errors after the build. In my case there were problems with the build of the Java files. The compiler didn't compile any of my *.java files. In this apparently bad situation, it is the great information if you have only such the errors. It is very easy to repair it. You must only compile all the Java classes with any Java compiler. I did it with Eclipse, but you can do it even with the Windows Command Line. In the next point I will describe how I compiled it in Eclipse and how to start the first Java-VTK application.

The very important thing you must do yet is to edit your PATH environment variable. You turned on the option BUILD_SHARED_LIBS thus you must let Windows know where to find the DLLs. I used the Debug con-

figuration thus I added to the PATH variable the following path: E:\vtk-5.0.2\VTKbin\bin\debug. It is recommended that you add this entry in the beginning of the PATH definition.

Moreover, as the Java JDK and VTK has been installed, you need to set your CLASSPATH environment variable to include the VTK classes. You must include vtk\java directory. In my case the path was: E:\vtk-5.0.2\VTKbin\java\.

4 Configuration of Java environment in Eclipse

You can build the Java classes in very basic and manual manner. Firstly, you have to localize them. In my case I found them in the path: E:\vtk-5.0.2\VTKbin\java. Inside this folder there is another folder vtk which contains all the Java source classes (*.java). You only need to compile them to obtain the *.class files. The vtk folder is like the Java package. Thus, you must create a new project in Eclipse (File→ New→ Project→ Java Project). The name of the project is not important (in my case it was New) as you are going to use it only to compile the classes. After creating the new, empty project in Eclipse, add the source folder src. Then copy (do not cut it better) the whole folder vtk from the java folder (E:\vtk-5.0.2\VTKbin\java) to the newly created project in the Eclipse workspace (in my case the path was: D:\Program Files\eclipse-SDK-3.2\eclipse\workspace\New\src). Then come back to Eclipse environment, right-click on the New project and choose the Refresh option. Afterwards, you will have all your classes built to the *.class files in your Eclipse project path, in the folder bin. Now you must copy all the *.class files to the your vtk folder in the binaries VTK (E:\vtk-5.0.2\VTKbin\java\vtk). Now, you can create the *.zip archive, packing the vtk folder into it. It will contain both, source and binary files. In my case I created vtk.zip archive. See compact version of steps:

1. Unzip your vtk.zip package
2. Create in Eclipse empty project (any name)
3. Add to this project in Eclipse, the Source folder called src
4. Copy the unpacked vtk folder to the src folder
5. Click right on the project in Eclipse and press Refresh (Eclipse builds the files automatically to folder bin/vtk in Eclipse)
6. There can be one error in this package (for VTKJavaWrapped file) so delete it
7. Copy all the files from bin/vtk to src/vtk replacing all the files with the same name
8. Zip the vtk package from /src/vtk path to the file vtk.zip

9. Include this package to any VTK project and...

SUCCESS, it should work!

Now, you can run your first Java - VTK application. Let's say you want to run the Cone.java example given by the VTK. You can find it in the VTK folder (in my case: E:\vtk-5.0.2\VTK\Examples\Tutorial\Step1\Java). Therefore, create the new project in Eclipse called Cone. Then add the source folder as earlier and copy the Cone.java into it. Refresh the project under Eclipse. Now, you will need to include the vtk.zip archive containing all the VTK classes into the project. Just right-click on the project, choose Properties and then in Java Build Path press Add External Jar button. Find you vtk.zip archive and accept. Now you can compile and run your first Java-VTK application without any problem.

5 Summary

You can find more information about installing VTK on UNIX from [2]. Any Java with VTK examples you can find on [4]. Some addition to the previous ones: the 3D data viewer based on VTK and Java called CASSANDRA, look at [1].

I would be very grateful with any feedback and comments. Please do not hesitate to send me an e-mail (wilkowskib@gmail.com). With your suggestions this manual can be better and better. Send me also some useful links connected with VTK. Thanks in advance.

References

- [1] *3D data viewer based on VTK and Java (Cassandra)*. <http://dev.artenum.com/projects/cassandra>.
- [2] *Building VTK on Linux with Java Support*. http://www.duke.edu/~iwd/howto/VTK-Linux-Java_HOWTO.html.
- [3] *CMake web page*. www.cmake.org.
- [4] *ImageJ Plugins webpage - Java VTK Examples*. <http://ij-plugins.sourceforge.net/vtk-examples/index.html>.
- [5] *VTK web page and VTK User's Guide*. www.vtk.org.