Viewmol

Version 2.4.1

by

Jörg-Rüdiger Hill

September 18, 2004

1 Purpose

VIEWMOL is a graphical front end for computational chemistry programs. It is able to graphically aid in the generation of molecular structures for computations and to visualize their results. The program's capabilities include:

- Building and editing of molecules
- Visualization of the geometry of a molecule
- Tracing of a geometry optimization or a MD trajectory
- Animation of normal vibrations or to show them as arrows
- Drawing of IR, Raman, and inelastic neutron scattering spectra
- Drawing of an MO energy level or density of states diagram
- Drawing of basis functions, molecular orbitals, and electron densities
- Display of forces acting on each atom in a certain configuration
- Display of Miller planes in crystals
- Calculation of thermodynamic properties for molecules and reactions
- Drawings generated by VIEWMOL can be saved as TIFF, PNG, HPGL, or PostScript files
- Animations of normal modes can be converted to a video file (MPEG), e. g. for inclusion into World Wide Web documents (requires additional programs available on the Internet)
- Interface to the ray tracing program POVRAY (input file generation and use of POVRAY from within VIEWMOL)
- Input and output in a variety of formats, new formats can be added easily by the user

VIEWMOL includes a Python interpreter for automation.

At present VIEWMOL includes input filters for DISCOVER, DMOL^3, GAMESS, GAUSSIAN 9X/03, GULP, MOPAC, PQS, TURBOMOLE, and VAMP outputs as well as for PDB files (VIEWMOL is therefore suited as a viewer for structural data on the World Wide Web). Structures can be saved as Accelrys' car-files, MDL files, and TURBOMOLE coordinate files. VIEWMOL can generate input files for GAUSSIAN 9X. VIEWMOL's file format has been added to OPENBABEL so that OPENBABEL can serve as an input as well as an output filter for coordinates.

VIEWMOL supports a space ball as input device.

2 Copyright, Bug Reports, Feature Requests

VIEWMOL is copyright ©1996-2004 by Jörg-Rüdiger Hill and others. All rights reserved. VIEWMOL is licensed under the GNU General Public License, version 2. A copy of this license can be found in

VIEWMOL shall be quoted as follows:

Program VIEWMOL, Version 2.4.1, Jörg-Rüdiger Hill, 2004.

VIEWMOL has been thoroughly tested on the computer systems mentioned below, but it cannot be excluded that there are any further bugs.

Please, report bugs at:
`https://sourceforge.net/tracker/?func=add&group_id=6845&atid=106845`
or to: joehill@users.sourceforge.net
Feature requests should be submitted to:
`https://sourceforge.net/tracker/?func=add&group_id=6845&atid=356845`
Support requests should go to:
`https://sourceforge.net/tracker/?func=add&group_id=6845&atid=206845`

There is also a discussion forum for VIEWMOL at
`https://sourceforge.net/forum/forum.php?forum_id=21242`.

3 Installation

VIEWMOL 2.4.1 was developed on an Athlon PC with an Nvidia graphics card running Red Hat Linux 9. Hardware acceleration for 3D is not strictly necessary, but recommended. VIEWMOL also works with

Brian Paul's OpenGL compatible library Mesa. VIEWMOL has been ported to PCs with FreeBSD, Silicon Graphics computers, IBM RS/6000, DEC Alpha, Suns, Hewlett Packard 9000/735, Macintoshs with MacOS X, and Windows PCs using the Cygwin tools. Since Mesa runs on any machine which has the X Window System it should be possible to run VIEWMOL on any machine which supports the X Window System. However, for best performance a native OpenGL implementation is recommended.

Binaries are presently supported on the following operating systems:

- PC: Linux kernel 2.4.x and 2.6.x, FreeBSD 5.1
- PC: Microsoft Windows using Cygwin (thanks go to Noda Tomoyuki, noda.tomoyuki@canon.co.jp, for the port)
- Macintosh: MacOS X (courtesy of Gert von Helden, helden@fhi-berlin.mpg.de)

On these operating systems the program was tested. Others may also work, but this is not certified. Previous versions of VIEWMOL have also been tested on the following systems:

- DEC Alpha: OSF1 V4.0 (tested only with Mesa; courtesy of Pablo Vitoria Garcia, qibvigap@lg.ehu.es)
- IBM: AIX 4.1
- HP: HP–UX 9.5 (tested only with Mesa),
- SGI: Irix 6.3
- Sun: SunOS (courtesy of Keith Refson, Keith.Refson@earth.ox.ac.uk)

but since the author does not have access to any of these systems testing of newer versions was not possible. It can, however, be assumed that VIEWMOL will at least compile on these systems.

VIEWMOL 2.4.1 is provided precompiled for a number of architectures. Precompiled binaries are packaged separately from source, documentation, and examples:

- x86 Linux `viewmol-2.4.1.bin.Linux.tgz`
- x86 FreeBSD `viewmol-2.4.1.bin.FreeBSD.tgz`
- Macintosh `viewmol-2.4.1.bin.MacOSX.tgz`

VIEWMOL can be downloaded from:

- SourceForge
  `http://viewmol.sourceforge.net/`

Before installing VIEWMOL you need to make sure that the following software/libraries are available:

- OpenGL or Mesa (libGL.so, libGLU.so or libMesaGL.so, libMesaGLU.so)
  This is probably only relevant for installations on Linux. Most Linux distributions now include support for OpenGL programs. Mesa can be found at `http://sourceforge.net/projects/mesa3d`.

- X Window System
  This is only relevant for installations on Microsoft Windows and Macintoshs. For Microsoft Windows Cygwin needs to be installed which includes the X Window System `http://www.cygwin.com/`. Apple distributes the X Window System for Mac OS X at `http://www.apple.com/macosx/`.
- Motif or Lesstif (libXm.so)
  This is probably only relevant for installations on Linux, Microsoft Windows, and Macintoshs. Motif can be found at `http://www.motifzone.org/download/`. Lesstif can be found at `http://www.lesstif.org/`. For Microsoft Windows Cygwin includes Lesstif. For Mac OS X the Fink distribution `http://fink.sourceforge.net/` includes Lesstif.
- Python >=2.2
  Linux distributions usually include Python. For other systems you need to obtain Python from `http://www.python.org/`
- TIFF library (libtiff.so)
  Linux distributions include this library. On Silicon Graphics this library is installed if the `ifl_eoe` and `ifl_dev` packages have been installed. For other systems you need to obtain and compile this library (e. g. from `ftp://ftp.remotesensing.org/pub/libtiff`)
- PNG library (libpng.so)
  Linux distributions include this library. For other systems you need to obtain and compile this library (from Sourceforge under `http://libpng.sourceforge.net/`)

Installation of the program is simple. VIEWMOL comes as gzipped tar file, `viewmol-2.4.1.src.tgz`. Unzip and untar it using `gunzip viewmol-2.4.1.src.tgz` and `tar -xvf viewmol-2.4.1.src.tar`. You get six subdirectories, `source`, `doc`, `scripts`, `tests`, `locale`, and `examples`, and the configuration file `viewmolrc`. Copy all files you got into an arbitrary directory. If you want to install precompiled binaries, download the appropriate file for your operating system and unpack it from the same directory you unpacked the source code. This will create a subdirectory in the source directory which holds the binaries (the name of this directory starts with the name of your operating system as you get it from `uname -s` and may contain a CPU specific ending). If you run the supported operating systems you have to set the environment variable `$VIEWMOLPATH` to point to the directory where you unpacked VIEWMOL (the compiled-in default for `VIEWMOLPATH` is `/usr/local/lib/viewmol`) and the installation is complete. Otherwise you have to recompile the program (cf. p. 5). The program uses dynamical memory allocation so that every size of a molecule can be handled which fits the hardware limits of your computer.

The installation directory also contains a file `viewmolrc`. You might have a look into this file and adapt it to your needs. The format is described at page 45. In general the defaults should work fine.

VIEWMOL uses by default English as language, but it has been written so that other languages can easily be used[1]. The distribution contains localized versions of all the program messages, menus, dialog boxes etc. in the directory `locale` (currently for English, French, German, Hungarian, Polish, Russian, Spanish, and [partly] Turkish). If you want to use a different language for a system wide installation, copy the corresponding `Viewmol` file to your applications default directory (usually `/usr/lib/X11/app-defaults`). If you want to use a different language only for some users, instruct them to configure the language through

[1]This manual assumes that the English version of VIEWMOL is used. The shortcuts for other languages are different, but obvious in the menus.

VIEWMOL's `Configuration` menu. VIEWMOL will run without any of the localized files installed. So if you are happy with English and want to change only a few settings you do not need to install any of the files from the `locale` directory.

VIEWMOL needs a few external programs for some of its functions. Once you have installed VIEWMOL and set `VIEWMOLPATH`, you can start VIEWMOL, press `Cancel` in the file selection box which will appear, and press the right mouse button in the blue VIEWMOL window. A popup menu will appear where the last but one option is `Configuration ....` Choosing this option displays a dialog where you can set path names to four external programs. These are (including their defaults)[2]:

```
Location of Web browser:                     mozilla %s
Location of Moloch:                          moloch
Location of Rayshade:                        x-povray +I%s +O%s +W%d +H%d
Location of display program for images:      xv %s
```

If these program are installed and can be found in your path VIEWMOL will automatically display the correct path names in the dialog. The `%s` and `%d` are placeholders for the file name and dimensions and are required for programs which use command line arguments. Once you have set these path names, choose `Save` from the buttons in the dialog and these settings will be stored permanently in `$HOME/.Xdefaults`.

The correct installation of VIEWMOL can be tested through the included test scripts (cf. p. 55).

4 Compilation

VIEWMOL 2.4.1 has been written in C. For compilation of VIEWMOL you need a C compiler. TIFF files are supported by the freely available TIFF library which is also necessary to compile the program. It can be found on many ftp sites, e. g. at `ftp://ftp.remotesensing.org/pub/libtiff/`. PNG files are supported by the freely available PNG library which is also necessary to compile the program. It can be found at `http://libpng.sourceforge.net/`. If you want to link VIEWMOL with Mesa instead of with OpenGL you will need Mesa (`http://www.mesa3d.org/`).

Linux users need Motif to compile and run the program (if the program complains about "`viewmol: can't load library 'libXm.so.1'`" Motif is missing). Motif is available from `http://www.motifzone.org/download/`. The Motif clone Lesstif (`http://www.lesstif.org/`) can be used with Viewmol starting with version 0.81. There are, however, some glitches with Lesstif (e. g. shortcuts don't work).

If you want to recompile the program and you are running one of the supported operating systems you may type `make`. The shell script `getmachine` determines the operating system you are running and sets some options for the compiler. If this does not work you should have a look into the `Makefile`. The options set are explained there. They are the following:

- `OPT`
  The optimization flag for your compiler (on Linux `-O6 -mcpu=X` where `X` may be `pentium` or `pentiumpro`; `-O2` otherwise).

[2]Moloch may be called TurboPROP if you got TURBOMOLE from Accelrys Inc.

- `CFLAGS`
  Additional flags for C compiler, used to handle special optimizations and some incompatibilities between different Unix versions.

- `LDFLAGS`
  Additional flags needed for linker, currently only used for SGI to distinguish between different library versions.

- `INCLUDE`
  The path to the include files. This is set to point to the include files for libtiff, libpng, and Python (the script asks the user for these paths at the beginning, there is really no need to change anything in the file here).

- `LIBRARY`
  The path to the additional libraries required. These are libtiff, libpng, and the Python library (the script asks the user for these paths at the beginning, there is really no need to change anything in the file here).

- `LIBS`
  The libraries needed to link VIEWMOL. They may differ between different operating systems.

The `getmachine` shell script will ask you for the path names to the TIFF, PNG and Python libraries and to the include files necessary with these libraries. You may specify these path names using environment variables if you put the name of the variable in parentheses (e. g. `$(HOME)`). These path names are assigned to the `LIBTIFF`, `TIFFINCLUDE`, `LIBPNG`, `PNGINCLUDE`, `PYTHONINCLUDE`, and `LIBPYTHON` flags and stored in a file `.config.<OS>` where `<OS>` is the output of the `uname -s` command on your machine. If this file already exists, `getmachine` does not ask for these path names.

Silicon Graphics compilers on 64-bit operating systems (IRIX64 – R8000, R10000, R12000) will produce a lot of warning messages concerning casts of pointers to integers. These can be safely ignored.

The make procedure will build the program in a directory whose name depends on the operating system and type of CPU you are using. You will find all executables in this directory. After compilation follow the steps under Installation to complete the installation. If you have compiled VIEWMOL yourself it is recommended that you use the included test scripts to ensure VIEWMOL is functioning correctly (cf. p. 55).

5 Synopsis

VIEWMOL can be called as follows:

```
viewmol [[-bio | -dmol | -gamess | -gauss | -gulp | -mopac |
          -pdb | -pqs | -tm | -tmmsi | -vamp] file]
```

VIEWMOL has an automatic file format detection algorithm build in and should be able to identify output files of the programs supported without user intervention. VIEWMOL will also run Python scripts given on the command line when the Python script has the string `python` within the first 1024 characters (this is usually the case if the script starts with the common `#!/usr/local/bin/python` or similar first line). If VIEWMOL is called without parameters it will bring up a file selection box to select the file to be viewed.

If the option `-bio` is used the file name of a DISCOVER file has to be specified. One can use the `.car`, the `.cor`, or the `.arc` file of DISCOVER. VIEWMOL also looks for a file with the extension `.hessian` and tries to read the vibrational spectrum from it, if it was found.

If the option `-dmol` is used the file name of a DMOL/DSOLID/DMOL^3 output file has to be specified.

If the option `-gamess` is used the file name of a GAMESS output file has to be specified.

If the option `-gauss` is used the file name of a GAUSSIAN 9X output file has to be specified.

If the option `-gulp` is used the file name of a GULP output file has to be specified.

If the option `-mopac` is used the file name of a MOPAC output file has to be specified. VIEWMOL also looks for a file with the extension `.gpt` and reads information about basis sets and MOs from it, if it was found.

If the option `-pdb` is used the file name of a PDB file has to be specified.

If the option `-pqs` is used the file name of a PQS output file has to be specified.

If the options `-tm` or `-tmmsi` are used the file name of a TURBOMOLE file containing at least the data group `$coord` has to be specified. `-tm` reads the original TURBOMOLE output while `-tmmsi` allows VIEWMOL to read TURBOMOLE outputs from the TURBOMOLE version distributed by Accelrys (the only difference is the ordering of the normal modes in the `control` file).

If the option `-vamp` is used the file name of a Vamp output file has to be specified. VIEWMOL also looks for a file with the extension `.gpt` and reads information about basis sets and MOs from it, if it was found.

6 Usage and Operating Modes

6.1 Data Read From Input Files

- DISCOVER
  The file names for DISCOVER files can be `file_name.car`, `file_name.cor`, or `file_name.arc`. The base name is used to construct the file name `file_name.hessian` (the file with frequencies and normal coordinates). All necessary data are extracted from these files.

- DMOL/DSOLID/DMOL^3
  The necessary data are collected from the `.outmol` file.

- GAMESS
  GAMESS output files are first checked for the occurrence of the string `GAMESS`. If it is found the necessary data are collected from this file.

- GAUSSIAN 9X
  Gaussian output files are first checked for the occurrence of the string `Entering Gaussian System`. If it is found the necessary data are collected from this file. To use the wave function related topics in VIEWMOL with GAUSSIAN outputs GAUSSIAN must run with `GFPRINT` and `Iop(5/33=1)`[3] to print basis set and MO coefficients. Due to the vastly different outputs created by the GAUSSIAN 9X series of programs, it is not guaranteed that a particular output

[3]GAUSSIAN 98 seems to have a bug with respect to this option – no MO coefficients are printed anymore. Use Iop(5/33=2) instead which, unfortunately, also prints the density matrix.

can be successfully read. The common types of output have been tested, but non-default routes through the program might have generated output which cannot be read.

- MOPAC/VAMP
  VIEWMOL first checks for the presence of a file with the extension `.gpt` and the same basename as the MOPAC output file. This file is generated if MOPAC has been run with the keyword `GRAPH`. If such a file is found coordinates, basis functions, and MO coefficients are read from this file. If such a file does not exist, coordinates are read from the MOPAC output file under the header `CARTESIAN COORDINATES`. Finally, vibrational frequencies and normal modes are read from the MOPAC output file, if present.

- PDB files
  Only the cartesian coordinates and atomic symbols are read from this file, the connectivity information is ignored and will be determined by VIEWMOL itself.

- PQS
  Coordinates are read from the section identified by `Coordinates (Angstroms)`. Forces, the energy, vibrational frequencies, and normal modes are collected from the corresponding sections of the output.

- TURBOMOLE
  The program reads the following data groups from the `control` file:

  - `$atoms`
  - `$basis`
  - `$pople`
    The basis functions are read from these data groups. These data will be read only if they are available.
  - `$closed shells`, `$alpha shells`, `$beta shells`
    These data group are read to determine which molecular orbitals are occupied by how many electrons. The data is necessary for the calculation of electron densities.
  - `$coord`
    The cartesian coordinates of the molecule calculated. This data group must be available.
  - `$grad`
    The cartesian coordinates and gradients of all previous steps of a geometry optimization. This data group will be read only if it is available.
  - `$scfmo`
    The symmetry labels, energies, and MO coefficients for closed shells are read from this data group. These data will be read if they are available and if the file contains either converged or first order molecular orbitals.
  - `$uhfmo_alpha`
  - `$uhfmo_beta`
    The symmetry labels, energies, and MO coefficients for open shells are read from this data group. These data will be read if they are available and if the file contains either converged or first order molecular orbitals.
  - `$symmetry`
    The point group of the molecule. This data group will be read only if it is available.

- `$title`
  The title of the calculation. This data group will be read only if it is available.
- `$vibrational spectrum`
- `$vibrational normal modes`
  The results of a force constant calculation. These two data groups will be read only if they are available.

6.2 The Main Window

The program displays the molecule according to the coordinates in the main window. Following manipulations are possible (cf. Figure 1):

- Holding down the left mouse button and moving the mouse horizontally
  This rotates the molecule, the view point, or a light source around the y axis.
- Holding down the left mouse button and moving the mouse vertically
  This rotates the molecule, the view point, or a light source around the x axis.
- Holding down the middle mouse button and moving the mouse horizontally
  This rotates the molecule, the view point, or a light source around the z axis.
- Holding down one of the shift keys and the left mouse button and moving the mouse
  This moves (translates) the molecule or an annotation.
- Holding down one of the shift keys and the middle button and moving the mouse
  The scaling of the molecule is changed.
- Pressing the cursor keys for moving up <↑> or down <↓>
  The scaling of the molecule is changed. By pressing <↑> the molecule will be enlarged and by pressing <↓> the molecule will be made smaller.
- Clicking on an atom with the left mouse button
  This selects this atom, you will hear a beep. If you have clicked on one atom and then pressed the right mouse button, the average of all bond lengths at this atom is displayed. If you have clicked on two atoms and then pressed the right mouse button, the distance between these two atoms is displayed. If you have clicked on three atoms and then pressed the right mouse button, the angle between these three atoms is displayed. If you have clicked on four atoms and then pressed the right mouse button, the torsion angle between these four atoms is displayed. To delete the displayed values use either the `Geometry` menu items or repeat the steps above. Clicking with the left mouse button on an atom may also be necessary for setting or selecting some atom specific values (vide infra).
- Holding down one of the shift keys and clicking on a molecule with the left mouse button
  The molecule is selected. The window title will show the name of this molecule. All subsequent translations/rotations act only on this molecule.
- Pressing the right mouse button without clicking on an atom before
  A menu will appear. The menu contains the following topics (the key combination in parentheses can be used as a shortcut in the English version):

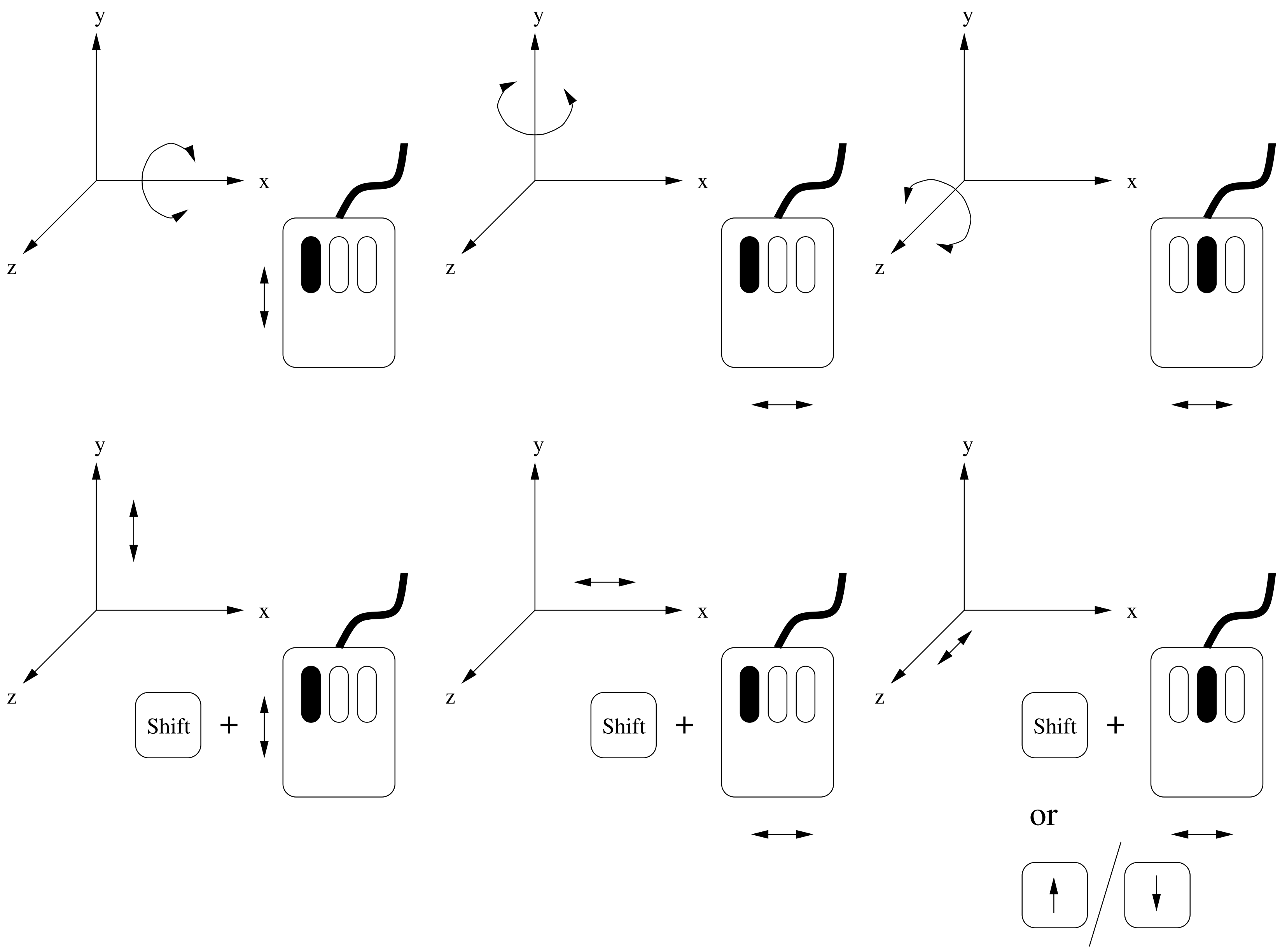
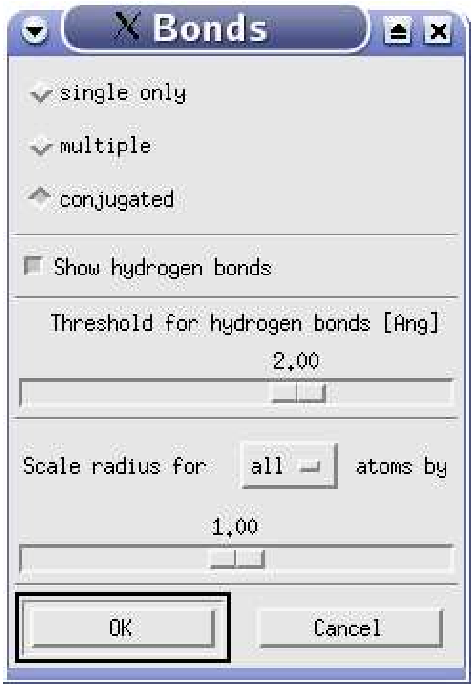


Figure 1: How the mouse can be used to manipulate a molecule or an annotation in the main window.

- Molecule ...
  A submenu is provided with the following topics:
  * Load molecule ...
    Brings up a file selection box to load a molecule.
  * Save molecule ...
    Brings up a format selection dialog and a file selection box to save the currently selected molecule to file. Output formats are supported through external filters (similar to the input filters) and can be installed using options in the `viewmolrc` file (cf. p. 45). Coordinates and bond information are passed to the corresponding output filter which writes the file. Currently, the only output formats provided are Accelrys car-files, MDL mol-files, and TURBOMOLE.
  * Delete molecule ...
    Deletes the currently selected molecule.
  * New molecule ...
    Brings up the molecule editor and starts the building of a new molecule (cf. p. 27).
  * Modify molecule ...
    Brings up the molecule editor to modify an existing molecule (cf. p. 27).
- Select molecule
  Provides a submenu with the names of all molecules currently loaded and an item `All` and can be used to change the currently selected molecule. Other possibilities to select a molecule consist of clicking on the molecule (preferably while holding the shift key down) or pressing the `Tab` key, which cycles through all entries in the `Select molecule` submenu.
- Wire model (Alt+W)
  The molecule will be drawn with lines. This is the default.
- Stick model (Alt+T)
  The molecule will be drawn with sticks.
- Ball and stick model (Alt+A)
  The molecule will be drawn with balls and sticks.
- CPK model (Alt+C)
  The molecule will be drawn with CPKs.
- Geometry ...
  A submenu is provided with the following topics:
  * Clear all (Ctrl+A)
    If this topic is selected all labels of bond lengths, bond angles, and torsion angles are deleted.
  * Clear last (Ctrl+L)
    If this topic is selected the label of a bond length, bond angle, or torsion angle created last is deleted.
  * Undo geometry change (Ctrl+U)
    If this topic is selected, the last geometry change (cf. p. 27) is reversed. All geometry changes are buffered in the undo buffer and can be reversed one by one by repeatedly using this menu item. Geometry changes can also be reversed through the molecule editor.
- Bond types ...
  Brings up the dialog box shown in Figure 2.


Bonds

single only

multiple

[✓ conjugated]

Show hydrogen bonds

Threshold for hydrogen bonds [Ang]

2.00

Scale radius for all atoms by

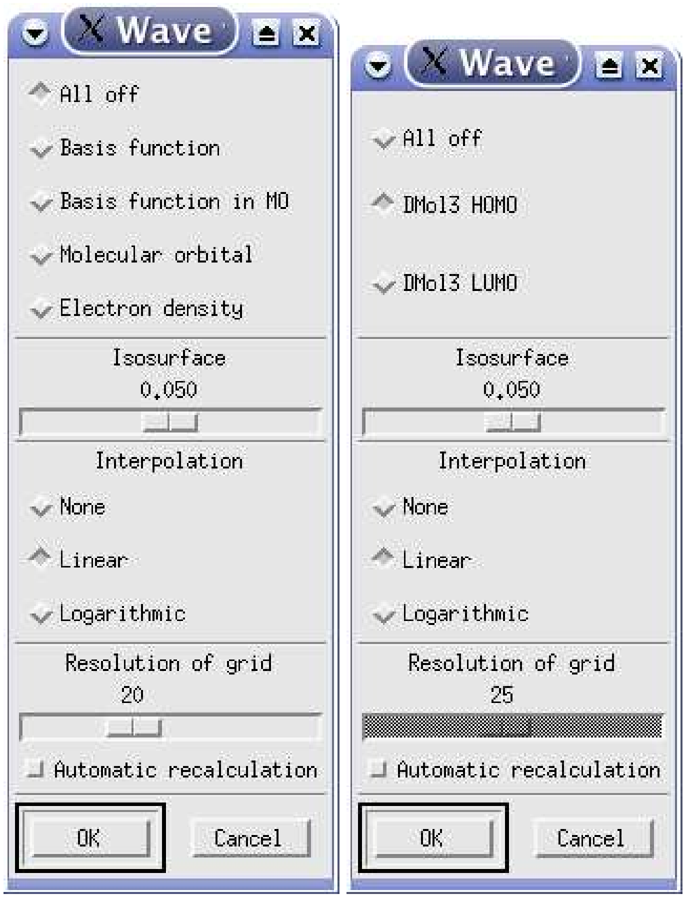
1.00

OK

Cancel

Figure 2: The dialog box for setting options for bonds.

- single only
  All bonds are drawn as single bonds.
- multiple
  VIEWMOL determines the bond order for each bond considering connectivity and elements only (only the following elements are used in the determination of bond orders: H, C, N, O, F, Si, P, S, Cl, Ge, Br, I). VIEWMOL then draws bonds with the corresponding bond order. Bond orders can also be changed in the molecule editor.
- conjugated
  VIEWMOL determines bond orders as for the “multiple” option. It then determines whether multiple bonds are conjugated and draws them as such. This is the default, but can be overwritten using resources (see p. 47).
- Show hydrogen bonds
  This button toggles the display of hydrogen bonds. Hydrogen bonds are determined automatically by VIEWMOL based on a distance threshold.
- Threshold for hydrogen bonds [Ang]
  This slider can be used to set the distance threshold for the automatic determination of hydrogen bonds. A hydrogen bond is shown if the distance between a hydrogen atom and another atom is larger than the sum of their radii, but smaller than this threshold. The default is 2 Å.
- Scale radius for all atoms by
  This menu and the slider beneath it can be used to scale the Van der Waals radius for an element. Since the Van der Waals radius determines the connectivity of the atoms in the molecule, changing it will also change the connectivity. The option menu can be used to select which element to scale and the slider allows to set the scaling factor.



[✓ All off]
Isosurface 0.050
Interpolation: [✓ Linear]
Resolution of grid 20

[✓ DMol3 HOMO]
Isosurface 0.050
Interpolation: [✓ Linear]
Resolution of grid 25

Figure 3: The dialog box for setting options for wave function related topics.

- Wave function ... (Alt+V)
  The dialog box shown in Figure 3 is presented. This topic is only available with the outputs of electronic structure programs (such as TURBOMOLE, GAUSSIAN, or MOPAC), and even then only if either MO coefficients and basis functions or a grid with an orbital could be read in. If TURBOMOLE output is used and the point group of the molecule in `$symmetry` is not C_1, TURBOMOLE's moloch program must be available (vide supra, p. 5) and only basis functions, occupied MO's, and electron densities can be drawn in this case. Since TURBOMOLE can handle up to g functions and GAUSSIAN can handle up to f functions the same limitations apply to VIEWMOL.

  If any wave function related drawing is displayed and the grid resolution is changed the drawing disappears and the recalculation has to be explicitly demanded by selecting this menu item again, since large molecules require significant time for the recalculation. At the top of this dialog box are five buttons which can be used to select the property which shall be shown.

  * All off
    This topic disables the drawing of any wave function related topic. This is the default.
  * Basis function
    This topic allows drawing of basis functions. After selecting it and closing the wave function dialog VIEWMOL prompts for an atom in its main window. Clicking on an atom with

the left mouse button will present a dialog box with all basis functions centered on this atom. After selecting one of these basis functions and pressing the OK button the corresponding basis function will be drawn.

  ∗ Basis function in MO
    This topic allows drawing of basis functions multiplied by the corresponding coefficient in a molecular orbital. This topic works similar to the previous one, except that the menu will show the MO coefficients in front of all basis functions. If no molecular orbital has been selected in the MO energy diagram window a warning message will be displayed.
  ∗ Molecular orbital
    This topic allows the drawing of a molecular orbital. If no molecular orbital has been selected in the MO energy diagram window a warning message will be displayed.
  ∗ Electron density
    This topic allows the drawing of the total electron density.

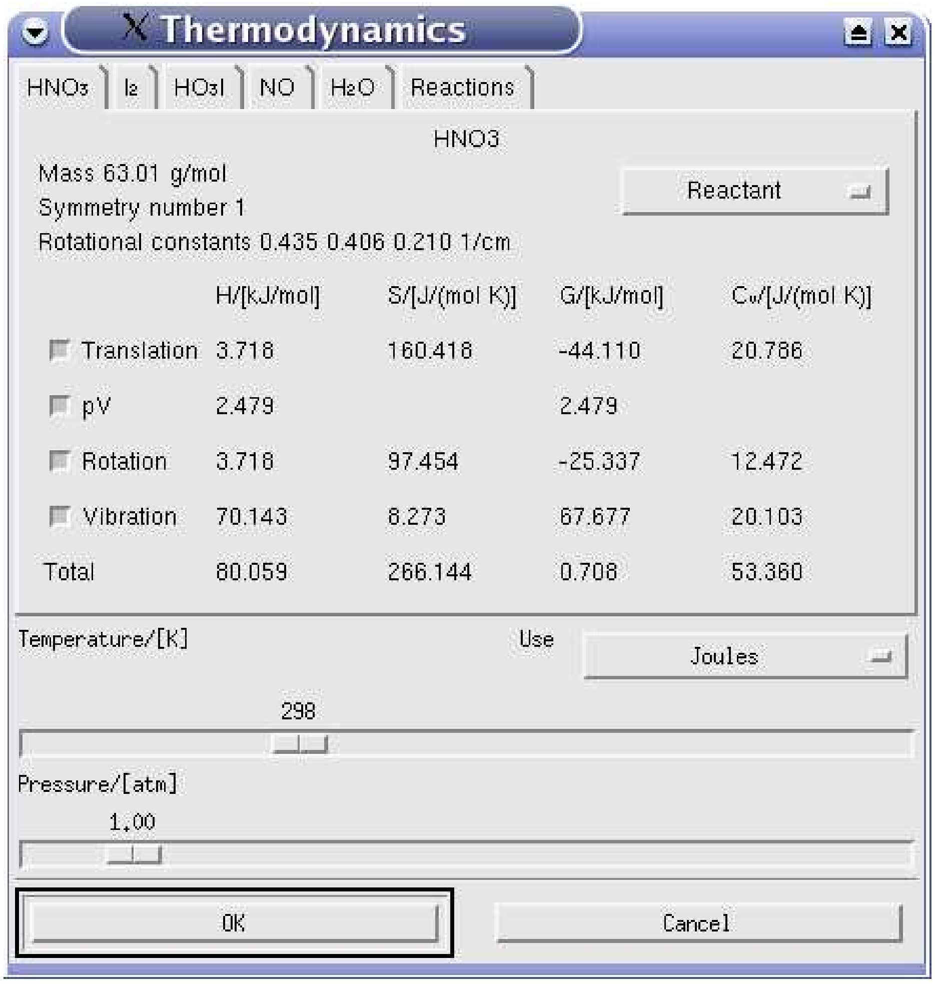
If grids have been read in there will be additional buttons to select one of these grids. If no basis functions and MO coefficients have been read in, but grids only the buttons to select a grid and the "All off" button are displayed (cf. right screenshot in Fig. 3).

Next to these buttons there is a slider which can be used to select the value of the isosurface used to draw the property selected. Following this slider another three buttons allow the selection of the interpolation method used in drawing the property.

  ∗ None
    No interpolation is done. The resulting drawing normally has a lot of edges.
  ∗ Linear
    A linear interpolation is done between grid points. This gives a much smoother surface.
  ∗ Logarithmic
    A logarithmic interpolation is done between grid points. This improves the quality of drawing further.

The default is linear, but this can be overwritten in the resource file (vide infra, p. 47). Following is another slider with can be used to set the resolution of the grid. As higher the number selected here as finer the grid and as smoother the resulting surface, but the calculation time goes with the third power of this number. Default is 20, but this can be overwritten in the resource file (vide infra, p. 47). At the bottom of the dialog box is a toggle button which can be used to turn automatic recalculation of MOs etc. on whenever the energy level is changed. Since these calculations can be quite time consuming, this button is off by default, but this can be overwritten in the resource file (vide infra, p. 47).

– Energy level diagram (Alt+E)
  A new window will appear which shows the calculated energies of the MOs in an energy level diagram. This topic is only available using DMOL, GAMESS, GAUSSIAN 9X, MOPAC, or TURBOMOLE outputs. In TURBOMOLE outputs either the data group `$scfmo` or the data groups `$uhfmo_alpha` and `$uhfmo_beta` must be available.
– Optimization history (Alt+O)
  A diagram is plotted in a second window which shows the energies and gradient norms of the geometry optimization. With the cursor keys for moving to the left <←> and to the right <→> one can see how the geometry optimization works. Alternatively, the red cross can be dragged with the mouse.



HNO3

Mass 63.01 g/mol
Symmetry number 1
Rotational constants 0.435 0.406 0.210 1/cm

	H/[kJ/mol]	S/[J/(mol K)]	G/[kJ/mol]	Cv/[J/(mol K)]
Translation	3.718	160.418	-44.110	20.786
pV	2.479		2.479	
Rotation	3.718	97.454	-25.337	12.472
Vibration	70.143	8.273	67.677	20.103
Total	80.059	266.144	0.708	53.360

Temperature/[K] 298

Pressure/[atm] 1.00

Figure 4: The dialog box for handling thermodynamics calculations.

– Show forces (Alt+F)
  The calculated forces acting on the atoms are drawn as arrows. This topic is not available using DISCOVER outputs. The topic is also not available, if no forces were found for the current coordinates.

– Spectrum (Alt+S)
  A new window will appear which shows the calculated spectrum for the molecule. This topic is only available if a force constants calculation has been performed.

– Thermodynamics (Alt+Y)
  The dialog box shown in Figure 4 is displayed. On the top of this dialog box a number of tabs can be found. All tabs except the last one allow to select the display of thermodynamical data for one of the molecules loaded in VIEWMOL. The last tab will display thermodynamical data for one or more defined reactions among the molecules loaded.

  The screen for a molecule shows the title of that molecule on top. Underneath the molecular mass (in g/mol), the symmetry number, and either the rotational constants (for molecules in cm^{-1}) or the density (for solids in g/cm^3) are displayed on the left hand side. On the right hand side there is a popup menu which allows the user to select whether this molecule should be a

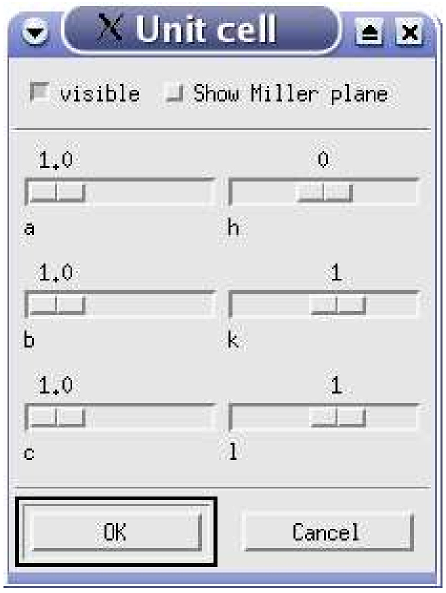


Figure 5: The dialog box for modifying the unit cell.

reactant or a product in a reaction, or whether it is not involved in a reaction at all. The last item in this popup menu, `All reactions`, will force VIEWMOL to determine a linear independent set of possible reactions between all molecules where this item has been selected.

The remainder of the screen shows thermodynamical data for the molecule. On the left hand side there are a number of buttons which can be used to select which contributions (translation, pV, rotation, vibration) to include in the total which is used to calculate thermodynamical data for a reaction. The enthalphy, entropy, Gibbs energy, and heat capacity are listed to the right, split into contributions from translation, pV, rotation, and vibration.

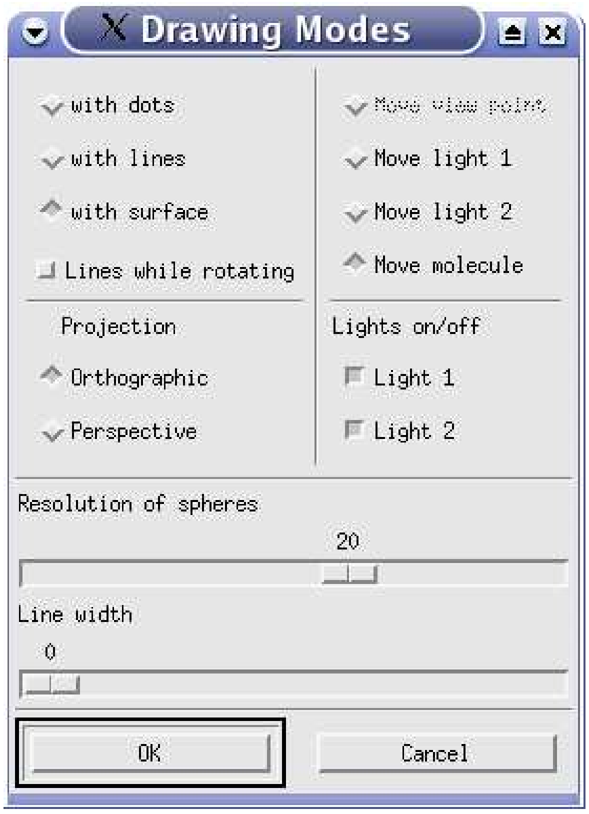
The screen for reactions (not shown) shows the reaction equation on top. Underneath the values of reaction enthalphy, entropy, Gibbs energy, heat capacity, and the (decadic) logarithm of the equilibrium constant are listed. The electronic and statistical-mechanic contributions to the reaction enthalphy are listed separately.

At the bottom all screens share a popup menu for selecting the units to be used (Joules, calories, or thermochemical calories) and two sliders. The top slider can be used to select the temperature at which the thermodynamical data are to be calculated, the bottom slider serves the same purpose for the pressure.

– Unit cell (Alt+N)
  The dialog box shown in Figure 5 is displayed. The first button on the left, `visible`, allows to turn the display of the unit cell on or off. The three sliders on the left hand side can be used to increase or decrease the number of unit cells displayed in each crystallographic direction. By default between one and five unit cells can be selected.

  The first button on the right, `Show Miller plane`, allows to turn the display of Miller planes on or off. The three sliders underneath this button can be used to select the Miller plane to be displayed. By default, all combinations between -5 and 5 for the Miller indices are possible.

– Show ellipsoid of inertia (Alt+I)
  The display of the ellipsoid of inertia is toggled.



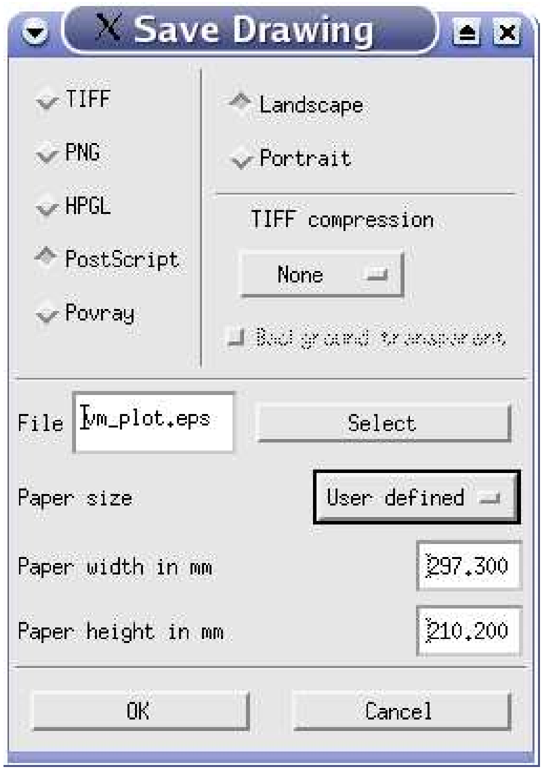
Resolution of spheres 20

Line width 0

Figure 6: The dialog box for setting options for drawing style.

- Drawing modes ... (Alt+M)
  The dialog box shown in Figure 6 is provided:
  - with dots
    Drawing of sticks, balls, CPKs and/or molecular orbitals is done as a dot cloud.
  - with lines
    Drawing of sticks, balls, CPKs and/or molecular orbitals is done with meshes.
  - with surface
    Drawing of sticks, balls, CPKs and/or molecular orbitals is done with an opaque surface which has the properties defined in the `viewmolrc` file (these properties hold for sticks as well as for balls or CPKs).
  - Lines while rotating
    If this option is selected the drawing of the molecule will be done with lines during translations and rotations. This speeds up movements on low-end graphics systems. The default is off, but this can be changed in the resource file.
  - Orthographic projection
    The molecule is drawn using an orthographic projection.
  - Perspective projection
    The molecule is drawn using a perspective projection. This kind of projection resembles more closely the way the human eye perceives things.
  - Move molecule

  If this button is selected all translations and rotations carried out with the mouse act on the currently selected molecule.
  * Move view point
    If this button is selected all translations and rotations carried out with the mouse act on the viewpoint. This option is only available if perspective projection is used. Moving the view point allows the user to move into a molecule.
  * Move light 1
    If this button is selected all translations and rotations carried out with the mouse act on light 1.
  * Move light 2
    If this button is selected all translations and rotations carried out with the mouse act on light 2.
  * Lights on/off, Light 1
  * Lights on/off, Light 2
    These two button can be used to switch lights on and off. Lights have only an effect if the drawing mode is "with surface" and either the stick, ball-and-stick, or CPK model is selected.
  * Resolution of spheres
    The number of polygons used for the drawing of sticks, balls and/or CPKs is changed. A higher value makes the surfaces more smoothly looking, but also decreases drawing speed. A lower value makes the surfaces rougher looking, but increases drawing speed.
  * Line width
    The line width used for drawing the molecule as wire frame model can be selected. A value of 0 means dynamic determination of the line width based on the size of the window. This value is the default.
- Ground color (Alt+G)
  The color editor appears in a separate window which can be used to change the color of the ground if perspective drawing is enabled (see description of the color editor below, p. 31).
- Background color (Alt+B)
  The color editor appears in a separate window which can be used to change the background color of the window (see description of the color editor below, p. 31).
- Label atoms (Alt+L)
  The atoms are labeled with atom symbols from input files. A number counting the atoms according to their order in the input is concatenated to the symbol.
- Annotate (Ctrl+N)
  Annotations can be created in the main window using this topic. After selecting this topic the cursor turns into a text input cursor. Clicking at any point in the main window now allows to enter an arbitrary text string. Pressing <return> ends the annotation function. Existing annotations can be edited by simply clicking on them or deleted by deleting all characters in the string. Annotations can be moved in the same way as the molecule can be moved: hold a shift key down, click on the annotation and move the mouse. Annotations support the clipboard, i. e. annotations can be cut and pasted between applications.
- Run script
  This topic pops up a second menu which can contain any number of items each representing an installed Python script. At least one entry is always present `Select ...` which allows the user to run arbitrary Python scripts.



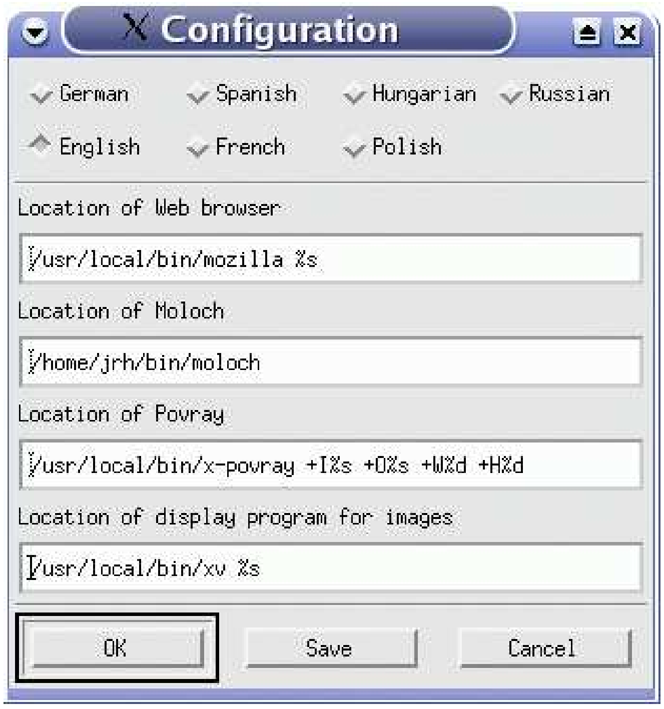
[✓ PostScript]
[✓ Landscape]
TIFF compression: None
File: vm_plot.eps
Paper size: User defined
Paper width in mm: 297.300
Paper height in mm: 210.200

Figure 7: The dialog box for setting options to save a drawing

    * Select ... (Ctrl+R)
      A file selection box is displayed which allows the selection of a Python script to be run within VIEWMOL.

- Save drawing (Alt+D)
  The dialog box shown in Figure 7 is provided which can be used to set file formats, file names and other options for writing the drawing to a file.
    * TIFF
      The current drawing of the molecule is written out as a TIFF file. The “TIFF compression” menu permits the selection of a compression algorithm. If normal modes are animated while this option is selected a series of 20 TIFF files will be written out, each containing a single frame of the animation. If the optimization history is animated when this option is selected one TIFF file will be written for each conformation of the molecule encountered in the optimization.
    * PNG
      The current drawing of the molecule is written out as a PNG file. The “Background transparent” check box allows to create a PNG file with a transparent background. If normal modes are animated while this option is selected a series of 20 PNG files will be written out, each containing a single frame of the animation. If the optimization history is animated when this option is selected one PNG file will be written for each conformation of the molecule encountered in the optimization. By using standard image manipulation tools available on the Internet it is possible to generate a video file (MPEG) from these TIFF or PNG files which can be included in multimedia documents (vide infra, p. 44).

    - HPGL
      The current drawing of the molecule is written out as a HPGL file for plotting on a plotter or a laser printer. This topic is not available if the drawing is done with sticks, balls, or CPKs and surfaces or when animations are running.
    - PostScript
      The current drawing of the molecule is written out as a PostScript file. If this topic is selected while the drawing is done with sticks, balls, or CPKs and surfaces a PostScript file containing a bitmap is created. Such a file can become rather large (the size depends on the size of the window) and it is not resolution independent which can result in artefacts on printing.
    - Povray
      The current drawing of the molecule is written out as an input file for POVRAY 3.5. This topic is only available if the drawing is done with sticks, balls, or CPKs. If molecular orbitals are drawn this topic is only available if the MO is drawn with a surface. If normal modes are animated while this option is selected a series of 20 input files for POVRAY will be written out, each containing a single frame of the animation.
    - Landscape
    - Portrait
      The orientation of the drawing on the page can be chosen if the drawing is written out as either a HPGL or a Postscript file.
    - TIFF compression
      Can be used to select the compression mode for TIFF files. Due to a software patent on the LZW compression algorithm this compression cannot be provided.
    - Background transparent
      If a PNG file is to be written this check box can be used to specify a transparent background.
    - File
      The name for the file to be generated.
    - Select
      Brings up a file selection box to select the name for the file to be generated.
    - Paper size
      Can be used to select the paper size for HPGL and PostScript outputs.
    - Paper width in mm
    - Paper height in mm
      Input user defined paper width and height. Only available if "Paper size" is set to "User defined".
  - Help/Manual (Alt+H)
    This topic opens a window with the VIEWMOL manual. It requires that the file `viewmol.html` can be accessed in the location `$VIEWMOLPATH/doc`.
  - Configuration ...
    The dialog box shown in Figure 8 is provided. At the top of this dialog the language VIEWMOL uses in its interface can be selected. VIEWMOL loads its language specific data from files `Viewmol` from the directory `$VIEWMOLPATH/locale/X` where `X` stands for a language identifier. The four text input fields can be used to specify the location of helper programs VIEWMOL needs for some of its operations. If the corresponding program was found in the path the dialog box will already show the correct information. If the programs specified here need file names as parameters, put `%s` as a place holder for the file name in the command.


Configuration

German Spanish Hungarian Russian
[✓ English] French Polish

Location of Web browser
/usr/local/bin/mozilla %s
Location of Moloch
/home/jrh/bin/moloch
Location of Povray
/usr/local/bin/x-povray +I%s +O%s +W%d +H%d
Location of display program for images
/usr/local/bin/xv %s

Figure 8: The dialog box for setting configuration options.

A the bottom is a button `Save` which allows the information entered in this dialog as well as some other settings to be stored as resources in `$HOME/.Xdefaults`. The following settings are saved: position and size of all open windows, window colors, selected model, selected drawing mode, selected bond type, setting of "lines while rotating", selected interpolation mode, resolution of spheres, line width, selected isosurface, selected resolution for density of states, setting of "automatic recalculation", paper size, and hydrogen bond threshold. **Note:** On Linux the setting of resources is kept across different invocations of the program. Saving the configuration and restarting VIEWMOL will therefore apparently not work. To get rid of the old resource settings issue the command `xrdb -remove $HOME/.Xdefaults` or log out and in again.

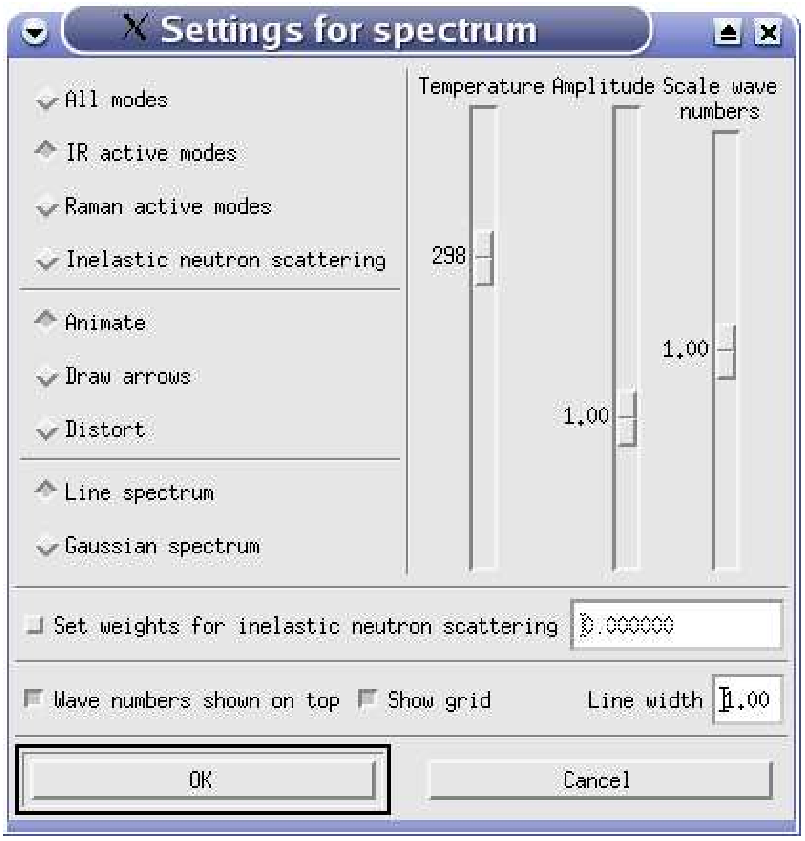
- Quit Viewmol (Q)
  This quits the program.

6.3 The Spectrum Window

Choosing `Spectrum` from the main window menu will result in a new window showing the calculated spectrum for the molecule. In this window the mouse acts as follows:

- Clicking with the left mouse button on a line in the spectrum
  The molecule shows the corresponding normal vibration.

- Clicking with the middle mouse button in the window, holding it down and moving the mouse
  This displays a rubber band box with which one can zoom into the spectrum.


Settings for spectrum

All modes
[✓ IR active modes]
Raman active modes
Inelastic neutron scattering

[✓ Animate]
Draw arrows
Distort

[✓ Line spectrum]
Gaussian spectrum

Temperature 298
Amplitude 1.00
Scale wave numbers 1.00

Set weights for inelastic neutron scattering 0.000000

Wave numbers shown on top Show grid Line width 1.00

Figure 9: The dialog box for setting options for the spectrum

- Pressing the right mouse button
  A menu will appear.

  The menu contains the following topics:

  - Settings for spectrum ... (Alt+S)
    Selecting this topic displays the dialog box shown in Figure 9.
    - Type of spectrum
      The buttons labeled `All modes`, `IR active modes`, `Raman active modes`, and `Inelastic neutron scattering` can be used to select the type of spectrum desired. IR active modes are the default.
    - Animation
      The buttons labeled `Animate`, `Draw arrows`, and `Distort` can be used to select whether the normal modes are to be shown animated or with arrows or whether you want to distort the molecule along a normal mode. A distorted molecule can be saved using the `Save molecule` option from the main menu. Animation is the default.
    - Line shapes
      The buttons `Line spectrum` and `Gaussian spectrum` can be used to select whether the spectrum is drawn as simple line spectrum or whether a Gaussian band shape [1] should be applied. Line spectrum is the default.

    * Set weights for inelastic neutron scattering
      When you activate this option you can enter a value in the field to the right. After choosing the `OK` button all weights are drawn at the atoms in the main window and you can set an atom's weight just by clicking on it with the left mouse button. All weights are set initially to zero, so that selecting `Inelastic neutron scattering` as spectrum type produces nothing.
    * Wave numbers shown on top
      This option controls the orientation of the spectrum. The default is to have zero intensity on top with the bands pointing downwards. If this option is deactivated zero intensity is at the bottom and the bands are pointing upwards.
    * Show grid
      This option controls whether a grid is shown in the spectrum window. The default is on.
    * Line width
      The value entered in this text box controls the line width of the drawing in the spectrum window.
    * Temperature
      This slider can be used to set the temperature. The temperature is used for the calculation of the inelastic neutron scattering intensities and for the Gaussian shaped spectrum.
    * Amplitude
      This slider can be used to change the amplitude of the vibration. Its value is multiplied with the standard amplitude of a vibration. This slider can also be used to change the distortion of the molecule while `Distort` is selected.
    * Scale wave numbers
      This slider can be used to scale the wave numbers.
- Select molecule
  Provides a submenu with the names of all molecules currently loaded and can be used to change the molecule for the currently displayed spectrum. Another possibility to select a molecule consists of pressing the `Tab` key, which cycles through all entries in the `Select molecule` submenu.
- Imaginary wave numbers
  If the conformation of the molecule is a saddle point you have imaginary wave numbers. The corresponding "normal modes" can be shown by selecting a imaginary wave number from this submenu.
- Read observed spectrum ... (Alt+R)
  This topic can be used to read a spectrum from a file and display it along the calculated spectrum. Selecting this topic opens a file selection box to chose the file. This file has to contain a spectrum with one wave number and intensity per line. All points read are connected by a line to form a continuous spectrum (there is currently no possibility to read a line spectrum). Lines with a '#' or a letter in the first column are ignored.
- Delete observed spectrum ... (Alt+E)
  This topic deletes a spectrum read with "Read observed spectrum".
- Zoom out (Alt+Z)
  This topic can be used to zoom out of the spectrum after previous zoom-ins. The zoom mechanism stores all previous enlargement steps. By selecting this topic you move back one step.

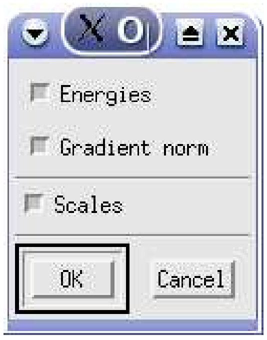


Figure 10: The dialog box for setting options for the optimization history

- Save drawing (Alt+D)
  The same dialog box as for `Save drawing` in the main menu is shown and allows you to save the spectrum as a TIFF, PNG, HPGL, or PostScript file (cf. p. 19).
- Foreground color (Alt+F)
  The foreground color of the spectrum can be changed using the color editor (vide infra, p. 31).
- Background color (Alt+B)
  The background color of the spectrum can be changed using the color editor (vide infra, p. 31).
- Quit spectrum (Alt+Q)
  This closes the spectrum window.

• Pressing the arrow keys <→> and <←>, respectively
  The next lower and higher wave number, respectively is selected and the molecule shows the corresponding normal vibration. Normal vibrations of imaginary wave numbers can be displayed in this manner, too.

6.4 The Optimization History Window

Choosing `Optimization history` from the main window menu will result in a new window showing the energy and gradient norm in dependence of the step number of the geometry optimization. In this window the mouse acts as follows:

• Pressing the left mouse button and holding it down
  The red cross showing the actual step of the geometry optimization can be moved with the mouse. The main window shows the corresponding geometry. Changing the actual step can also be achieved using the cursor keys for moving to the left <←> and to the right <→>, respectively.

• Pressing the right mouse button
  A menu will appear.

  The menu contains the following topics:

  - Settings for history ... (Alt+S)
    Selecting this topic displays the dialog box shown in Figure 10.

        * Energies
          If this button is selected the energy curve is drawn. The default is on.
        * Gradient norms
          If this button is selected the gradient norm curve is drawn. The default is on.
        * Scales
          This button toggles drawing of the scales on and off. The default is on.
    - Select molecule
      Provides a submenu with the names of all molecules currently loaded and can be used to change the molecule for the currently displayed optimization history. Another possibility to select a molecule consists of pressing the `Tab` key, which cycles through all entries in the `Select molecule` submenu.
    - Animate (Alt+A)
      Selecting this topic will animate the optimization history. To stop the animation select this topic again.
    - Save drawing (Alt+D)
      The same dialog box as for `Save drawing` in the main menu is shown and allows you to save the optimization history as a TIFF, PNG, HPGL, or PostScript file (cf. p. 19).
    - Color for energy (Alt+E)
      The color for the energy curve can be changed using the color editor (vide infra, p. 31).
    - Color for gradient norm (Alt+G)
      The color for the gradient norm curve can be changed using the color editor (vide infra, p. 31).
    - Background color (Alt+B)
      The background color of the diagram can be changed using the color editor (vide infra, p. 31).
    - Quit history (Q)
      This closes the optimization history window.

6.5 The Energy Level Diagram Window

Choosing `Energy level diagram` from the main window menu will result in a new window showing the calculated MO energies in an energy level diagram. In this window the mouse acts as follows:

- Clicking with the left mouse button on a line
  A box appears containing the symmetry and the energy for this MO. Selecting a MO can also be achieved by pressing the cursor keys for moving up <↑> and down <↓>, respectively.
- Clicking in the window, pressing the middle mouse button and holding it down
  With the rubber band box drawn one can zoom into the diagram.
- Pressing the right mouse button
  A menu will appear.

  The menu contains the following topics:

    - Settings for energy level diagram ... (Alt+S)
      Selecting this topic displays the dialog shown in Figure 11.

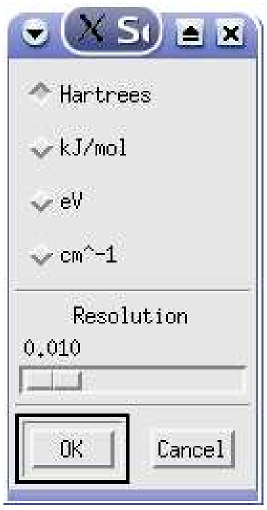


Figure 11: The dialog box for setting the energy units for the MO energy level diagram

  * Units
    The four buttons can be used to select the energy unit. Available are Hartrees (default), kJ/mol, eV, and cm^{-1}.
  * Resolution for density of states
    This slider can be used to change the resolution for the density of states. A smaller value results in a higher resolution. The default is 0.01. The minimum, maximum, and default values can be set in the resource file. (vide infra, p. 47).

- Select molecule
  Provides a submenu with the names of all molecules currently loaded and can be used to change the molecule for the currently displayed energy level diagram. Another possibility to select a molecule consists of pressing the `Tab` key, which cycles through all entries in the `Select molecule` submenu.
- Transition (Alt+T)
  This topic can be used to calculate the energy for a transition between two MO's. This topic is available only if one MO was selected by clicking on it. Choosing this menu topic followed by clicking on an other MO draws a line showing the transition and a box containing the symmetry labels and the energy difference between these two MO's. Further clicks on other MO's repeat calculations of energy differences. To leave this mode click either somewhere in the window were no MO's are or select this topic from the menu again.
- Zoom out (Alt+Z)
  This topic can be used to zoom out of the diagram after previous zoom-ins. The zoom mechanism stores all previous enlargement steps. By selecting this topic you move back one step.
- Save drawing (Alt+D)
  The same dialog box as for `Save drawing` in the main menu is shown and allows you to save the energy level diagram as a TIFF, PNG, HPGL, or PostScript file (cf. p. 19).
- Draw density of states (Alt+D)
  This topic can be used to toggle between the density of states and the energy level diagram. The energy level diagram is the default.

- Foreground color (Alt+F)
  The foreground color of the diagram can be changed using the color editor (vide infra, p. 31).
- Background color (Alt+B)
  The background color of the diagram can be changed using the color editor (vide infra, p. 31).
- Quit energy diagram (Q)
  This closes the energy level diagram window.

7 Editing molecules

VIEWMOL allows the building and editing of molecules. Bond lengths, bond angles, and torsion angles of an existing molecule can be changed, atoms can be replaced and added or deleted. New molecules can be build.

7.1 Changing bond lengths, bond angles, and torsion angles

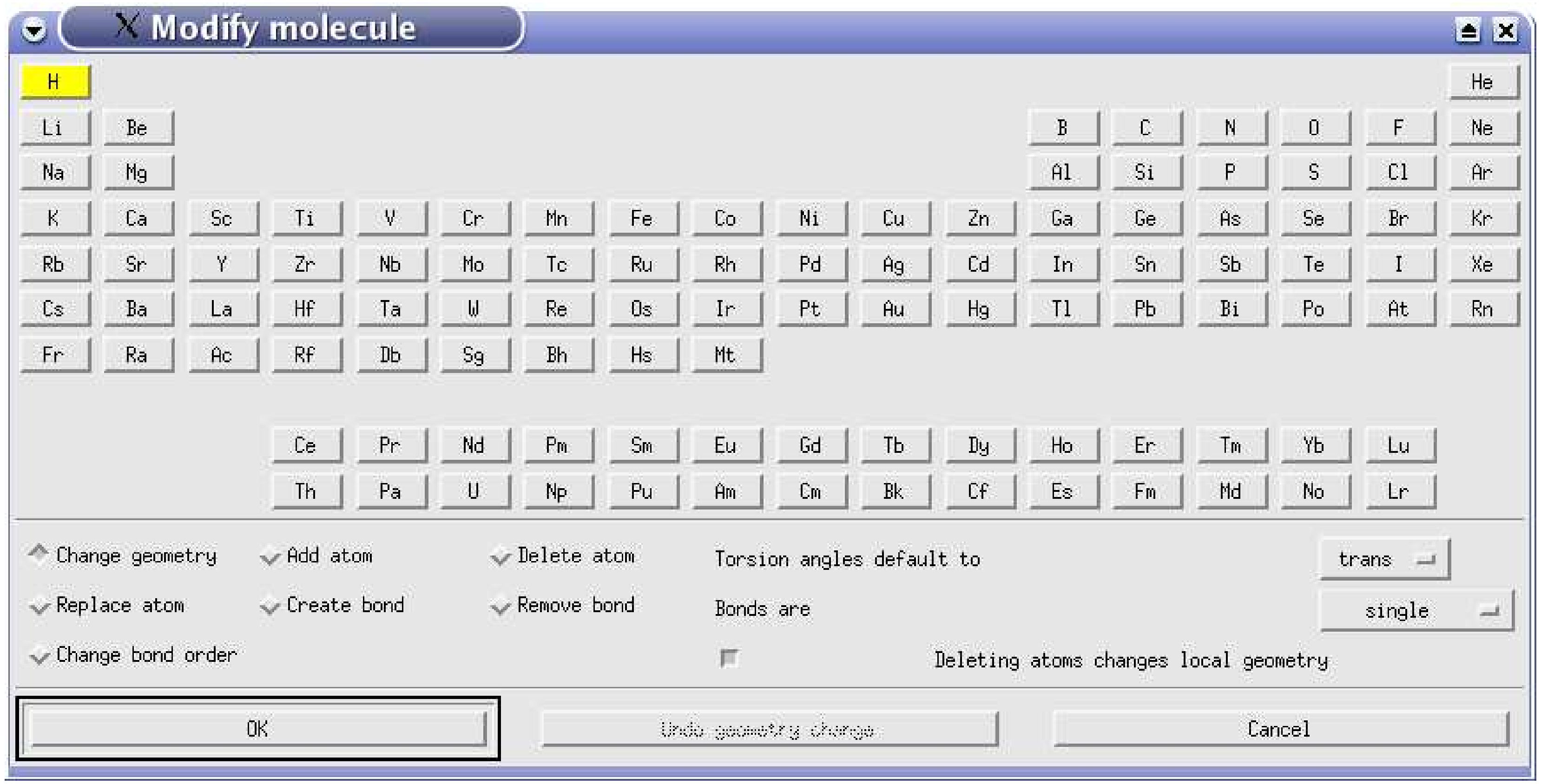
To change a bond length, a bond angle, or a torsion angle click on the corresponding atoms and then press the right mouse button. This will display the length of the bond and the value of the bond or torsion angle, respectively (cf. p. 9). Now click on the number with the left mouse button. A cursor will appear and the value displayed can be changed. After pressing `Return` the new value for the bond length, bond angle, or torsion angle will be set. The atom which has been clicked on first (and all atoms connected to it) will be moved. It is impossible to change bond lengths, bond angles, or torsion angles if they are part of a ring. All changes in geometry can be reversed by using `Undo geometry change` from the `Geometry` menu or the corresponding button in the molecule editor dialog box. The number of undos is unlimited.

7.2 Adding or replacing atoms

From the main window menu select the `Modify molecule` entry in the `Molecule` submenu. The dialog box shown in Figure 12 is displayed. The upper part contains the periodic table of elements and allows the selection of the element to be added or used as replacement.

In the middle there are a number of buttons for selecting different operations and certain defaults. These are the following:

- Change geometry
  If this item has been chosen atoms can be selected with the mouse and geometry changes carried out as described in the previous section.
- Add atom
  An atom of the element selected in the periodic table will be attached to the atom in the molecule clicked on with the left mouse button. The new bond will have the bond order selected in the editor dialog box and a bond length which is 90 % of the sum of the atomic radii (read from the `viewmolrc` file). The local geometry of the atom clicked on will be changed to reflect the current coordination of this atom (two bond partners – linear, three – trigonal planar, four – tetrahedral, five – trigonal bipyramidal, six – octahedral, seven – pentagonal bipyramidal etc.).


Modify molecule
H He
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
Cs Ba La Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
Fr Ra Ac Rf Db Sg Bh Hs Mt
Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr
Change geometry
Add atom
Delete atom
Replace atom
Create bond
Remove bond
Change bond order
OK
Undo geometry change
Cancel

Torsion angles default to trans

Bonds are single

Deleting atoms changes local geometry

Figure 12: The molecule editor

- Delete atom
  An atom clicked on with the left mouse button will be deleted. If `Deleting atoms changes local geometry` is turned on the local geometry of the atom(s) connected to the deleted one will be changed in the same way as described under `Add atom`. Otherwise the local geometry remains as before.

- Replace atom
  The atom clicked on with the left mouse button is replaced by the element selected in the periodic table. Bond lengths are adjusted to reflect the new element as long as the atom changed is not part of a ring.

- Create bond
  A new bond will be created between the two atoms clicked on with the left mouse button. This bond will have the order selected under `Bonds are`.

- Remove bond
  The bond between the two atoms clicked on with the left mouse button is deleted. Bonds created or removed by the user have precedence over bonds created automatically. This means that once a bond has been created by the user only the user can remove it and vice versa regardless of what happens to the molecule.

- Change bond order
  The bond between the two atoms clicked on with the left mouse button is assigned the bond border selected under `Bonds are`.

- Torsion angles default to
  While building a molecule bond lengths of newly created bonds are set to 90 % of the sum of the atomic radii of the bonded atoms and bond angles are assigned according to the coordination. Torsion angles can be selected from this menu. Available values are trans (180°), cis (90°), gauche (60°), and $-$gauche ($-60°$). This allows the construction of more complicated molecules. The torsion angle is always measured along the backbone of a molecule, i. e. while building e. g. a hydrocarbon chain the torsion angle is always measured between the carbon atoms. The backbone of a molecule is determined by counting all atoms attached to one atom and following the bonds which connect the atoms with the largest number of other atoms attached.

- Bonds are
  This menu allows the selection of the bond order for bonds. Available are single, double, and triple. Bond conjugation and hydrogen bonds are determined automatically according to the respective settings in the `Bond type` menu.

- Deleting atoms changes local geometry
  If an atom is deleted the local geometry of the atom(s) bonded to it can either be left unchanged or modified according to the new coordination. This switch can be used to select which behavior is preferred.

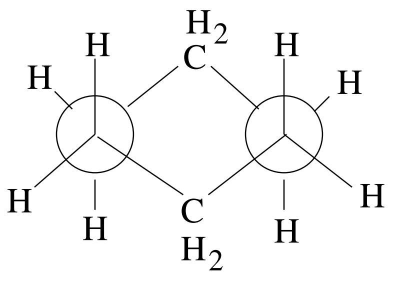
In the bottom row of the molecule editor dialog box is a button `Undo geometry change` which can be used to reverse all changes to the molecule (except changes to bonds). The number of undos is unlimited.

7.3 Example: Building cyclohexane

Cyclohexane in its chair configuration has a structure which seems quite complicated to generate, but it can easily be built using VIEWMOL's molecule builder. First, start VIEWMOL without any parameter on the command line and press `Cancel` in the file selection box which pops up. You should now have an empty window on your screen. Press the right mouse button within this window and select `Molecule/New molecule ...` from the pop-up menu. The molecule editor will open up. We start building cyclohexane by building one of the tetrahedra:

- Select `C` from the periodic table.
- Click with the left mouse button in VIEWMOL's window. This will place the first carbon atom at the origin.
- Click with the left mouse button on the first carbon atom. This will attach the second carbon atom to the first.
- Click with the left mouse button on the second carbon atom. This will attach the third carbon atom to the second.
- Select `H` from the periodic table and click twice on the central carbon atom. The molecule should now look like the left one in Figure 13.

To continue building cyclohexane we have to remember that the torsion angles among the carbon atoms alternate between gauche and −gauche which can be determined easily by looking at the Newman projection:



- Select `C` again and set the default torsion angle to gauche.
- Click on one of the end carbon atoms. Then select `H` and click on the same carbon atom two more times.
- Change the default torsion angle to −gauche and repeat the last two steps on the just added carbon atom. The molecule should now look like the one in the middle of Figure 13.
- Change the default torsion angle to gauche again and add a carbon atom and one hydrogen atom to the last carbon atom. The molecule will now look like the right one in Figure 13.
- Change the default torsion angle to trans and add the second hydrogen atom to the carbon atom added last.
- Now change the default torsion angle to −gauche and add one hydrogen atom to the next carbon atom in the ring, then change the default torsion angle to trans and add the second hydrogen atom.

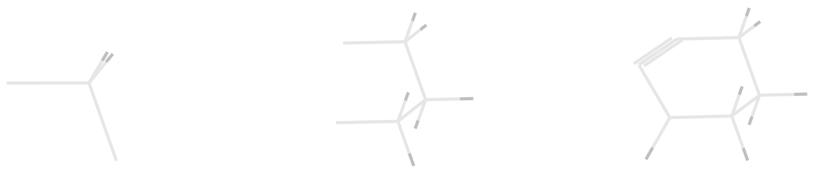


Figure 13: Building cyclohexane

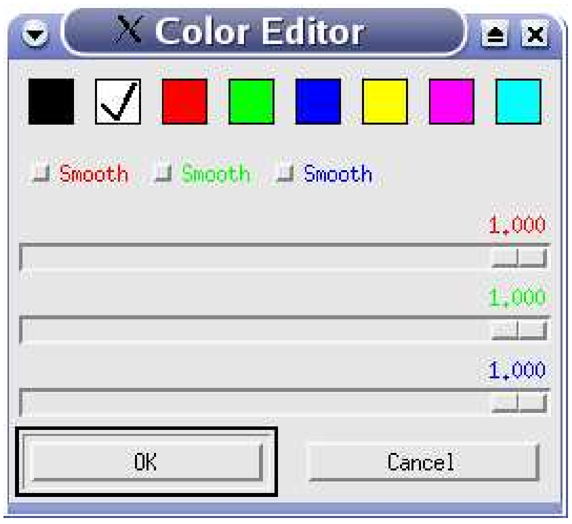


Figure 14: The color editor

- Finally, complete cyclohexane by adding first a hydrogen with the default set to trans and than a hydrogen with the default set to gauche to the next carbon atom in the ring.

8 The color editor

All menus contain items for changing colors. Selecting one of these items displays the color editor (cf. Figure 14). In the first row of the color editor dialog box are a number of different colored boxes. You can select one of these colors just by clicking in the corresponding box with the left mouse button. The color you want to change in the picture changes, too, but this is not a permanent change until you click the `OK` button.

If none of the colors in the boxes satisfies your needs you can set up your own color by moving one of the sliders for red, green, or blue with the left mouse button. The color in the picture you want to change shows the currently selected color and by clicking the `OK` button you change your color permanently.

If the color editor has been brought up to edit the background color of the main window three buttons labeled `Smooth` are present. Activating any of these buttons will create a color ramp for the corresponding color

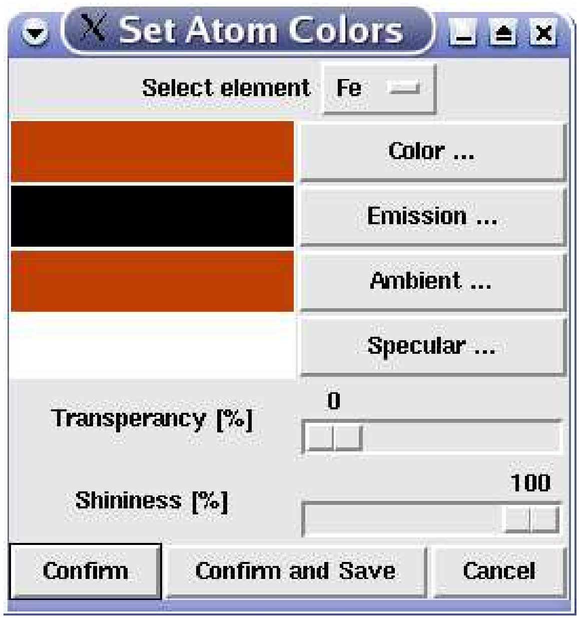


Figure 15: The dialog to change atom colors

and mix it with the other RGB components. The slider for this color then has no effect. That way interesting shading effects in the background can be generated.

9 Script tools

VIEWMOL can be extended using Python scripts. A number of such scripts are part of the VIEWMOL distribution and come installed in the "Run script" menu or as output filter. The following scripts are included:

- Demonstration
  This script runs a short demonstration of VIEWMOL's capabilities. It can be interrupted by pressing Ctrl-C.
- Set atom colors
  This script allows to interactively change the colors for an element. It allows to save the changes to the `viewmolrc` file so that manual editing of this file might no longer be necessary. If started from the "Set atom colors" submenu entry a dialog as shown in Fig. 15 is presented[4]. The pulldown menu at the top allows to select which element to operate on. It contains all the elements from the molecule(s) currently loaded in VIEWMOL. The four colored boxes and associated buttons allow the selection of the element color, the emission color, the ambient color, and the specular color. Pressing the button brings up a color selection dialog. Beneath the buttons there are two sliders which allow to change the transparency and the shininess for the selected element. The three buttons at the bottom allow the user to confirm the changes and therefore apply them permanently to the molecule(s), to confirm

[4]This script requires Tkinter and the Python Mega Widgets (Pmw) to be installed. Tkinter is usually available with Python. The Python Mega Widgets can be found at http://pmw.sourceforge.net/.

the changes and also save them in the `$HOME/.viewmolrc` file in the users `HOME` directory, or to cancel the changes and reverting back to the original colors.

- Template
  This is a script template which shows how to make Python scripts register themselves with VIEWMOL. It does nothing useful.
- UFF optimization
  If you have TURBOMOLE this script can be used to perform a geometry optimization using the UFF module shipping with TURBOMOLE. The necessary input files are generated, the UFF optimization is performed, and the resulting structure is read back.
- Gaussian 98 input file
  This script generates a complete input file for GAUSSIAN 98. It is not accessible from the "Run script" menu, but is installed as output filter and can therefore be accessed through the "Molecule/Save molecule ..." menu items. The script presents the dialog shown in Fig. 16 which allows setting of various options for GAUSSIAN 98. This script is courtesy of Stephan Schenk (stephan.schenk@uni-jena.de).

10 Programming VIEWMOL with Python

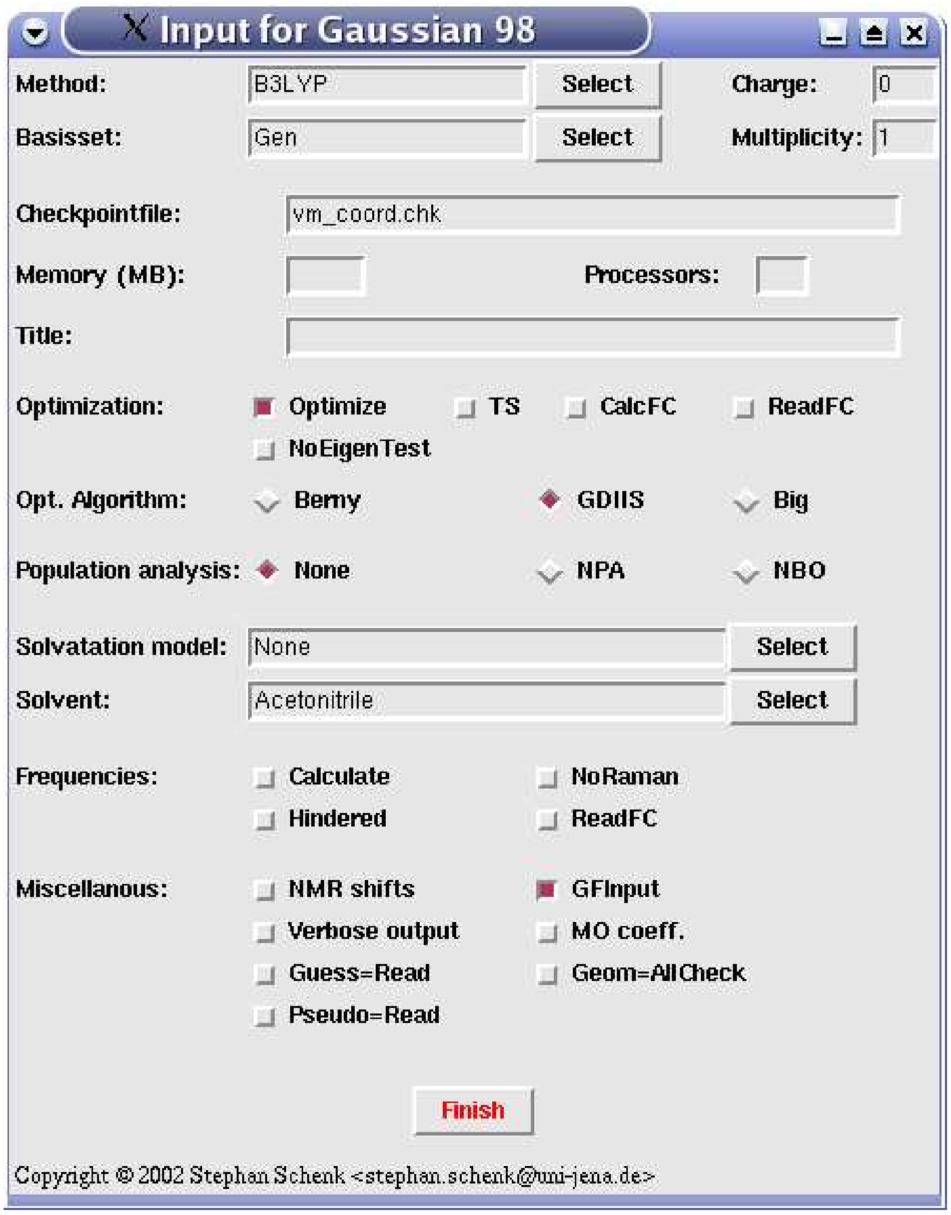
VIEWMOL can be programmed using Python. The Python interpreter embedded in VIEWMOL has been extended by the modules:

- `atom`
- `element`
- `energylevel`
- `history`
- `label`
- `light`
- `molecule`
- `spectrum`
- `viewmol`

which allow Python programs access to VIEWMOL's internal data model and functions. Each module provides a number of functions for use in Python programs.

10.1 The `atom` module

`getElement()`
  Returns the `element` object for an atom.



Input for Gaussian 98

Method:	B3LYP	Select	Charge:	0
Basisset:	Gen	Select	Multiplicity:	1

Checkpointfile: vm_coord.chk

Memory (MB): Processors:

Title:

Optimization: [✓ Optimize] TS CalcFC ReadFC NoEigenTest

Opt. Algorithm: Berny [✓ GDIIS] Big

Population analysis: [✓ None] NPA NBO

Solvatation model: None

Solvent: Acetonitrile

Frequencies: Calculate NoRaman Hindered ReadFC

Miscellanous: NMR shifts [✓ GFInput] Verbose output MO coeff. Guess=Read Geom=AllCheck Pseudo=Read

Finish

Copyright © 2002 Stephan Schenk <stephan.schenk@uni-jena.de>

Figure 16: The dialog to create an input file for GAUSSIAN 98

`coordinates(` *[x, y, z]* `)`
: Sets or returns the Cartesian coordinates for an atom. If *x*, *y*, and *z* are given as doubles the Cartesian coordinates are set to *x*, *y*, and *z*. Otherwise a tupel with the x, y, and z coordinates and the name of an atom is returned.

`radius(` *[rad]* `)`
: Sets or returns the Van der Waals radius for an atom. If *rad* is given it has to be a double. The Van der Waals radius is measured in Ångstrøms.

`radiusScaleFactor(` *[scaleFactor]* `)`
: Sets or returns the scale factor for the radius of an atom. If *scaleFactor* is given it has to be a double. The scale factor must be greater than or equal zero.

`neutronScatteringFactor(` *[factor]* `)`
: Sets or returns the neutron scattering factor for an atom. If *factor* is given it has to be a double. The neutron scattering factor must be greater than or equal zero.

`name(` *[name]* `)`
: Sets or returns the name of an atom. If *name* is given it has to be a string. The maximum length of the string is eight characters.

`replace(`*elementSymbol*`)`
: Replaces the atom with an atom of element *elementSymbol*. *elementSymbol* has to be a string and a valid element symbol.

`delete()`
: Deletes this atom. Note: Atoms cannot be deleted using Python's `del` operator since Python works on VIEWMOL's data structures.

10.2 The `element` module

`getSymbol()`
: Returns the element symbol for an element.

`darkColor(` *[red, green, blue]* `)`
: Sets or returns the red, green, and blue values for the darkest part of an atom of an element. If *red*, *green*, and *blue* are given they have to be floats between 0.0 and 1.0. If the color is queried, it is returned as a tupel of red, green, and blue values.

`lightColor(` *[red, green, blue]* `)`
: Sets or returns the red, green, and blue values for the lightest part of an atom of an element. If *red*, *green*, and *blue* are given they have to be floats between 0.0 and 1.0. If the color is queried, it is returned as a tupel of red, green, and blue values.

`emissionColor(` *[red, green, blue]* `)`
: Sets or returns the red, green, and blue values of the color an atom of an element emits. If *red*, *green*, and *blue* are given they have to be floats between 0.0 and 1.0. If the color is queried, it is returned as a tupel of red, green, and blue values.

`ambientColor(` *[red, green, blue]* `)`
: Sets or returns the red, green, and blue values of the ambient color of an atom of an element. If *red*, *green*, and *blue* are given they have to be floats between 0.0 and 1.0. If the color is queried, it is

returned as a tupel of red, green, and blue values.

`specularColor(`*[red, green, blue]* `)`
: Sets or returns the red, green, and blue values of the specular color of an atom of an element. If *red*, *green*, and *blue* are given they have to be floats between 0.0 and 1.0. If the color is queried, it is returned as a tupel of red, green, and blue values.

`shininess(`*[shininess]* `)`
: Sets or returns the shininess of an atom of an element. If *shininess* is given it has to be a float between 0.0 and 128.0.

`transperancy(`*[transperancy]* `)`
: Sets or returns the transperancy of an atom of an element. If *transperancy* is given it has to be a float between 0.0 and 1.0.

10.3 The `energylevel` module

`show()`
: Displays the energy level diagram window for a molecule.

`unit(`*[unit]* `)`
: Sets or returns the energy unit for the energy level diagram. *unit* has to be one of the integer constants `HARTREE`, `KJ/MOL`, `EV`, or `1/CM` defined in the `energylevel` module.

`resolution(`*[resolution]* `)`
: Sets or returns the resolution for the energy level diagram. *resolution* has to be a double greater than zero.

`mode(`*[mode]* `)`
: Sets or returns the mode for the energy level diagram. *mode* has to be one of the integer constants `ENERGY_LEVELS` or `DENSITY_OF_STATES` defined in the `energylevel` module.

`selectMO(`*mo1, [mo2]* `)`
: Selects one or two molecular orbitals. *mo1* and *mo2* have to be integers between 0 and the number of molecular orbitals.

`deselect()`
: Deselects molecular orbitals which have been selected with a call to `selectMO`.

`saveDrawing(`*format, filename*`)`
: Saves the energy level diagram to file. *format* has to be one of the integer constants `TIFF`, `PNG`, `HPGL`, or `POSTSCRIPT` defined in the `viewmol` module. *filename* has to be a string containing the name of the file the drawing is saved to.

The energy level diagram window can be closed by deleting the `energylevel` object.

10.4 The `history` module

`show()`
: Displays the optimization history window for a molecule.

`showEnergy(`*status*`)`
: Sets the displays of the energy curve to *status*. *status* has to be one of the integer constants `ON` or `OFF`

defined in the `viewmol` module.

`showGradient(`*status*`)`
: Sets the displays of the gradient curve to *status*. *status* has to be one of the integer constants `ON` or `OFF` defined in the `viewmol` module.

`showScales(`*status*`)`
: Sets the displays of the scales to *status*. *status* has to be one of the integer constants `ON` or `OFF` defined in the `viewmol` module.

`animate(`*status*`)`
: Sets the animation of the optimization history to *status*. *status* has to be one of the integer constants `ON` or `OFF` defined in the `viewmol` module.

`iteration(` *[iteration]* `)`
: Sets or returns the iteration displayed. *iteration* has to be an integer between 0 and the number of optimization steps for this molecule.

`saveDrawing(`*format, filename*`)`
: Saves the energy level diagram to file. *format* has to be one of the integer constants `TIFF`, `PNG`, `HPGL`, or `POSTSCRIPT` defined in the `viewmol` module, *filename* the name of the file the drawing is saved to.

The optimization history window can be closed by deleting the `history` object.

10.5 The `label` module

`label(` *[mode]* `)`
: Creates a new label. *mode* is an optional integer denoting whether the label is editable by the user, moveable by the user, or both. In the first case *mode* has to be set to the integer constant `EDITABLE`, in the second case to the integer constant `MOVEABLE`, and in the last case to `EDITABLE | MOVEABLE`. The integer constants are defined in the `label` module. The default mode is editable and moveable.

`translate(`*x, y, z*`)`
: Sets the postion of a label. *x*, *y*, and *z* are integers specifying the coordinates of the label in pixels. The z coordinate has no effect.

`text(` *[text]* `)`
: Sets or returns the text of an label. If present *text* has to be a string. The maximum number of characters for a label is limited to 255.

`setColor(`*red, green, blue, alpha*`)`
: Sets the red, green, blue, and alpha components of a label's color. *red*, *green*, *blue*, and *alpha* have to be floats between 0.0 and 1.0. The default color for a label is black.

`delete()`
: Deletes a label.

10.6 The `light` module

`rotate`(*x, y, z*)
: Rotates a light. *x*, *y*, and *z* are the angles the light is to be rotated about the x, y, and z axis, respectively. These are integers and are measured in degrees.

`switch`(*status*)
: Switches a light on or off. *status* is one of the integer constants `ON` or `OFF` defined in the `viewmol` module.

10.7 The `molecule` module

`molecule()`
: Creates a new instance of a molecule object and returns a reference to it. Note: To obtain object references to molecules already loaded into VIEWMOL use the `getMolecules` function of the `viewmol` module.

`translate`(*x, y, z*)
: Translates (shifts) molecule by *x*, *y*, and *z* along the x, y, and z axis, respectively. *x*, *y*, and *z* have to be integers and are measured in pixels of the screen.

`rotate`(*x, y, z*)
: Rotates molecule by *x*, *y*, and *z* about x, y, and z axis, respectively. *x*, *y*, and *z* have to be integers and are measured in degrees.

`getSpectrum()`
: Creates a new instance of a spectrum object and returns a reference to it if there is spectral information associated with this molecule.

`getEnergyLevels()`
: Creates a new instance of an energy level object and returns a reference to it if there is information about energy levels associated with this molecule.

`getHistory()`
: Creates a new instance of a history object and returns a reference to it if there is information about the optimization history associated with this molecule.

`showForces`(*status*)
: Sets the display of forces for all molecules to *status*. *status* has to be one of the integer constants `ON` or `OFF` defined in the `viewmol` module.

`getAtoms()`
: Returns a list containing references to all atom objects the molecule is composed off.

`getBonds()`
: Returns a list of tupels describing all bonds in the molecule. The tupels consists of three integers `(atom1, atom2, order)` where `atom1` and `atom2` are the indices of the two atoms forming the bond and `order` is the bond order. The bond order can be one of the constants `HYDROGENBOND, CONJUGATED, SINGLE, DOUBLE`, or `TRIPLE` for hydrogen/Van der Waals, conjugated, single, double, and triple bonds, respectively. These constants are defined in the `molecule` module.

`getWavenumbers()`

Returns a list of tupels describing all wave numbers of the molecule. The tupel consists of four floats and one string `(waveNumber, IRIntensity, RamanIntensity, INSIntensity, symmetry)` where `waveNumber` is the wave number in cm^{-1}, `IRIntensity`, `RamanIntensity`, and `INSIntensity` are the IR, Raman, and inelastic neutron scattering intensities in per cent, and `symmetry` is a label describing the symmetry of the mode.

`getMOEnergies()`
Returns a list of tupels providing information about all molecular orbitals of the molecule. The tupel consists of two floats, one integer, and one string `(energy, occupation, spin, symmetry)` where `energy` is the energy of the molecular orbital in Hartrees, `occupation` is the number of electrons in this orbital, `spin` in one of the constants `ALPHA+BETA, ALPHA`, or `BETA` describing what spin has been assigned to this orbital, and `symmetry` is a label describing the symmetry of the molecular orbital. The constants are defined in the `molecule` module.

`title(` *[title]* `)`
Sets or returns the title of a molecule. *title* has to be a string. The maximum length of the title is limited to 255 characters.

`bondAverage(`*atom*`)`
Returns the average of the lengths of all bonds involving atom *atom* in Ångstrøms. *atom* is an atom object.

`bondLength(`*atom1, atom2, [length, unit]* `)`
Returns or sets the length of the bond between atoms *atom1* and *atom2*. *atom1* and *atom2* have to be atom objects, the bond length is returned in Ångstrøms. If *length* and *unit* are given, the bond length is set. *length* is a double, *unit* a string containing either `Ang`, `au` or `bohr`, or `pm` for Ångstrøms, atomic units, or picometers. Everything else is interpreted as Ångstrøms.

`bondAngle(`*atom1, atom2, atom3, [angle]* `)`
Returns or sets the bond angle *atom1–atom2–atom3*. *atom1*, *atom2*, *atom3* are atom objects. If *angle* is given the bond angle is set. *angle* has to be a double and is measured in degrees.

`torsionAngle(`*atom1, atoms2, atom3, atom4, [torsionAngle]* `)`
Returns or sets the torsion angle *atom1–atom2–atom3–atom4*. *atom1*, *atom2*, *atom3*, *atom4* are atom objects. If *torsionAngle* is given the torsion angle is set. *torsionAngle* has to be a double and is measured in degrees.

`getThermodynamics(`*property, type*`)`
Returns a thermodynamical property of the molecule. *property* and *type* are integers. *property* can be one of the integer constants `ENTHALPY`, `ENTROPY`, `GIBBS_ENERGY`, or `HEAT_CAPACITY` defined in the `molecule` module. *type* can be one of the integer constants `TRANSLATION`, `PV`, `ROTATION`, `VIBRATION`, or `TOTAL` also defined in the `molecule` module. The returned thermodynamic property will be in SI units.

`reaction(` *[side]* `)`
Sets or returns whether the molecule is a reactant or a product in a reaction. *side* is an integer and can be set to one of the integer constants `REACTANT`, `PRODUCT`, or `ALLREACTIONS` defined in the `molecule` module.

`showElectrons(`*type, [gridResolution, interpolation]* `)`

Displays wave function related properties of the molecule. *type* is an integer and can be set to one of the integer constants `BASIS_FUNCTION`, `BASIS_IN_MO`, `MO`, or `DENSITY` defined in the `molecule` module. *gridResolution* is a double specifying the resolution of the grid used to calculate the isosurface. Larger values for *gridResolution* result in smoother displays. *interpolation* is an optional integer and can be one of the integer constants `IP_NONE` (no interpolation), `IP_LINEAR` (linear interpolation), or `IP_LOG` (logarithmic interpolation) defined in the `molecule` module.

`showGrid(`*which, [interpolation]*`)`
: Displays a property which has been read as a grid. *which* is an integer between 1 and the number of grids which have been read for this molecule and identifies the grid to be shown. *interpolation* is an optional integer and can be one of the integer constants `IP_NONE` (no interpolation), `IP_LINEAR` (linear interpolation), or `IP_LOG` (logarithmic interpolation) defined in the `molecule` module.

`selectBasisfunction(`*atom, name, count*`)`
: Selects a basis function for display. *atom* is an atom object specifying which atom the basis function belongs to. *name* is a string specifying what kind of basis function (s, p, d, etc.) to select. *count* is an integer specifying which of the s, p, d, etc. functions to select. Assume atom 1 is a carbon atom in a calculation using a DZVP basis set. It therefore has three s functions. `selectBasisfunction(1, "s", 1)` would select the 1s function, `selectBasisfunction(1, "s", 2)` the first 2s function, and `selectBasisfunction(1, "s", 3)` the second 2s function.

`unitCell(`*visible, [afac, bfac, cfac]*`)`
: Sets visibility and number of replicas of unit cell. *visible* has to be one of the integer constants `ON` or `OFF` defined in the `viewmol` module to turn display of the unit cell on or off. *afac*, *bfac*, and *cfac* are doubles specifying the number of replicas of the unit cell to be displayed along the a, b, and c axis, respectively. Fractions are allowed for *afac*, *bfac*, and *cfac*.

`millerPlane(`*visible, [h, k, l]*`)`
: Sets visibility and orientation of Miller plane. *visible* has to be one of the integer constants `ON` or `OFF` defined in the `viewmol` module to turn display of a Miller plane on or off. *h*, *k*, and *l* are integers specifying the Miller indices of the plane to display.

`addAtom(`*symbol, [attach]*`)`
: Adds an atom to the molecule. *symbol* is a string containing the element symbol of the atom to add. *attach* is an atom object specifying the atom the newly added atom should be attached to. *attach* can be omitted, but this is only useful for adding the first atom to a molecule.

10.8 The `spectrum` module

`show()`
: Displays the spectrum window for a molecule.

`mode(`*[mode]*`)`
: Displays the mode specified or returns the mode displayed. *mode* has to be an integer between 1 and the maximum number of vibrational modes for this molecule.

`deselect()`
: Deselects modes previously selected with a call to `mode`.

`type(`*[type]*`)`

Sets or returns the type of the spectrum displayed. *type* has to be one of the integer constants `ALLMODES`, `IRMODES`, `RAMANMODES`, or `INSMODES` defined in the `spectrum` module to set the display to all modes, IR active modes, Raman active modes, or inelastic neutron scattering display.

`display(` *[type]* `)`
: Sets or returns the way normal modes are displayed. *type* has to be one of the integer constants `ANIMATE`, `ARROWS`, or `DISTORT` defined in the `spectrum` module to display normal modes animated, with arrows, or as distortion.

`style(` *[style]* `)`
: Sets or returns the display style of the spectrum. *style* has to be one of the integer constants `LINES` or `GAUSSIANS` defined in the `spectrum` module to set the display style to line spectrum or gaussian spectrum.

`amplitude(` *[amplitude]* `)`
: Sets or returns the amplitude of a normal mode. *amplitude* has to be a double.

`scaleFactor(` *[factor]* `)`
: Sets or returns the scale factor for the wave numbers in a spectrum. *factor* has to be a double.

`zoom(`*x1, y1, x2, y2*`)`
: Sets the zoom of the spectrum. *x1*, *y1*, *x2*, and *y2* are doubles specifying the minimum and maximum values of wave numbers and intensities, respectively, to be displayed. *x1* and *x2* as well as *y1* and *y2* cannot be equal.

`saveDrawing(`*format, filename*`)`
: Saves the spectrum to file. *format* has to be one of the constants `TIFF`, `PNG`, `HPGL`, or `POSTSCRIPT` defined in the `viewmol` module, *filename* the name of the file the drawing is to be saved to.

The spectrum window can be closed by deleting the `spectrum` object.

10.9 The `viewmol` module

`load(`*filename*`)`
: Loads a molecule into VIEWMOL. *filename* has to be a string containing the name (and path if necessary) of the file to load.

`save(`*molecule, filename, format*`)`
: Saves molecule *molecule* in the format *format* to file *filename*. *molecule* has to be a molecule object, *filename* a string giving the name of the file (including path, if appropriate) the molecule is to be saved to, and *format* a string describing the format in which the molecule is to be saved. *format* can be any of the strings given after the `output` keyword in `viewmolrc` (currently `car`, `arc`, `mol`, or `tm`).

`delete(`*molecule*`)`
: Deletes molecule *molecule*. *molecule* has to be a molecule object. Note: Molecules cannot be deleted using Python's `del` operator since Python works on VIEWMOL's data structures.

`getMolecules()`
: Returns a list of the molecules loaded into VIEWMOL.

`getLights()`
: Returns a list of the available lights.

`getLabels()`

Returns a list of all labels known to VIEWMOL.

`model` (*[model]*)
: Sets or returns the model used to display molecules. *model* has to be one of the integer constants `WIREMODEL`, `STICKMODEL`, `BALLMODEL`, or `CPKMODEL` defined in the `viewmol` module to set the model to wire model, stick model, ball-and-stick model, and CPK model, respectively.

`drawingMode` (*[mode]*)
: Sets or returns the drawing mode for molecules. *mode* has to be one of the integer constants `DOT`, `LINE`, or `SURFACE` defined in the `viewmol` module to set the drawing mode correspondingly.

`projection` (*[projection]*)
: Sets or returns the projection. *projection* has to be one of the integer constants `ORTHO` or `PERSPECTIVE` defined in the `viewmol` module.

`sphereResolution` (*[resolution]*)
: Sets or returns the resolution for spheres and cylinders. *resolution* has to be an integer. Higher resolutions result in smoother looking spheres and cylinders.

`lineWidth` (*[width]*)
: Sets or returns the line width for wire model displays. *width* has to be an integer. If *width* is set to zero the line width is calculated based on the size of the window.

`groundColor` (*[red, green, blue]*)
: Sets or returns the color of the ground displayed if the projection is set to `PERSPECTIVE`. *red*, *green*, and *blue* are floats specifying the red, green, and blue components of the ground color. They have to be between 0.0 and 1.0. If the ground color is retrieved, a tupel with the red, green, and blue values is returned.

`backgroundColor` (*[red, green, blue]*)
: Sets or returns the color of the background. *red*, *green*, and *blue* are floats specifying the red, green, and blue components of the background color. They have to be between 0.0 and 1.0. If the background color is retrieved, a tupel with the red, green, and blue values is returned.

`labelAtoms` (*status*)
: Specifies whether atoms should be labeled. *status* is an integer set to one of the constants `ON` or `OFF` defined in the `viewmol` module.

`saveDrawing` (*format, filename*)
: Saves the drawing to file. *format* has to be one of the constants `TIFF`, `PNG`, `HPGL`, `POSTSCRIPT`, or `RAYTRACER` defined in the `viewmol` module, *filename* the name of the file the drawing is to be saved to.

`isosurface` (*[level]*)
: Sets or returns which isosurface to display for wave function related drawings. *level* has to be a double.

`showThermodynamics` (*[select]*)
: Displays the thermodynamics dialog. *select* is an integer specifying which tab to display. This integer has to be either the integer constant `REACTION` defined in the `viewmol` module or an integer between 1 and the number of molecule loaded. In the first case the reaction page is displayed, in all other cases the page for the corresponding molecule is shown.

`redraw()`

Redraws the main window of VIEWMOL. Redraws are necessary to make changes visible performed using other methods of the `viewmol` module.

`getFramesPerSecond()`
: Returns the drawing speed of the last redraw of VIEWMOL's main window in frames per second.

`write(`*string*`)`
: Write *string* to the Python message dialog.

`registerMenuItem(`*moduleName*`)`
: Add *moduleName* to the "Run script" menu.

`showWarnings(`*warning*`)`
: Controls the display of warning messages. *warning* is an integer set to one of the constants `ON` or `OFF` defined in the `viewmol` module. If warnings are turned off no dialog boxes requiring user interaction are displayed while a Python script is running.

`getWindowSize()`
: Returns a tupel containing the size of VIEWMOL's main window in pixels. The tupel consists of *(width, height)*.

`quit()`
: Quits VIEWMOL.

10.10 Installing Python scripts in the "Run script" menu

Python scripts can be installed as menu entries in the "Run script" submenu. For this to work the Python script needs to implement a class with the same name as the script. This class needs to provide at least the methods `register` and `run`. The `register` method is passed a language setting as abbreviation of a locale (currently `en_US`, `fr`, `de`, `hu`, `pl`, `ru`, or `es`). The method needs to call the `registerMenuItem` function implemented in the `viewmol` module to pass a string back which will appear as the identification of this script in the submenu.

The `run` method does not take any argument, but is the entry point for the script when the user selects the corresponding menu item in the "Run script" submenu.

A minimal Python script which will install itself in the "Run script" menu would be the following (available at `$VIEWMOLPATH/scripts/template.py`):

```
import viewmol

class template:
  def run(self):
    print "This is a script template which just prints this silly message."

  def register(self, language):
    viewmol.registerMenuItem("Script template")

# The following three lines are not necessary for running this script
# from the "Run script" menu, but allow the script to also be run
# standalone
```

```
if __name__ == '__main__':
  script=template()
  script.run()
```

All scripts which are to appear in the “Run script” menu have to be installed in the `scripts` subdirectory under `VIEWMOLPATH`.

11 Adapting the Program to a Different Language

VIEWMOL has been written to take full advantage of the language independence of the X Window System. All program messages, menus, dialog box texts etc. are stored outside of the program in resource files. Therefore it is possible that different users can run the same VIEWMOL executable in different languages on the same computer. Currently, eight languages are supported: English, French[5], German, Hungarian[6], Polish[7], Russian[8], Spanish[9], and (partly) Turkish[10]. To adapt VIEWMOL to another language only the `Viewmol` file in `$VIEWMOLPATH/locale/en_US` has to be translated and copied into the corresponding language directory under `$VIEWMOLPATH/locale`. For a native speaker of the language this will take between 45 and 60 minutes.

The `Viewmol` files provided with VIEWMOL have three sections. The first section is related to the installation of external support programs, the second section contains default settings (see p. 46). The third section contains all language dependent messages and this is the only section which needs translation. To translate, translate everything right of the colon. The strings '%s', '%d', or '%f' mark the position of names or numbers which are filled in by the program and must remain in the translated version at the appropriate position. After translation install the new `Viewmol` file as described in the Installation section (cf. p. 2) and VIEWMOL will talk in your language. The author would appreciate to get a copy of the translated resource file for inclusion in the next public release.

The internationalization of Python scripts is handled using the `gettext` facility available with Python. Messages for Python scripts are stored in the `$VIEWMOLPATH/locale/X/LC_MESSAGES/Viewmol.po` file where `X` is an abbreviation indicating the language. All lines starting with `msgstr` need to be translated. After the translation the corresponding binary files can be created by executing `make translations` in the `$VIEWMOLPATH/source` directory.

12 The making of multimedia files

If normal modes or the optimization history are animated and the user selects `Save drawing/TIFF` or `Save drawing/PNG` from the main window menu a series of TIFF or PNG files is written out, one for each frame of the animation (currently 20 frames for normal modes which cannot be changed by the

[5]Many thanks to Ludovic Douillard (douillard@DRECAM.cea.fr).

[6]Many thanks to Gábor Magyarfalvi (gmagyarf@para.chem.elte.hu).

[7]Many thanks to Nikodem Kuznik (nikodem@zeus.polsl.gliwice.pl).

[8]My Russian is a little bit rusty. Apologies for any grammar mistakes and/or unrecognizable meanings. I would appreciate a check by a native speaker.

[9]Many thanks to Jose R. Valverde (jrvalverde@cnb.uam.es).

[10]Many thanks to Murat Guner (muratguner1982@hotmail.com).

user). These TIFF or PNG files can easily be converted to a video file (MPEG) showing the animation using standard image manipulation tools from the Internet. One possible MPEG encoder is `mpeg_encode` which is available from mm-ftp.CS.Berkeley.EDU via anonymous ftp. This encoder expects its input files either in PPM, PNM, or YUV format. To convert the TIFF files written by VIEWMOL you can use the PBMPLUS or NETPBM libraries which have a filter tifftopnm (you also need pnmflip, since tifftopnm changes the orientation of the picture). The following shell script will do the conversion (for sh and ksh users) if the default files from VIEWMOL have been used:

```
for i in vm_image*.tiff
do
  j=`basename $i tiff`pnm
  tifftopnm $i | pnmflip -topbottom > $j
done
```

The resulting PNM files can then be processed by `mpeg_encode` to produce a MPEG file which can, e. g., be included in a World Wide Web document.

Selecting `Save drawing/Povray` from the main window menu with an animation running will write a series of input files for POVRAY. These files can also be processed by POVRAY and used to generate a movie of the animation. This process can, however, be very time consuming.

13 Data files

VIEWMOL uses a data file named `viewmolrc` for getting information about atoms and available input and output filters. There may be three of these files. VIEWMOL looks at first in the current directory for this file, then in the users HOME directory for a file `.viewmolrc` and finally in the directory where the environment variable `VIEWMOLPATH` points to. In one of these three locations such a file must be found. The file should contain the following data:

- Lines of the format:
  option <name of option> <name of input filter> [<command line options for input filter>] "<characteristic string>"
  These lines define the input filters for VIEWMOL and the command line options connected with them (i. e. you can change the command line option if you want). <option> is the command line option VIEWMOL expects on its command line or the word `default`. The input filter connected with `default` is used if no command line option is passed to VIEWMOL. If no default input filter is specified VIEWMOL displays a file selection box. <name of input filter> is the path to and name of the input filter executable. If the input filter requires command line option (e. g. a file name) they can be specified after the name of the input filter. `'%s'` is used as a placeholder for file names. The path or name of the input filter can contain environment variables or the string `$OSNAME`. The latter is replaced by the subdirectory name for the machine VIEWMOL is currently running on. "<characteristic string>" is a string which is used to identify the type of a particular file. The first 1024 characters of an input file passed to VIEWMOL are scanned for this string and the input filter connected with the string is then used to read the file. Therefore these strings have to be unique for each input filter and have to be in every file of a certain type within the first 1024 characters. Since most programs write their names out at the beginning these restrictions seem to be no problem.

- Lines of the format:
  output <reference to resources> <name of output filter> [<command line options for output filter>] `'%s'`
  These lines define the output filters for VIEWMOL. <reference to resources> is an arbitrary string which must refer to a resource in the `$VIEWMOLPATH/locale/en_US/Viewmol` file. This string is used to provide the label for the output filter in the output filter selection box. <name of output filter> is the path to and name of the output filter executable. All output filters should at least accept the name of the output file from their command lines. If additional parameters are required they can also be specified after the name of the output filter. `'%s'` is used as a placeholder for the output file name. Environment variables or the string `$OSNAME` can be used in the same way as for input filters.

- Lines of the format:
  <symbol> <rad> <rd> <gd> <bd> <rl> <gl> <bl> <surface>
  <symbol> is an atomic symbol, <rad> is the Van der Waals radius of this atom in Ångstrøms, <rd>, <gd> and <bd> are the red, green and blue color for the darkest part of this atom and <rl>, <gl> and <bl> are the red, green and blue color for the lightest part of the atom. There are five reserved strings for <symbol>. If <symbol> is `bd` the following description describes a hydrogen bond. The <rad> field is also interpreted as the radius of all bond sticks. All other fields are only applied to hydrogen bonds. `uc` specifies a unit cell corner. Radius and color given here affect the appearance of the unit cell. `ps` and `ms` specify the surface properties for the positive and negative isosurface, respectively, used to draw wave function related topics. In these cases the radius is not used. `??` specifies the properties for atoms where the atomic symbol could not be found. <surface> is an optional specification for the surface used when stick, ball, or CPK drawing with surfaces is activated. <surface> is a list of one or more of the following options

  – emission <r> <g> <b>
    The emission color of the surface. Using this option causes the surface to emit light. <r>, <g>, and <b> are the red, green and blue components for the light color.
  – ambient <r> <g> <b>
    The ambient light which is reflected by the surface. <r>, <g>, and <b> are the red, green and blue components for the light color.
  – specular <r> <g> <b>
    The specular light which is reflected by the surface. <r>, <g>, and <b> are the red, green and blue components for the light color.
  – shininess <n>
    A parameter which determines the kind of reflection. <n> can be in the range 0 ... 128.
  – alpha <n>
    This parameter determines the transparency of the surface. 0.0 indicates an opaque surface, 1.0 a fully transparent one.

  All color specifications can be between 0.0 and 1.0. The total length of a line specifying an atom is restricted to 132 characters. The keywords for the surface specifications can be abbreviated with the first two letters.

Any line starting with '#' is treated as a comment.

VIEWMOL makes extensive use of X Windows resources. All standard search algorithms for the location of the resources apply (see e. g. O'Reilly books on the X Window System). VIEWMOL has English resources compiled in. Resources for other languages are provided in the directory `$VIEWMOLPATH/locale` and might be installed as described in the installation section of this manual (p. 2).

The following resources are used to specify the defaults. They can be overwritten in the user's `$HOME/.Xdefaults` file or, in part, by the configuration options available in the program. Defaults configurable from within the program are marked with an asterisk (*).

```
Viewmol*geometry:                               500x500+50+50  (*)
Viewmol.history.geometry:                       500x250+50+590 (*)
Viewmol.spectrum.geometry:                      500x250+50+590 (*)
Viewmol.MODiagram.geometry:                     250x500+565+50 (*)
Viewmol.Bell:                                         <no default>
Viewmol.model:                                            wire (*)
Viewmol.drawingMode:                                   surface (*)
Viewmol.bondType:                                   conjugated (*)
Viewmol.sphereResolution:                                   20 (*)
Viewmol.lineWidth:                                           0 (*)
Viewmol.simplifyWhileRotating:                           False (*)
Viewmol.interpolation:                                  linear (*)
Viewmol.bondLength:                                      %7.4f Ang
Viewmol.bondAngle:                                       %7.2f deg
Viewmol.torsionAngle:                                    %7.2f deg
Viewmol.wavenumbers:                                        0:5000
Viewmol.isosurface:                                       0.05 (*)
Viewmol.densityResolution:                                0.01 (*)
Viewmol.reservedColors:                                          0
Viewmol.hydrogenBondThreshold:                             2.0 (*)
Viewmol.automaticRecalculation:                          False (*)
Viewmol.thermoUnits:                                    joules (*)
Viewmol*spectrumForm*amplitudeSlider.decimalPoints:              2
Viewmol*spectrumForm*amplitudeSlider.minimum:                 -250
Viewmol*spectrumForm*amplitudeSlider.maximum:                  250
Viewmol*spectrumForm*scaleSlider.decimalPoints:                  2
Viewmol*spectrumForm*scaleSlider.minimum:                       50
Viewmol*spectrumForm*scaleSlider.maximum:                      150
Viewmol*thermoForm*pressureSlider.decimalPoints:                 2
Viewmol*thermoForm*pressureSlider.minimum:                       1
Viewmol*thermoForm*pressureSlider.maximum:                    1000
Viewmol*wavefunctionForm*level.decimalPoints:                    3
Viewmol*wavefunctionForm*level.minimum:                          1
Viewmol*wavefunctionForm*level.maximum:                        100
Viewmol*wavefunctionForm*grid.minimum:                           4
Viewmol*wavefunctionForm*grid.maximum:                          40
Viewmol*wavefunctionForm*grid.value:                            20
Viewmol*MODiagramForm*resolution.minimum:                        1
Viewmol*MODiagramForm*resolution.maximum:                     1000
```

```
Viewmol*MODiagramForm*resolution.decimalPoints:                      3
Viewmol*MODiagramForm*resolution.value:                             10
Viewmol*unitcellForm*avalue.minimum:                                10
Viewmol*unitcellForm*avalue.maximum:                                50
Viewmol*unitcellForm*avalue.decimalPoints:                           1
Viewmol*unitcellForm*bvalue.minimum:                                10
Viewmol*unitcellForm*bvalue.maximum:                                50
Viewmol*unitcellForm*bvalue.decimalPoints:                           1
Viewmol*unitcellForm*cvalue.minimum:                                10
Viewmol*unitcellForm*cvalue.maximum:                                50
Viewmol*unitcellForm*cvalue.decimalPoints:                           1
Viewmol*unitcellForm*hvalue.minimum:                                -5
Viewmol*unitcellForm*hvalue.maximum:                                 5
Viewmol*unitcellForm*kvalue.minimum:                                -5
Viewmol*unitcellForm*kvalue.maximum:                                 5
Viewmol*unitcellForm*lvalue.minimum:                                -5
Viewmol*unitcellForm*lvalue.maximum:                                 5
Viewmol*bondForm*thresholdSlider.minimum:                          100
Viewmol*bondForm*thresholdSlider.maximum:                          250
Viewmol*bondForm*thresholdSlider.decimalPoints:                      2
Viewmol*bondForm*scaleRadius.minimum:                                1
Viewmol*bondForm*scaleRadius.maximum:                              200
Viewmol*bondForm*scaleRadius.decimalPoints:                          2
Viewmol*infoForm*text*rows:                                          6
Viewmol*infoForm*text*columns:                                      80
Viewmol.paperSize:                                              A4 (*)
Viewmol.elementSortOrder:                                    C,H,N,O,S
Viewmol.viewer*font:                                          variable
Viewmol.spectrum*font:                                        variable
Viewmol.history*font:                                         variable
Viewmol.MODiagram*font:                                       variable
Viewmol.viewer.background:                                   white (*)
Viewmol.viewer.foreground:                                 gray75 (*)
Viewmol.viewer.delay:                                                0
Viewmol*spectrum.spectrum.background:                        white (*)
Viewmol*spectrum.spectrum.foreground:                        black (*)
Viewmol*history.history.background:                          white (*)
Viewmol*history.history.foreground:                          blue  (*)
Viewmol*MODiagram.MODiagram.background:                      white (*)
Viewmol*MODiagram.MODiagram.foreground:                      black (*)
Viewmol.MODiagram.MODiagram.greekFont:
                                   -adobe-symbol-medium-r-normal--14-*
Viewmol*foreground:                                          black (*)
```

The `Viewmol.Bell` resource is the only resource which does not have a default. As long as this resource is not set the standard keyboard bell is rung as soon as a selection in one of the windows is made by mouse click. This resource can be set to the name (and command line parameters) of any program which shall be run instead, preferably one which produces a nicer sound effect.

The `Viewmol.model` resource can be set to wire, stick, ball, or cpk. The `Viewmol.drawingMode` resource can be set to dot, line, or surface. The `Viewmol.bondType` resource can be set to single, multiple, or conjugated. The `Viewmol.interpolation` resource can be set to none, linear, or logarithmic. The resources for specifying formats for bond lengths, bond angles, and torsion angles have to contain a valid C format string for printing a floating point number. The resource for the bond lengths recognizes Ang, pm, bohr, and au in the format string as units and converts the bond lengths accordingly. The `Viewmol.reservedColors` resource can be used to limited the number of colors allocated by VIEWMOL if it runs in colormap mode. VIEWMOL tries to allocate as much colors as it can. This might interfere with others program. In this case `Viewmol.reservedColors` can be used to tell VIEWMOL to leave the specified number of colors unallocated. The `Viewmol.thermoUnits` resource can be set to joules, calories, or thermochemical calories. In case of the specifications for the sliders the values given for minimum and maximum have to be multiplied by $10^{decimalPoints}$. I. e., if the number of decimals is to be changed also minimum and maximum have to be changed. Paper sizes currently recognized are A5, A4, A3, Letter, Legal, and <width> x <height> where <width> and <height> are in millimeters.

The `Viewmol.viewer.foreground` resource is used for the color of the ground if perspective drawing is enabled. The `Viewmol.viewer.delay` resource can be set to the delay for an animation. If this resource is set to a value larger than zero VIEWMOL pauses the given number of microseconds after drawing a new frame of an animation.

14 Programming Your Own Input Filter

VIEWMOL can be easily adapted to read outputs of other programs or other file formats. All you have to do is to write a new input filter which extracts the data from the corresponding file. These input filters are stand-alone programs and can be written in every programming language you want. Examples in C and awk are included.

The input filter has to read the following data from the output file and write them to its standard output in the format described below. This format follows the file format of TURBOMOLE very closely. A few sections had to be extended to allow data which is currently not supported by TURBOMOLE (e. g. unit cells).

- the cartesian coordinates and atom symbols (required)
  Write to standard output in the following format:

  ```
  $coord factor
   x1   y1   z1   symbol1   xyz
   x2   y2   z2   symbol2   xyz
   ...
  ```

  `factor` is the conversion factor the coordinates have to be multiplied with to convert them to Ångstrøms. Any combination of x, y, and z at the end of the line (optional) indicates that the corresponding atom has been kept fixed in that direction during a geometry optimization. Consequently, VIEWMOL will not draw the forces acting on this atom in the fixed direction.

- the title (optional)
  Write to standard output in the following format:

```
$title
title
```

- the wave numbers and intensities (optional)
  Write to standard output in the following format:

  ```
  $vibrational spectrum
   symmetry1  wavenumber1  IR-intensity1  Raman-intensity1
   symmetry2  wavenumber2  IR-intensity2  Raman-intensity2
   ...
  ```

  `symmetry` is the symmetry label for the vibrational mode, `wavenumber` is its wave number and `IR-intensity` and `Raman-intensity` are its IR and Raman intensity, respectively. If the symmetry labels for the vibrational modes are unknown they should be set to a default (e. g. A1).

- normal coordinates (optional)
  Write to standard output in the following format:

  ```
  $vibrational normal modes
   i1   i2  nm(1,1)  nm(2,1)  nm(3,1)  nm(4,1)  nm(5,1)
   i1   i2  nm(6,1)  ...      nm(3*natom,1)
   i1   i2  nm(1,2)  nm(2,2)  nm(3,2)  nm(4,2)  nm(5,2)
   i1   i2  nm(6,2)  ...      nm(3*natom,2)
   ...
   i1   i2  nm(1,nmodes)      ...               nm(5,nmodes)
   i1   i2  nm(6,nmodes) ...  nm(3*natom,nmodes)
  ```

  `i1` and `i2` are integers which are skipped during reading. `nm(i,j)` are the normal mode coefficients. They have to be provided ordered by cartesian coordinates (all x components of the first atom first, then all y components of the first atom etc.).

- optimization history or MD trajectory (optional)
  Write to standard output in the following format:

  ```
  $grad factor
    cycle = nc SCF energy = E_nc |dE/dxyz| = gradnorm_nc
    [unitcell a b c alpha beta gamma]
    [unitcell vectors
     xa ya za
     xb yb zb
     xc yc zc]
    x1  y1  z1   symbol1
    x2  y2  z2   symbol2
    ...
    xn  yn  zn   symboln
    gx1 gy1 gz1
    gx2 gy2 gz2
    ...
  ```

```
  gxn gyn gzn
  cycle = nc+1 SCF energy = E_nc+1 |dE/dxyz| = gradnorm_nc+1
  ...
```

`factor` is the conversion factor the coordinates have to be multiplied with to convert them to Ångstrøms. `nc` is a counter for the cycle, `E_nc` is the energy for the configuration of cycle nc, and `gradnorm_nc` is the gradient norm of cycle nc. The line starting with `unitcell` is optional and can be used to specify the current unit cell, e. g. during a constant pressure MD run. Unit cells can be specified either by providing the lengths of the edges and the angles between them or by providing the three vectors which span the unit cell. The `x`, `y`, and `z` are the cartesian coordinates for each atom, `symbol` is the atomic symbol. The `gx`, `gy`, and `gz` are the gradients for each atom. This structure can be repeated for as many cycles as necessary.

- MO energies and coefficients (optional)
  Write to standard output in the following format for closed shell systems:

```
$scfmo [symmetrized] [gaussian]
     n symmetry_label_n eigenvalue=MO_E_n nsaos=norb
moc(n,1) moc(n,2) moc(n,3) moc(n,4)
moc(n,5) ...      moc(n,norb)
     n+1 symmetry_label_n+1 eigenvalue=MO_E_n+1 nsaos=norb
...
```

  or for open shell systems:

```
$uhfmo_alpha [symmetrized] [gaussian]
     n symmetry_label_n eigenvalue=MO_E_n nsaos=norb
moc(n,1) moc(n,2) moc(n,3) moc(n,4)
moc(n,5) ...      moc(n,norb)
     n+1 symmetry_label_n+1 eigenvalue=MO_E_n+1 nsaos=norb
...

$uhfmo_beta [symmetrized] [gaussian]
     n symmetry_label_n eigenvalue=MO_E_n nsaos=norb
moc(n,1) moc(n,2) moc(n,3) moc(n,4)
moc(n,5) ...      moc(n,norb)
     n+1 symmetry_label_n+1 eigenvalue=MO_E_n+1 nsaos=norb
...
```

  The string `symmetrized` is optional and can be used to notify VIEWMOL of the fact that the MO coefficients are with respect to symmetrized AOs rather than with respect to AOs. VIEWMOL needs moloch from the TURBOMOLE package to handle symmetrized AOs. If moloch is not installed and symmetrized AOs are input, MOs and electron densities cannot be drawn. The string `gaussian` is also optional and notifies VIEWMOL that the MO coefficients are normalized and ordered GAUSSIAN style. `n` is a counter counting the MOs, `symmetry_label_n` is the symmetry label for MO n, `MO_E_n` is the MO energy for MO n, and `norb` is the total number of orbitals. The `moc(n,i)` are the MO coefficients for MO n.

- basis functions and occupation numbers (optional)
  Write to standard output in the following format:

```
$atoms
atom_symbol1 list_of_indices1 \
  basis=basis_set_name1
atom_symbol2 list_of_indices2 \
  basis=basis_set_name2
...
$basis
*
basis_set_name1
*
  number_of_primitives  angular_momentum
  exponent1   coefficient1
  exponent2   coefficient2
  ...
  exponentn   coefficientn
  number_of_primitives  angular_momentum
  ...
*
basis_set_name2
*
  ...
*
$closed shells
  symmetry_label    list_of_indices   (2)
$alpha shells
  symmetry_label    list_of_indices   (1)
$beta shells
  symmetry_label    list_of_indices   (1)
$pople   [6d/10f/15g]
```

  `atom_symbol` is the atom symbol of an element and `list_of_indices` contains the indices of all atoms of the particular element according to the list of coordinates read in under `$coord`. The list can be either comma separated and/or contain hyphens for indicating ranges (e. g. c 1,3,7-10 is a valid descriptor). `Basis_set_name` can be an arbitrary string describing a particular basis set. It is only used to find the corresponding basis set in the list read under `basis`. This list simply states the name for a basis set and then lists the primitive functions which make up a contracted Gaussians starting with the number of primitives in that particular contracted Gaussian and its angular momentum (s, p, d, f, ...). Than the exponents and contraction coefficients are listed line by line. This is repeated for all contracted Gaussians of that particular basis set. `$closed shells`, `$alpha shells`, and `$beta shells` are used to tell VIEWMOL which MOs are occupied with how many electrons. `symmetry_label` is the symmetry label for a number of MOs and `list_of_indices` is a list of integers stating which of the MOs of that particular symmetry are occupied by either one or two electron(s). This list can be either comma-separated or contain hyphens to indicate ranges of MOs. **Note:** `$closed shells`, `$alpha shells`, and `$beta shells` have to appear after `$scfmo` in the output written by the input filter. `$pople` is used to indicate that d, f, or g functions have 6, 10, or 15 components instead of 5, 7, or 9. **Note:** This data group has to appear after the `$coord` or `$grad` in the output. Otherwise VIEWMOL will fail.

- grid files
  Write to standard output in the following form:

  ```
  $grid #n
  origin x y z
  vector1 x y z
  vector2 x y z
  vector3 x y z
  grid1 start s delta d points np
  grid2 start s delta d points np
  grid3 start s delta d points np
  type ty
  title for this grid
  t
  plotdata
  d(1,1,1) d(1,1,2) d(1,1,n) ... d(1,2,1)
  ... d(1,n,n) d(2,1,1) ... d(n,n,n)
  ```

  here `n` is an integer identifying the grid. `origin` is used to specify the x, y, and z coordinates of the origin of the grid. `vector1`, `vector2`, and `vector3` are used to specify the three vectors spanning the grid. `grid1`, `grid2`, and `grid3` are used to specify the starting point, `s`, the step size, `d`, and the number of points, `np`, on each of the three vectors spanning the grid. `ty` can be either `mo` or `density` specifying whether the data represents a molecular orbital or a density. `t` is a string giving the grid a title which is used in the wave function dialog to allow the user to select the grid. Finally, `d(i,j,k)` are the values for the property at each grid point.

- the unit cell (optional)
  Write to standard output in one of the following forms:

  ```
  $unitcell    a    b    c   alpha   beta   gamma
  ```

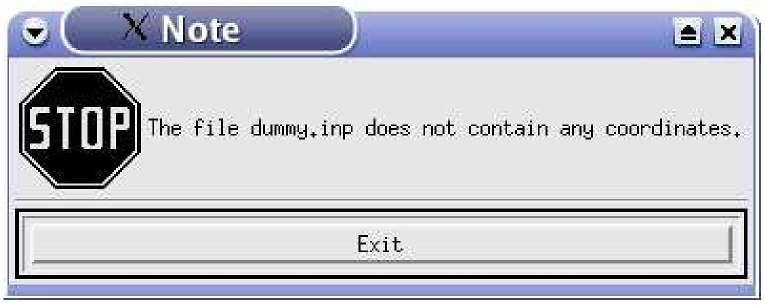
  or

  ```
  $unitcell vectors
  xa ya za
  xb yb zb
  xc yc zc
  ```

  where each row contains the components of one of the three vectors spanning the unit cell (this is also known as the Bravais matrix).

- errors occuring during file processing (optional)
  Write to standard output in the following form:

  ```
  $error errorLabel severity additionalInformation
  ```



```
The file dummy.inp does not contain any coordinates.
```

Figure 17: The error dialog produced by the sample error message

errorLabel is an arbitrary one word label which refers to an error message in the resources. severity is a label for the severity of the error. Set it to 0 if the program can continue despite this error. Set it to 1 if the program must stop. additionalInformation is any additional information you want to be displayed in the error message (e. g. the name of a file which was not found). Currently, the following errorLabels are in use: noFile, notConverged, unsupportedVersion, wrongFiletype, noCoordinates, noEnergy, and unknownErrorMessage. If your input filter wants to return an error because it is missing coordinates in the input file "dummy.inp" you can have it writing the following line to standard output:

```
$error missingCoordinates 1 dummy.inp
```

Then you have to specify a resource for the error message in $HOME/.Xdefaults:

```
Viewmol.missingCoordinates: The file %s does not
contain any coordinates.
```

With these two lines in place any encounter of no coordinates in an input file will lead to the display of the error dialog in Figure 17. There is no need to recompile VIEWMOL to achieve this.

The last line of the data written to standard output by the input filter must be $end.

The input filter can be installed by adding a line to the viewmolrc file.

15 Programming Your Own Output Filter

VIEWMOL can be easily adapted to write files in any format. All you have to do is to write a new output filter which formats the data provided by VIEWMOL. These output filters are stand-alone programs and can be written in every programming language you want. Examples in awk are included.

The output filter has to accept the following data from its standard input and write them to a file whose name is given as a command line parameter to the filter. VIEWMOL passes the following data groups to the output filter:

- the unit cell (if present) is sent in the following format (a, b, and c in atomic units, the angle in degrees)

```
$unitcell a b c alpha beta gamma
```

- the cartesian coordinates are sent in the following format (in atomic units)

  ```
  $coord
   x1   y1   z1   symbol1
   x2   y2   z2   symbol2
   ...
  ```

- the bond information is sent in the following format

  ```
  $bonds
   atom1 atom2 bond_order
   ...
  ```

  where `atom1` and `atom2` are the numbers of the atoms according to the list in `$coord` which form the bond. `bond_order` is the actual order of the bond, -2 if the bond is part of a conjugated system, or -1 if it is a hydrogen bond.

`$end` is passed to the output filter as last line.

16 Test Scripts

VIEWMOL now contains a number of test scripts which can be used to check the correct functioning of the software. These test scripts are located in the `tests` subdirectory. The following scripts are available

- autotest.py
  This script executes all the functionality of VIEWMOL which is accessible through Python. It requires the examples in the `examples` subdirectory. To start the script call it from the "Run script/Select ..." menu item.

- importtest.py
  This script tests the functionality of the Gamess and Turbomole input filters by automatically loading all examples which come with either Gamess or Turbomole. It requires Tix/Tkinter and the examples from Gamess or Turbomole. The examples can be installed in an arbitrary directory. To start the script call it from the "Run script/Select ..." menu item.

17 Limitations

VIEWMOL currently cannot handle GAUSSIAN outputs which contain cartesian f functions (10f).

If VIEWMOL runs in color map (that should only happen on IBM RS6000 with Sabine graphics adapters if anybody still has one of these, in this case the background of the title screen has a constant color instead of the usual dark top and lighter bottom) shadows are not drawn if the drawing mode is "with surface".

On some graphics boards VIEWMOL will not be able to get a stencil buffer (notably Intel graphics chips). VIEWMOL will silently continue without the stencil buffer, but this will result in artefacts in the shadows drawn.

If TIFF or PNG files are saved make sure no other window (including dialog boxes) overlaps with the window to be saved. The information is read from the screen and overlapping windows might show up in the saved file (this is hardware dependent).

HPGL outputs for drawings which are labeled with non-latin characters will not contain any labels. HPGL output has only support for German umlauts, Postscript provides all ISO-8859-1 and KOI-8 characters.

If VIEWMOL runs on FreeBSD there are problems with saving molecules in any format. The output filters never return and therefore VIEWMOL seems to hang. Pressing Ctrl-C in the terminal VIEWMOL was started from terminates the output filter and makes VIEWMOL work again, but no meaningful information is put into the output file. A work-around for this problem is currently not known.

VIEWMOL is usable with Lesstif ($>$ 0.81). There are, however, some glitches, e. g. shortcuts don't work.

18 Frequently asked questions

1. VIEWMOL on Linux reports on start up:
   `viewmol: can't load library 'libMesaGLU.so.3'`
   or another library.
   VIEWMOL does not find a dynamical linked library it needs. The reason for this is that either the library is not installed, the wrong version is installed, or the dynamic linker is not set up to find this library. VIEWMOL needs the following dynamic libraries:

   ```
   libtiff.so.3
   libpng12.so.0
   libz.so.1
   libGLU.so.1
   libGL.so.1
   libXm.so.3
   libXmu.so.6
   libXp.so.6
   libXi.so.6
   libXext.so.6
   libXt.so.6
   libX11.so.6
   libpthread.so.0
   libutil.so.1
   libdl.so.2
   libm.so.6
   libc.so.6
   libjpeg.so.62
   libstdc++.so.5
   libgcc_s.so.1
   libGLcore.so.1
   libSM.so.6
   libICE.so.6
   /lib/ld-linux.so.2
   ```

These libraries can normally be found in `/lib`, `/usr/lib`, and `/usr/X11R6/lib`. The dynamic linker checks the major version number and will refuse any library where the major version number does not match. The minor version number does not matter. The dynamic linker has to be set up to search the directories which contain these libraries. This is done in the file `/etc/ld.so.conf`. After modifying this file run `ldconfig -v` as root. Alternatively, the environment variable `LD_LIBRARY_PATH` can be set to point to these directories.

2. VIEWMOL on Linux reports on start up:
   ```
   viewmol: Symbol 'jpeg_resync_to_restart' is not
   defined.
   ```
   There are two different versions of `libtiff.so` distributed with different Linux distributions. One contains jpeg code (Debian) the other doesn't (RedHat). VIEWMOL has now been linked with the version which does not contain jpeg code so that this error will probably not occur anymore. If this error occurs only a recompilation will help. Please notify the maker of your Linux distribution so that they can make their distribution compatible.

3. When I try to recompile VIEWMOL I get a lot of error messages:

   ```
   cc -c -Wall -DLINUX -I/usr/compat/linux/usr/include/GL/
   -I/usr/compat/linux/usr/include/gr/ -O6 -m486
   -fomit-frame-pointer
   ../annotate.c
   ../annotate.c:19: X11/StringDefs.h: No such file or directory
   ../annotate.c:20: X11/cursorfont.h: No such file or directory
   ../annotate.c:21: Xm/Xm.h: No such file or directory
   ../annotate.c:22: Xm/Text.h: No such file or directory
   In file included from ../annotate.c:24:
   ../viewmol.h:20: X11/Intrinsic.h: No such file or directory
   ../viewmol.h:21: GL/gl.h: No such file or directory
   ../viewmol.h:22: GL/glx.h: No such file or directory
   *** Error code 1
   ```

   To recompile VIEWMOL you need to install the X Window System and OpenGL development environments which, in most Linux distributions, are separate packages. In this case you are missing all X Window System and OpenGL header files. You also need the development environment of lesstif or Motif for all header files in `/usr/include/Xm`.

4. While trying to recompile VIEWMOL the link step fails with:

   ```
   ld:
   Unresolved:  __eprintf

   *** Error code 1 (bu21)
   ```

   or a similar message referring to `__eprintf` (encountered on SGIs and IBMs so far only). There is a problem with the TIFF library you are linking with. If you have built the library yourself make sure it was built on the same machine as where you try to link VIEWMOL. If you have been trying to use a vendor supplied version of the TIFF library try to download and compile the library yourself.

5. Parts of one or more of VIEWMOL windows are not drawn while VIEWMOL runs on a Linux system with a Nvidia graphics card:
   Some of the Nvidia drivers have this problem if full-scene antialiasing is enabled. Set the environment variable `__GL_FSAA_MODE` to 0 before starting VIEWMOL to disable full-scene antialiasing and see if the problem disappers.

19 History, Authors, and Contributors

VIEWMOL started its life somewhere in 1991 as a tool to draw IR and Raman spectra from Turbomole outputs. Since drawing only spectra soon turned out to be insufficient for writing a PhD thesis, capabilities were added to draw the molecule and animate normal modes. Since other people in the Arbeitsgruppe Quantenchemie at Humboldt University in Berlin got interested in the program and wanted extensions for other program's output Andreas Bünger (a then 16 year old high school student on a practical course in the group) and Andreas Bleiber started to write an input filter for GAUSSIAN 9X. Arne Dummer wrote a filter for DMOL. In the course of the research performed in the group other capabilities were asked for and added by the original authors and by Mariann Krossner (calculation of inelastic neutron scattering intensities) and Andries de Man (extension of GAUSSIAN 9X input filter to read density functional outputs). Version 1.2 was presented at the German/Austrian Academic Software Award competition in 1993 and honored as outstanding achievement.

With the advent of Linux it was recognized that the original Fortran/IrisGL version would be difficult to port to more affordable hardware. Version 2.0 of the program was a complete rewrite in C/OpenGL by Jörg-Rüdiger Hill now mainly done on a Linux system. Development of the program continues on Linux.

Contributions, mainly in form of bug reports, code snippets, and enhancement requests have come from a number of people. In no particular order (and hopefully without forgetting somebody) I want to thank:

- George P. Ford (gford@smu.edu)
- Dan Moenster Jensen (Dan.M.Jensen@uni-c.dk)
- John Nicholas (jb_nicholas@pnl.gov)
- Konrad Hinsen (hinsen@ibs.ibs.fr)
- Marc Pedulla (pedulla+@pitt.edu)
- Stanislav Bohm (Stanislav.Bohm@vscht.cz)
- Rinaldo Poli (poli@u-bourgogne.fr)
- Martin Brändle (braendle@inorg.chem.ethz.ch)
- Martin G. Schütz (schuetz@theochem.uni-stuttgart.de)
- Ödön Farkas (farkas@para.chem.elte.hu)
- Peter Pulay (pulay@comp.uark.edu)
- M. Fabien Gutierrez (gutierre@irsamc1.ups-tlse.fr)
- Eric I. Arnoth (earnoth@UDel.Edu)

- Pedro A. M. Vazquez (vazquez@iqm.unicamp.br)
- Keith Refson (Keith.Refson@earth.ox.ac.uk)
- Pablo Vitoria Garcia (qibvigap@lg.ehu.es)
- Andrew Dalke (dalke@bioreason.com)
- Marcus Gastreich (ghost@pcgate.thch.uni-bonn.de)
- Frank Schneider (uzs93a@uni-bonn.de)
- Stephen P. Molnar (smolnar@jadeinc.com)
- Michael Bootz (bootz@cup.uni-muenchen.de)
- David Haring (dave@ibp.cz)
- Ulf Ryde (Ulf.Ryde@teokem.lu.se)
- Dermot Brougham (Dermot.Brougham@dcu.ie)
- Shin Ick-Dong (a9523303@chunma.yu.ac.kr)
- Drew Parsons (dparsons@emerall.com)
- Bernard Delley (bernard.delley@psi.ch)
- Nelson Henrique Morgon (morgon@canario.iqm.unicamp.br)
- Masao Kawamura (kawamura@mlb.co.jp)
- Rene Windiks (rene.windiks@psi.ch)
- Fulvio Ciriaco (ciriaco@chimica.uniba.it)
- Julien Bossert (bossert@quantix.u-strasbg.fr)
- Gábor Magyarfalvi (gmagyarf@para.chem.elte.hu)
- Gert von Helden (helden@fhi-berlin.mpg.de)
- Sebastian Canagaratna (s-canagaratna@onu.edu)
- Stephan Schenk (stephan.schenk@uni-jena.de)
- Noda Tomoyuki (noda.tomoyuki@canon.co.jp)
- Michael Patzschke (michaelp@chem.helsinki.fi)
- Ulrich Wedig (U.Wedig@fkf.mpg.de)
- Ivan Powis (Ivan.Powis@nottingham.ac.uk)
- Lars Hecking (lhecking@users.sourceforge.net)
- Yuusuke Sato (ysato@msl.rdc.toshiba.co.jp)

I also have to thank Mark J. Kilgard (mjk@nvidia.com) and Brian Paul (brian_paul@avid.com) who posted a number of the algorithms used in VIEWMOL on the Internet.

Translations of VIEWMOL's interface to other languages have been provided by:

- French: Ludovic Douillard (douillard@DRECAM.cea.fr)
- Hungarian: Gábor Magyarfalvi (gmagyarf@para.chem.elte.hu)
- Polish: Nikodem Kuznik (nikodem@zeus.polsl.gliwice.pl)
- Spanish: Jose R. Valverde (jrvalverde@cnb.uam.es)
- Turkish: Murat Guner (muratguner1982@hotmail.com)

References

[1] R. D. Amos, N. C. Handy, and P. Palmieri. Vibrational properties of (R)-methylthiirane from Møller-Plesset perturbation theory. *J. Chem. Phys.*, 93:5796, 1990.

20 Appendix: Thermodynamics

The thermodynamical calculations performed by VIEWMOL are using the following formulas:

- Enthalphy
  - Translation $H^T = \frac{3}{2}RT$
  - pV (molecule) $pV = RT$
  - pV (solid) $pV = pV$
  - Rotation (linear molecule) $H^R = RT$
  - Rotation (non-linear molecule) $H^R = \frac{3}{2}RT$
  - Vibration
$$H^V = N_A \left[\sum_i \frac{hc\bar{\nu}_i}{2} + \sum_i \frac{hc\bar{\nu}_i \exp(-hc\bar{\nu}_i/k_BT)}{1 - \exp(-hc\bar{\nu}_i/k_BT)} \right] \tag{1}$$
- Entropy
  - Translation
$$S^T = \frac{5}{2}R + R\ln\left(\frac{k_BT}{p}\left(\frac{2\pi M k_B T}{N_A h^2}\right)^{3/2}\right) \tag{2}$$
  - Rotation (linear molecule)
$$S^R = R\left[1 + \ln\left(\frac{k_BT}{\sigma h A}\right)\right] \tag{3}$$
  - Rotation (non-linear molecule)
$$S^R = \frac{R}{2}\left[3 + \ln\left(\frac{\pi}{\sigma}\left(\frac{k_BT}{hc}\right)^3 \frac{1}{ABC}\right)\right] \tag{4}$$

- Vibration

$$S^V = R\sum_i \left(\frac{hc\bar{\nu}_i}{k_BT}\frac{\exp\left(-hc\bar{\nu}_i/k_BT\right)}{1-\exp\left(-hc\bar{\nu}_i/k_BT\right)} - \ln\left(1-\exp\left(-\frac{hc\bar{\nu}_i}{k_BT}\right)\right)\right) \tag{5}$$

- Heat capacity
  - Translation $C_v^T = \frac{5}{2}R$
  - Rotation (linear molecule) $C_v^R = R$
  - Rotation (non-linear molecule) $C_v^R = \frac{3}{2}R$
  - Vibration

$$C_v^V = R\sum_i \left(\frac{hc\bar{\nu}_i}{k_BT}\right)^2 \frac{\exp\left(-hc\bar{\nu}_i/k_BT\right)}{\left(1-\exp\left(-hc\bar{\nu}_i/k_BT\right)\right)^2} \tag{6}$$

R gas constant, T temperature, p pressure, V volume, N_A Avogadro's number, h Planck's number, c speed of light, $\bar{\nu}_i$ wave number, k_B Boltzmann's number, M molecular mass, A, B, and C rotational constants, σ symmetry number. Only wave numbers which are larger than 10 cm^{-1} are included.