VMD Installation Guide

A. Dalke, W. Humphrey, J. Ulrich

Version 1.2b1

March 24, 1997

Theoretical Biophysics Group[1]
University of Illinois and Beckman Institute
405 N. Mathews
Urbana, IL 61801
VMD WWW home page: `http://www.ks.uiuc.edu/Research/vmd/`

Description

This document describes how to install one of the precompiled releases of VMD and contains instructions on how to compile VMD from the source code release.

[1] http://www.ks.uiuc.edu/

1 Obtaining the VMD Source Code

You may obtain the VMD source from `ftp.ks.uiuc.edu` in `/pub/mdscope/vmd/` . For version 1.2b1 this is `vmd-1.2b1.all.tar.Z` or `vmd-1.2b1.all.tar.gz` . There may also be binary versions for other OSs here. The "all" distribution includes a pre-built binary for SGIs. Decompress and de-tar the file. This will produce a subdirectory named `vmd-1.2b1`. Unless otherwise specified, all references to VMD code will be from this subdirectory, so `cd` there.

1.1 Installing a pre-compiled version of VMD

If you simply want to install the pre-compiled version of VMD, then only three steps remain to be done.

- Edit the file `configure.parameters`. Change the value for the INSTALLLIBDIR to a directory in which vmd data files and executables should be installed, and change the following values:

```
   INSTALLBINDIR

This is the location of the startup script 'vmd'.  It should
be located in the path of users interested in running VMD.

   INSTALLLIBDIR

This is the location of all other VMD files.  This includes
the binary and helper scripts.  It should not be in the path.
```

- Next generate the proper Makefile based on these configuration variables. This is done by running `./configure` .
- After configuration is complete, `cd` to the `src` directory and type `make install`. This will put the code in the two directories listed above. After this, you just type `vmd` to begin.

1.2 Compiling Your Own Version of VMD

If for some reason you want to recompile VMD (i.e. you have made alterations to the source code etc.), then you will need to read the rest of this document.

2 Working with the Configure Script

2.1 Setting the Configuration Options

You must configure the compile time options you wish to use. These are set with the `./configure` script which produces Makefiles for the `src` and `$ARCH` files. After doing the configuration you will do the make. The configuration can be set in two ways:

- On the command line, as in:

```
./configure IRIX5 GL TCL OPTIMIZE FORMS
```

This will save the options to the file `./configure.options` so the next time you want to regenerate the Makefiles, you need but do

```
./configure
```

- Alternatively, you may edit the file `./configure.options` to the way you want it. (An advantage to keeping the information in a file is that we can set up several types of configurations and simply copy the one we want to use. For instance, for the SGI version we do something like:

  ```
  cp ./configure.options.IRIX5 ./configure.options
  ./configure
  ```

  for the HP version we simply do

  ```
  cp ./configure.options.HPUX9 ./configure.options
  ./configure
  ```

  However, these files are not in the distribution.)

If you are using g++, you must also change the following lines in configure from

```
LIBS          = -ll -lg-plus-plus $(OPTIONSLIBS) -lm
# For g++
# LIBS            = -lg-plus-plus $(OPTIONSLIBS) -lm
```

to

```
# LIBS          = -ll -lg-plus-plus $(OPTIONSLIBS) -lm
# For g++
LIBS            = -lg-plus-plus $(OPTIONSLIB) -lm
```

After you've set the options in the file, run `./configure` to propagate the new definitions to the Makefiles! The full list of compile-time options set by the configure script are:

- Which OS are you running? VMD can be compiled under several operating systems. The configure time options for these OSs are:

  ```
  Option:   Tested with
   IRIX5 -- SGI IRIX 5.3, SGI IRIX 6.x (see comments below)
   LINUX -- Linux 1.2.13 , Linux 2.0.27
   HPUX9 -- HPUX 9, HPUX 10
  ```

- Which type of display graphics will you use? There are four options which will be carefully explained below:

```
GL  -- SGI's GL graphics library
OPENGL -- OpenGL graphics library
MESA -- API implementation of OpenGL (uses only X calls)
NOGRAPHICS -- Do not use any display graphics
```

GL – We have managed to get the GL version running on IRIX5 (of course), HPUX9 (using the NPGL GL emulator from Portable Graphics) and AIX3 using IBM's implementation of GL. On older SGIs, OpenGL is emulated via GL, so you should probably compile for GL on these platforms.

OPENGL – We have managed to compile with OpenGL for SGIs. On the newer SGIs, GL is implemented via OpenGL, so you should compile for OpenGL on these platforms. If you have a mix of GL and OpenGL-based machines in the same environment and want to get the best performance out of each, you'll have to do a bit of hackery to start the appropriate binary. Our suggestion is to compile and install one version as GL, and compile the OpenGL and rename the binary vmd_IRIX5_ogl. You can place that binary in the VMDDIR and modify the 'vmd' startup script to recognize which machine is running and start the appropriate application. To get GL or OpenGL if you do not already have it/them, contact your vendor or other support services.

MESA – We have managed to compile with MESA 2.2 on all of the tested OSs. The Mesa library is available for free via ftp from `iris.ssec.wisc.edu` in the `pub/Mesa` directory. Mesa is also mirrored on sunsite in the directory `pub/packages/development/graphics/mesa`. See the web address
`http://www.ssec.wisc.edu/~brianp/Mesa.html` for more information, and the `README` in `./lib/Mesa` for information about how to get Mesa working with VMD.

NOGRAPHICS The NOGRAPHICS option could be used if you only wish to run VMD scripts, or compile VMD solely for use as a CGI client, or test VMD on a new, non-graphics platform, but with the `-dispdev none` command-line option, we can't imagine why you would want to worry about this.

BTW, reading of the configure script or code shows a STARBASE option. This is incomplete, does not work, and is not supported.

- VMD can be compiled to run in the CAVE. If you don't know what the CAVE is, see `http://evlweb.eecs.uic.edu/EVL/VR/` . To compile in the CAVE you will need to get the appropriate CAVE library. There are two, the GL (`libcave.a`) and OpenGL (`libcave_ogl.a`). VMD will compile in the correct one based on the type of graphics option you chose (GL or OPENGL), but so far will only compile with the GL version.

  To get access to the CAVE library, please contact them. We cannot distribute the library directly but the standard SGI binary distribution version is compiled with the CAVE library. To use that option, start VMD with the run-time option `-dispdev cave`.

  The configuration options are:

  ```
  CAVE -- include run-time support for CAVE display (GL or OpenGL).
  NOCAVE -- don't do it
  ```

- Which kind of GUI do you want to use? (The GUI is nearly independent from the graphics display method.) They are:

```
XFORMS -- the standard (soon to be "old") GUI
TK -- (not yet supported) the rewrite of the XFORMS interface in Tk
NOGUI -- don't compile in any GUI
```

  To date the GUI for VMD has been XForms or its GL-based predecessor, FORMS. In version 1.2b1 the GUI is still XForms based but we plan to convert to TK because of the increased flexibility available to scripts. For instance, this will allow user-scripts to create or modify the display on the fly; like drawing an RMSD plot in a new 2D graphics window. If you want to try out Tk, go ahead, but there will be no GUI or mouse control.

  You should compile with XForms, in which case you need to get the pre-built binaries from `http://bragg.phys.uwm.edu/xforms/` or from the ftp site `ftp://einstein.phys.uwm.edu/` . There are instructions in `./lib/xforms/README` describing what you need to do.

- There are options for conditionally compiling Tcl. New with version 1.2b1 and later versions you MUST compile with Tcl. (With a bit of work you should be able to get VMD working again without Tcl; the problem is that the VMD core commands now do some error reporting to the Tcl shell; but why worry about it?)

```
TCL     - required options
NOTCL   - obsolete and will cause compile-time errors
NOCAVE  - don't do it
```

  VMD was compiled against Tcl version 7.6p2 and TclX version 7.6 . You must have the 7.6 or better versions of both packages. We don't know if 8 will work. The latest versions can be found at:

```
Tcl 7.6 is at ftp://ftp.smli.com/pub/tcl
TclX 7.6 is at ftp://www.neosoft.com/languages/tcl/TclX
```

  Because Tcl and TclX are very useful programs, we make the assumption that these will be installed on your system, so there is no `./lib/tcl` directory. To configure VMD to use the appropriate Tcl/TclX directory, you may have to edit `./configure` and change a few variables. The current settings look in `/usr/local/[include,lib]` unless the corresponding environment variable is set. The variables are:

  From the configure script:

```
> TCL_INCLUDE_DIR=${TCL_INCLUDE_DIR-/usr/local/include}
> TCL_LIBRARY_DIR=${TCL_LIBRARY_DIR-/usr/local/lib}
>
> # location of TclX
> TCLX_INCLUDE_DIR=${TCLX_INCLUDE_DIR-/usr/local/include}
> TCLX_LIBRARY_DIR=${TCLX_LIBRARY_DIR-/usr/local/lib}
```

Either change the default values or override them with the environment variables. As a note, if compiling TclX without Tk, you'll need use the option `./configure --with-tk=NO`.

- Related to Tcl, VMD uses Tcl-DP to support socket-level communications with other programs. This is used internally to experiment with new types of user inputs (speech and gesture recognition) and to test ideas for multiple connected VMD sessions, as with classroom teaching or with networked collaborations. Because TclDP 4.0 is a run-time loadable package, this option will (should?) disappear with the final version of VMD 1.2. As it is now, the compile-time options are:

```
EXTERNAL  -- Use Tcl-DP to support connections with external programs
NOEXTERNAL  -- Don't
```

  As before, the configure script will use `/usr/local/{include,lib}` unless the standard Tcl-DP environment variables have been set. You may have to edit these lines if this is incorrect:

```
> # location of Tcl-DP (for EXTERNAL option)
> TCLDP_INCLUDE_DIR=${TCLDP_INCLUDE_DIR-/usr/local/include}
> TCLDP_LIBRARY_DIR=${TCLDP_LIBRARY_DIR-/usr/local/lib}
```

- VMD uses Amitabh Varshney's SURF program to compute the molecular surface. This is actually an external program that VMD calls; it is not compiled into VMD. You will need to compile SURF yourself to use option. The configuration options are:

```
SURF -- add the external call to SURF (uses DrawMolItemSurf.\{Ch\}
        and adds the option to the graphics pop-up list)
NOSURF -- don't use it.
```

  There is no reason not to compile SURF. It is actually distributed (with permission) as part of the VMD distribution. To compile surf, go to `./lib/surf` and follow the instructions in `README.VMD` (the `README` file if from surf itself). The end result will be a program named `surf_$ARCH` in that directory. Leave it there.

  For the main SURF distribution, see
  `ftp://ftp.cs.unc.edu/pub/projects/GRIP/SURF/surf.tar.Z`

- As with SURF, the alpha shapes code in VMD calls a set of external programs. As before, you can configure VMD to use/not use alpha shapes. Because VMD's use of the code is rather primitive, we suggest you use it only if you are interested in seeing what it looks like. Also, the interface that VMD uses to talk to the Alpha shapes code is not publically available (we are trying to get ahold of the author) so there isn't anyway you can even get the code. The configuration options are:

```
ALPHA -- Call the alpha shapes programs to compute the spectrum
NOALPHA -- don't
```

  You can get the alpha shapes code yourself, but this does not include the library VMD uses to read the output of the alpha shape binaries. If you want them, see `http://alpha.ncsa.uiuc.edu/alpha` or the ftp site
  `ftp.ncsa.uiuc.edu/Visualization/Alpha-shape` .

- In addition to the CAVE library, VMD can use the UNC Tracker library to get information about various 3DOF and 6DOF spatial trackers. We use it internally to support our Polhemus Fastrack and it also supports Ascension Flock-of-Birds and the BOOM. However, because the CAVE library has more people working on it, we suggest your future work depends more on that than the UNC code. The configurations options for using the UNC tracker library are:

  ```
  UNC -- use it
  NOUNC -- don't use it.
  ```

  The standard VMD distribution for IRIX5 is compiled with UNC. We have not tested it with any other platforms. See `./lib/unc/` for more information about the library and how to set it up with VMD.

- We are working on interactive simulations whereby VMD can control a running simulations. The communications uses MDComm to connect to NAMD. For more information about this, see the MDComm web page at
  `http://www.ks.uiuc.edu/Research/mdcomm/` .

  ```
  The configuration options for MDComm are:
     REMOTE -- use MDComm for doing remote simulations
     NOREMOTE -- don't use it.
  ```

  Please read `./lib/mdcomm/README` for information about setting up MDComm (and RAPP) for use in VMD.

- There are several compilation flags for VMD. They are:

  ```
  DEBUG or NODEBUG
  ```

  When turned on, these print all "msgdebug" messages. There are a lot of them and they really slow things down. (BTW, you can tell who wrote which sections of code by who uses msgDebug :) Also, set the -g flag when compiling. By default, VMD is compiled with NODEBUG. If you only want to add the -g flag, edit the configure script so the compiler is defined as "CC -g" (or "g++ -g").

  ```
  OPTIMIZE or NOOPTIMIZE
  ```

  Compile (or do not compile) with the -O flag. The flag is set via the ARCH_OPT_FLAG option in the configure file.

  ```
  MAKE or NOMAKE
  ```

  Automatically check-out and remake the code after doing a (re-)configuration. By default this is turned off. I'm not even sure what it will do if you don't have your code RCS'ed. The equivalent by hand is:

```
run "./configure <list of options here>"
cd src
make co.src co.h            (only if you are using RCS!)
make depend                 (only if you have g++!)
cd ../$ARCH                 (eg, "cd ../IRIX5")
make                        (in a multiprocessor machine, "make -P"!)
```

```
SILENT or NOSILENT
```

The default is SILENT, in which case you don't see the details of the compilation as it suppresses the make command echo. All this does is add or not add the line ".SILENT" near the top of the produced Makefile.

3 Building Libraries for VMD

Now, a detour. There are few libraries you will have to build. These are part of the standard VMD distribution but not part of VMD proper. They are:

3.1 libg++

These are modified versions of the GNU String and Regex classes. Their names are somewhat altered so there isn't a conflict with the original g++ distribution. To compile libg++ for a given architecture:

```
cd ./lib/libg++
-- Edit the Makefile and set the appropriate values for CC and CXX
make clean
make configure
make
-- if needed: mkdir lib_$ARCH  (e.g.: mkdir lib_IRIX5)
mv libg-plus-plus.a lib_$ARCH
-- for example: cp libg-plus-plus.a lib_LINUX
```

3.2 STRIDE

STRIDE is used by VMD to compute the secondary structure given the protein 3D coordinates. The appropriate STRIDE binary is included in the VMD binary distribution. To compile it yourself, see the web site at
`http://www.embl-heidelberg.de/stride/stride_info.html` for information on how to get the source and see `./lib/stride` for information on how to use it with VMD.

Change line 43 of `stride.h` from

```
#define MAX_AT_IN_RES           50
```

to

```
#define MAX_AT_IN_RES           75
```

because there are many structures with non-standard residues containing more than 50 atoms. Change line 96 of `stride.c` from

```
return(SUCCESS);
```

to

```
return(0);
```

since a program should return 0 if everything ended correctly.

3.3 Babel

Babel is used to convert different file formats automatically into PDB files(s) before reading them into VMD. Babel is a freely available general purpose molecular file conversion program. See the web site `http://mercury.aichem.arizona.edu/babel.html` for more information about it. This program must be available on the path somewhere or you can set the BABEL_BIN environment to point directly to the binary.

3.4 LASSP Tools

Some of the LASSP tools are used for computing matrix eigenvalues, determinates and inverses (needed for best-fit structure alignment). The code can be found at `http://www.lassp.cornell.edu/LASSPTools/LASSPTools.html`. See the README in `./lib/LASSP` for information on how to use the LASSPTools with VMD.

3.5 Renderers

VMD supports many types of rendering formats. You may want to get the packages associated with them. They are (among others) Raster3D, Radiance, and POV-Ray. Information about these and other programs can be found at:
`http://www.ks.uiuc.edu/Research/vmd/software`.

3.6 xmgr

The Labels form allows you to call an external program to graph a selected geometry value over time. The default program, which we suggest using, is xmgr, available from
`http://www.ccalmr.ogi.edu/CDF/xmgr.html`

4 Working with the VMD Source Code

Okay, so you've set the compile time options and run `./configure`, right? The next step would be to check out the latest versions of the files from RCS, but odds are you don't have VMD RCS'ed (or under any other revision control). If you were, you would do (INSIDE THE SRC DIRECTORY):

```
cd ./src
make co.src co.h
```

Note that the configure command also makes the subdirectory `$ARCH`.

Next you'll need to make the dependency list INSIDE THE SRC DIRECTORY. This is done with

```
  cd ./src
  make depend
```

However, the current implementation needs `gcc` to get the list of include files needed by the different source files. The default Makedata.depend files for the different platforms is available in the appropriate binary distribution. After the dependencies are generated, run `make` in the `$ARCH` directory

```
  cd ../$ARCH
(for example
  cd ../LINUX
  cd ../IRIX5
etc.)
```

This directory was made when you typed '`make co.src co.h`' above.

```
make
```

Special notes:

- If compiling with the vendor `lex` under IRIX, there will be many warnings for the file `AtomLexer.l` and `AtomParser.y` . This is fine.
- If linking with a version of Tcl that supports dynamic library loading, you may need the -ldl library (there may be an error about not finding `dlopen` or `dlsym`) . The most appropriate way to add the library is to edit the "configure" script (in .. if you are in the `$ARCH` subdirectory) and change the line:

  ```
  TCLLIB = -ltclx -ltcl
  to
  TCLLIB = -ltclx -ltcl -ldl
  ```

  After editing the file, run `./configure` again to propagate the Makefiles, `cd $ARCH`, and `make` .

- On some machines (depending on how things are set up) you might get one of the following two error message:

  ```
  > ld: cannot open -lXext: No such file or directory
  > CC: warning: Unable to find library for -lX11. (920)
  ```

  If this happens, run `find /usr -name 'libXext*' -print` . It should return something like:

  ```
  > /usr/X11R6/lib/libXext.a
  > /usr/X11R6/lib/libXext.so
  >  [...]
  ```

  or

```
> /usr/lib/X11R5/libXext.sl
> /usr/lib/X11R5/libXext.a
```

(libX11 should be in the same directory.) If this is so, change one the of the following lines in `configure`. (Uncomment one, comment out the wrong one):

```
MESA_LIBRARIES="-lMesaGL -lMesaGLU -lXext"
# If there is a failure for libXext.a, get the path using
#  find /usr -name libXext.a -print
# and use one of the following forms
## for HPUX you might need one of the following two
# MESA_LIBRARIES="-lMesaGL -lMesaGLU -L/usr/lib/X11R5 -lXext"
# MESA_LIBRARIES="-lMesaGL -lMesaGLU -L/usr/X11R5/lib -lXext"
## for LINUX you might need
# MESA_LIBRARIES="-lMesaGL -lMesaGLU -L/usr/X11R6/lib -lXext"
```

- If you get an error like

```
> ld: cannot open -ll: No such file or directory
```

(see section 2) change the following lines in configure from

```
LIBS          = -ll -lg-plus-plus $(OPTIONSLIBS) -lm
# For g++
# LIBS            = -lg-plus-plus $(OPTIONSLIBS) -lm
```

to

```
# LIBS          = -ll -lg-plus-plus $(OPTIONSLIBS) -lm
# For g++
LIBS            = -lg-plus-plus $(OPTIONSLIB) -lm
```