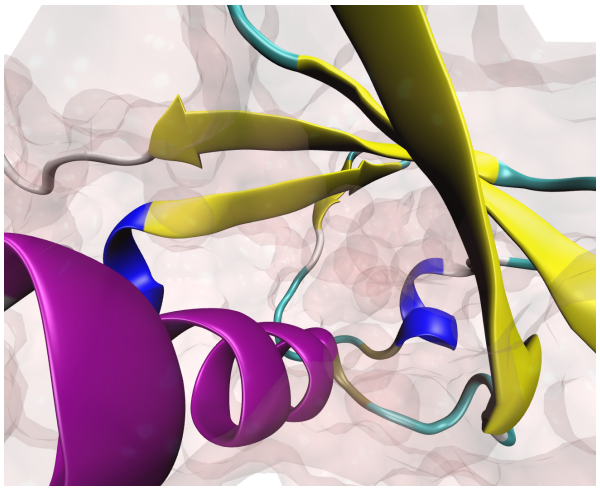
University of Illinois at Urbana-Champaign
Beckman Institute for Advanced Science and Technology
Theoretical and Computational Biophysics Group
Computational Biophysics Workshop

Using VMD



VMD Developer:
John E. Stone

Tutorial Contributors:
Alek Aksimentiev, Anton Arkhipov, Reuven Birnbaum, Robert Brunner, Jordi Cohen, Brijeet Dhaliwal, John Eargle, Jen Hsin, Fatemeh Khalili, Eric H. Lee, Zan Luthey-Schulten, Patrick O'Donoghue, Elijah Roberts, Anurag Sethi, Marcos Sotomayor, John E. Stone, Emad Tajkhorshid, Leonardo Trabuco, Elizabeth Villa, Yi Wang, David Wells, Dan Wright, Ying Yin

Aug. 2019

A current version of this tutorial is available at:
http://www.ks.uiuc.edu/Training/Tutorials/
Join the `tutorial-l@ks.uiuc.edu` mailing list for additional help.

Contents

Introduction

VMD (Visual Molecular Dynamics) is a molecular visualization and analysis program designed for biological systems such as proteins, nucleic acids, lipid bilayer assemblies, etc. The program is developed by the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign. Among molecular graphics programs, VMD is unique in its ability to efficiently operate on large biomolecular complexes and long-timescale molecular dynamics trajectories, its interoperability with a large number of molecular dynamics simulation tools, its integration of structure and sequence information, and its built-in support for advanced image rendering and movie making.

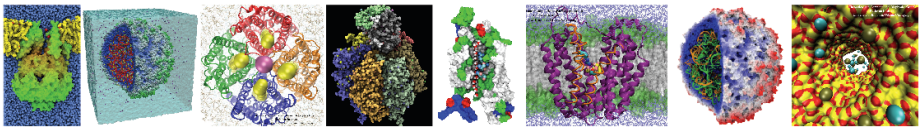


Figure 1: Example VMD renderings.

Key features of VMD include:

- General 3-D molecular visualization with extensive drawing and coloring methods and strong support for visualization of molecular dynamics
- Supports all major molecular data file formats, with no limits to structure or trajectory sizes except memory capacity and addressing
- Extensive atom selection syntax for choosing subsets of atoms for both analytical tasks and for graphical display
- Visualization of volumetric data
- Molecular analysis commands
- Rendering of high-resolution, publication-quality molecule images
- Built-in movie making tools
- Building and preparing systems for molecular dynamics simulations
- Interactive molecular dynamics simulations
- User-extensible through the built-in Tcl and Python scripting languages
- Extensible source code written in C and C++

This article will serve as an introductory VMD tutorial. It is impossible to cover all of VMD's capabilities, but here we will present several step-by-step examples of VMD's basic features. Topics covered in this tutorial include visualizing molecules in three dimensions with different drawing and coloring methods, rendering publication-quality figures, animating and analyzing molecular dynamics simulation trajectories, scripting in the text-based Tcl/Tk interface, and analyzing both sequence and structure data for proteins.

Downloading VMD

Before staring the tutorial you need to download the current version of VMD. This tutorial requires VMD version 1.9.2 or later. VMD supports all major computer platforms and can be obtained from the VMD homepage http://www.ks.uiuc.edu/Research/vmd. Follow the instruction online to install VMD in your computer. Once VMD is installed, to start VMD:

- **Mac OS X:** Double click on the VMD application icon in the Applications directory.
- **Linux and other Unix platforms:** Type `vmd` in a terminal window.
- **Windows:** Select Start → Programs → VMD.

When VMD starts, by default three windows will open (Fig. 2): the VMD Main window, the OpenGL Display window, and the VMD Console window (or a Terminal window on a Mac). To end a VMD session, go to the VMD Main window, and choose File → Quit. You can also quit VMD by closing the VMD Console window or the VMD Main window.

Tutorial Topics and Files

The tutorial contains six sections. Each section acts as an independent tutorial for a specific topic, with the section layout as shown in Contents. We suggest that readers with no prior VMD experience work through the sections in the order they are presented. Readers already familiar with the basics of VMD may selectively pursue sections of their interest. Several files have been prepared to accompany this tutorial. You need to download these files at http://www.ks.uiuc.edu/Training/Tutorials/vmd. The files needed for each chapter are illustrated in Fig. 3.

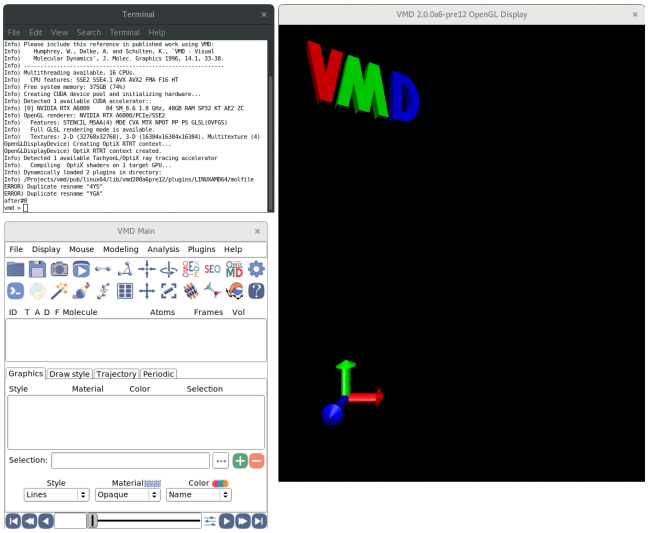


Figure 2: The VMD Main window, the OpenGL Display window, and the VMD Console window.

1 Working with a Single Molecule
`1ubq.pdb`

2 Trajectories and Movie Making
`ubiquitin.psf  pulling.dcd`

3 Scripting in VMD
`1ubq.pdb  beta.tcl`

4 Working with Multiple Molecules
`1fqy.pdb  1rc2.pdb`

5. Comparing Protein Structures and Sequences with the MultiSeq Plugin
`1fqy.pdb  1rc2.pdb  1lda.pdb  1j4n.pdb  spinach_aqp.fasta`

6 Data Analysis in VMD
`ubiquitin.psf  pulling.dcd  equilibration.dcd  distance.tcl`

Figure 3: The files needed for each section. All files are contained in the `vmd-tutorial-files` folder, which can be downloaded from http://www.ks.uiuc.edu/Training/Tutorials/vmd.

1 Working with a Single Molecule

In this section you will learn the basic functions of VMD. We will start with loading a molecule, displaying the molecule, and rendering publication-quality molecule images. This section uses the protein ubiquitin as an example molecule. Ubiquitin is a small protein responsible for labeling proteins for degradation, and is found in all eukaryotes with nearly identical sequences and structures.

1.1 Loading a Molecule

The first step is to load our molecule. A pdb file, `1ubq.pdb` (Vijay-Kumar et al., *JMB*, **194**:531, 1987), that contains the atom coordinates of ubiquitin is provided with the tutorial.

1 Start a VMD session. In the VMD Main window, choose File → New Molecule... (Fig. 4(a)). Another window, the Molecule File Browser window (Fig. 4(b)), will appear on your screen.

2 Use the Browse... (Fig. 4(c)) button to find the file `1ubq.pdb` in `vmd-tutorial-files` directory. Note that when you select the file, you will be back in the Molecule File Browser window. In order to actually load the file you have to press Load (Fig. 4(d)). Do not forget to do this!

Now, ubiquitin is shown in the OpenGL Display window. You may close the Molecule File Browser window at any time.

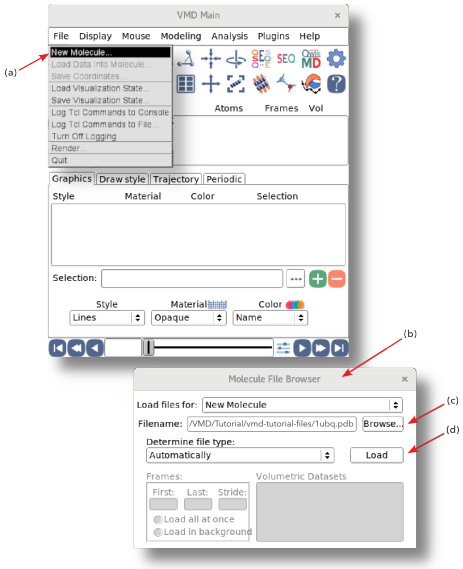


Figure 4: Loading a Molecule.

Webpdb. VMD can download a pdb file from the Protein Data Bank[1] if a network connection is available. Just type the four letter code of the protein in the File Name text entry of the Molecule File Browser window and press the Load button. VMD will download it automatically.

[1] Protein Data Bank website: http://www.pdb.org

1.2 Displaying the Molecule

In order to see the 3D structure of our protein, we will use the mouse in multiple modes to change the viewpoint. VMD allows users to rotate, scale and translate the viewpoint of your molecule.

1 In the OpenGL Display, press the left mouse button down and move the mouse. Explore what happens. This is the rotation mode of the mouse and allows you to rotate the molecule around an axis parallel to the screen (Fig. 5(a)).

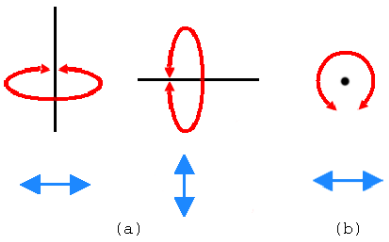


Figure 5: Rotation modes. (a) Rotation axes when holding down the left mouse key. (b) The rotation axis when holding down the right mouse key.

2 If you hold down the right mouse button and repeat the previous step, the rotation will be done around an axis perpendicular to your screen (Fig. 5(b)) (For Mac users, the right mouse button is equivalent to holding down the command key while pressing the mouse button).

3 In the VMD Main window, look at the Mouse menu (Fig. 6). Here, you will be able to switch the mouse mode from Rotation to Translation or Scale modes.

4 Choose the Translation mode and go back to the OpenGL Display. You can now move the molecule around when you hold the left mouse button down.

5 Go back to the the Mouse menu and choose the Scale mode this time. This will allow you to zoom in or out by moving the mouse horizontally while holding the left mouse button down.

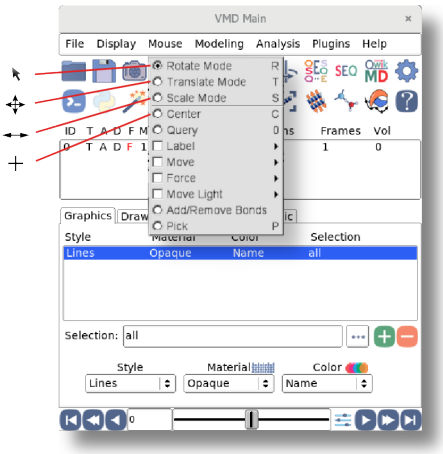


Figure 6: Mouse modes and their characteristic cursors.

It should be noted that these actions performed with the mouse only change your viewpoint and do not change any of the atomic coordinates associated with the molecule.

Mouse modes. Note that each mouse mode has its own characteristic cursor and its own shortcut key (`r`: Rotate, `t`: Translate, `s`: Scale). When you are in the OpenGL Display window, you can use these shortcut keys instead of the Mouse menu to change the mouse mode.

Another useful option is the Mouse → Center menu item. It allows you to specify the point around which rotations are done.

6 Select the Center menu item and pick one atom at one of the ends of the protein; The cursor should display a cross.

7 Now, press `r`, rotate the molecule with the mouse and see how your molecule moves around the point you have selected.

8 In the VMD Main window, select the Display → Reset View menu item to return to the default view. You can also reset the view by pressing the "=" key when you are in the OpenGL Display window.

1.3 Graphical Representations

VMD can display your molecule in various ways by the Graphical Representations shown in Fig. 7. Each representation is defined by four main parameters: the selection of atoms included in the representation, the drawing style, the coloring method, and the material. The selection determines which part of the molecule is drawn, the drawing method defines which graphical representation is used, the coloring method gives the the color of each part of the representation, and the material determines the effects of lighting, shading, and transparency on the representation. Let's first explore different drawing styles.

1.3.1 Exploring different drawing styles

1 In the VMD Main window is the Graphics tab, in which you can see the current default representation displaying your molecule (Fig. 7(a)).

2 In the Draw Style tab (Fig. 7(b)) we can change the style (Fig. 7(d)) and color (Fig. 7(c)) of the representation. In this section we will focus on the drawing style (the default is Lines).

3 Each Style has its own parameters. For instance, change the Thickness of the lines by using the controls on the left-hand side (Fig. 7(e)) of the Draw Style tab.

4 Click on the Style menu (Fig. 7(d)), and you will see a list of options. Choose VDW (van der Waals). Each atom is now represented by a sphere, allowing you to see more easily the volumetric distribution of the protein.

5 When you choose VDW for drawing method, two new controls would show up in the lower right-hand-side corner (Fig. 7(e)). Use these controls to change the Sphere Scale to 0.5 and the Sphere Resolution to 13. Be aware that the higher the resolution, the slower the display of your molecule will be.

6 Press the Default button. This allows you to return to the default properties of the chosen drawing method.

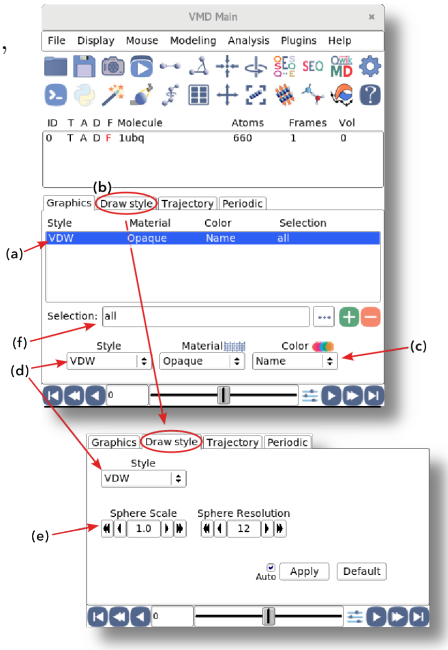


Figure 7: The Graphical Representations window.

The previous representations allow you to see the micromolecular details of your protein by displaying every single atom. More general structural properties can be demonstrated better by using more abstract drawing methods.

7 Choose the Tube style under Drawing Method and observe the backbone of your protein. Set the Radius at 0.8. You should get something similar to Fig. 8.

8 By looking at your protein in the tube drawing method, see if you can distinguish the helices, β-sheets and coils present in the protein.

More representations. Other popular representations are CPK and Licorice. In CPK, like in old chemistry ball & stick kits, each atom is represented by a sphere and each bond is represented by a thin cylinder (radius and resolution of both the sphere and the cylinder can be modified independently). The Licorice drawing method also represents each atom as a sphere and each bond as a cylinder, but the sphere radius cannot be modified independently.

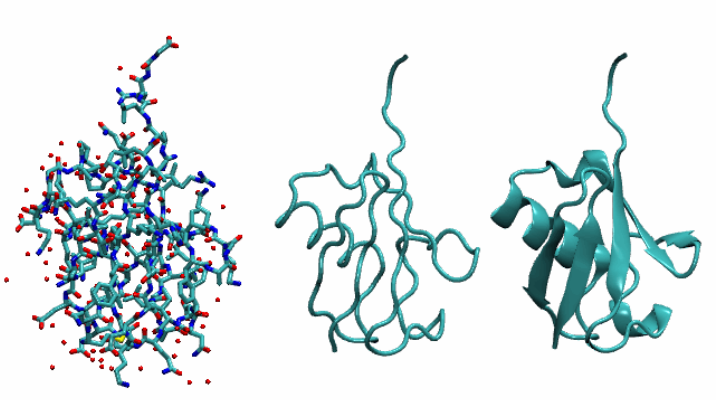


Figure 8: Licorice (left), Tube (center) and NewCartoon (right) representations of Ubiquitin

The last drawing method we will explore is NewCartoon. It gives a simplified representation of a protein based in its secondary structure. Helices are drawn as coiled ribbons, β-sheets as solid arrows and all other structures as a tube. This is probably the most popular drawing method to view the overall architecture of a protein.

9 In the Graphical Representations window, choose Drawing Method → NewCartoon. You can now easily identify how many helices, β-sheets and coils are present in the protein.

Structure of ubiquitin. Ubiquitin has three and one half turns of α-helix (residues 23 to 34, three of them hydrophobic), one short piece of 3_{10}-helix (residues 56 to 59) and a mixed β-sheet with five strands (residues 1 to 7, 10 to 17, 40 to 45, 48 to 50, and 64 to 72) and seven reverse turns. VMD uses the program STRIDE (Frishman et al., *Proteins*, **23**:566, 1995) to compute the secondary structure according to an heuristic algorithm.

1.3.2 Exploring different coloring methods

Now, let's explore different coloring methods for our representations.

10 In the Graphical Representations window, you can see that the default coloring method is Color → Name. In this coloring method, if you choose a drawing method that shows individual atoms, you can see that they have different colors, i.e: O is red, N is blue, C is cyan and S is yellow.

11 Choose Color → ResType (Fig. 7(c)). This allows you to distinguish non-polar residues (white), basic residues (blue), acidic residues (red) and polar residues (green).

12 Select Color → Secondary Structure (Fig. 7(c)) and confirm that the New-Cartoon representation displays colors consistent with secondary structure.

1.3.3 Displaying different selections

You can also only display parts of the molecule that you are interested in by specifying you selection in the Graphical Representations window (Fig. 7(f)).

13 In the Graphical Representations window, there is a Selected Atoms text entry (Fig. 7(f)). Delete the word `all`, type `helix` and press the Apply button or hit the Enter/Return key on your keyboard (remember to do this whenever you change a selection). VMD will show just the helices present in our molecule.

14 In the Graphs tab open the Selections window (Fig. 9(a)) by clicking the button with three dots (Fig. 9 red circle). In section Singlewords (Fig. 9(b)), you will find a list of possible selections you can type. For instance, try to display β-sheets instead of helices by typing the appropriate word in the Selected Atoms text entry.

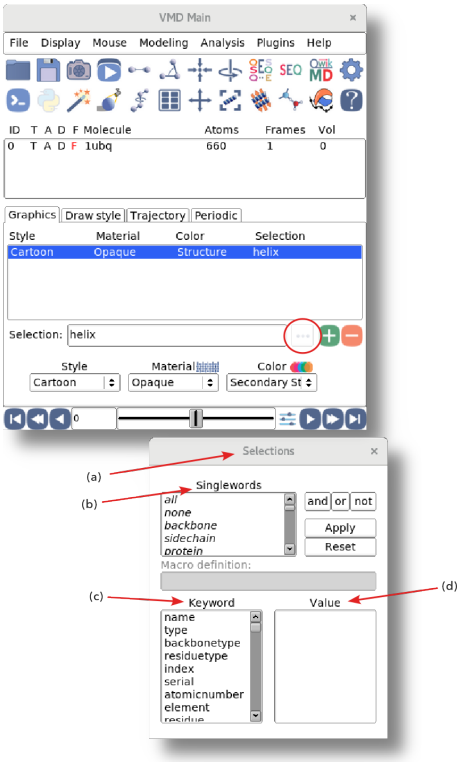


Figure 9: Graphical Representations window and the Selections window.

Combinations of boolean operators can also be used when writing a selection.

15 In order to see the molecule without helices and β-sheets, type the following in Selected Atoms: `(not helix) and (not betasheet)`. Remember to press the Apply button or hit the Enter/Return key on your keyboard.

16 In the section Keyword (Fig. 9(c)) of the Selections tab, you can see properties that can be used to select parts of a protein with their possible values. Look at possible values of the keyword resname (Fig. 9(d)). Display all the lysines and glycines present in the protein by typing `(resname LYS) or (resname GLY)` in the Selected Atoms. Lysines play a fundamental role in the configuration of polyubiquitin chains.

17 Now, change the current representation's Drawing Method to CPK and the Coloring Method to ResName in the Draw Style tab. In the screen you will be able to see the different Lysines and Glycines.

18 In the Selected Atoms text entry type `water`. Choose Color → Name. You should see the 58 water molecules (in fact only the oxygens) present in our system.

19 In order to see which water molecules are closer to the protein you can use the command `within`. Type `water and within 3 of protein` for Selected Atoms. This selects all the water molecules that are within a distance of 3 angstroms of the protein.

20 Finally, try typing the following selections in Selected Atoms:

Selection	Action
`protein`	Shows the Protein
`resid 1`	The first residue
`(resid 1 76) and (not water)`	The first and last residues
`(resid 23 to 34) and (protein)`	The α-helix

Table 1: Example atom selections.

1.3.4 Creating multiple representations

The button Create Rep (Fig. 10(a)) in the Graphical Representations window allows you to create multiple representations. Therefore, you can have a mixture of different selections with different styles and colors, all displayed at the same time.

21 For the current representation, in Selected Atoms type `protein`, set the Drawing Method to NewCartoon and the Coloring Method to Secondary Structure.

22 Press the Create Rep button (Fig. 10(a)). You should see that a new representation is created. Modify the new representation to get VDW as

the Drawing Method, ResType as the Coloring Method, and `resname LYS` as the current selection.

23 Repeating the previous procedure, create the following two new representations:

Selection	Coloring Method	Drawing Method
`water`	Name	CPK
`resid 1 76 and name CA`	ColorID → 1	VDW

Table 2: Example representations.

24 Create the last representation by pressing again the Create Rep button. Select Drawing Method → QuickSurf for drawing method, Color → Molecule for coloring method, and type `protein` in the Selected Atoms entry. Set the QuickSurf radius scale parameter to 0.7. For this last representation choose Transparent in the Material pull-down menu (Fig. 10(c)). This representation shows protein's volumetric surface in transparent.

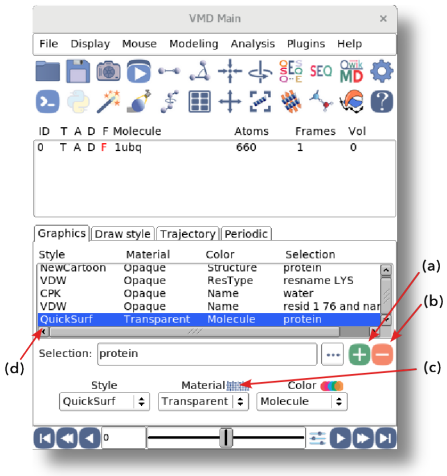


Figure 10: Multiple Representations of Ubiquitin.

25 Note that you can select and modify different representations you have created by clicking on a representation to highlight it in yellow. Also, you can switch each representation on/off by double-clicking on it. You can also delete a representation by highlighting it and clicking on the Delete Rep button (Fig. 10(b)). At the end of this section, your Graphical Representations window should look similar to Fig. 10.

1.4 Sequence Viewer Extension

When dealing with a protein for the first time, it is very useful to find and display different amino acids quickly. The sequence viewer extension allows you to view the protein sequence, as well as picking and displaying one or more residues of your choice easily.

1 In the VMD Main window, choose the Analysis → Sequence Viewer menu item. A window (Fig. 11(a)) with a list of the amino acids (Fig. 11(e)) and their properties (Fig. 11(b)&(c)) will appear in your screen.

2 With the mouse, try clicking on different residues in the list (Fig. 11(e)) and see how they are highlighted. In addition, the highlighted residue will appear in your OpenGL Display window in yellow and bond drawing method, so you can visualize its location within the protein easily.

3 Using the Zoom controls (Fig. 11(f)) you can display the entire list of residues in the window. This is especially useful for larger proteins

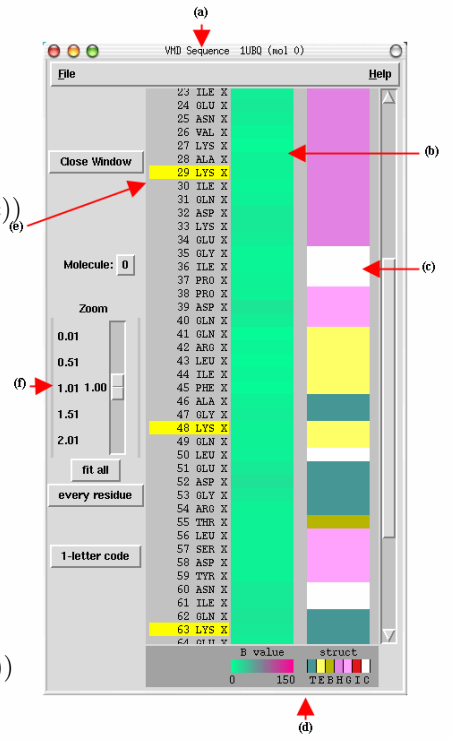


Figure 11: VMD Sequence window.

4 To pick multiple residues, hold the shift key and click on the mouse button. Try highlighting residues 11, 48, 63 and 29 (Fig. 11(e)).

5 Look at the Graphical Representations window, you should find a new representation with the residues you have selected using Sequence Viewer Extension. You can modify, hide or delete this representation similar to what you have done before.

Information about residues is color-coded (Fig. 11(d)) in columns and obtained from STRIDE. The B-value column (Fig. 11(b)) shows the B-value field (temperature factor). The struct column shows secondary structure (Fig. 11(d)), where each letter means:

1.5 Saving Your Work

The viewpoints and representations that you have created using VMD can be saved as a VMD state. This VMD state contains all the information needed to

T	Turn
E	Extended conformation (β-sheets)
B	Isolated bridge
H	Alpha helix
G	3-10 helix
I	Pi helix
C	Coil

Table 3: Secondary Structure codes used by STRIDE.

reproduce the same VMD session without losing what you have done.

1 Go to the OpenGL Display window, use your mouse to find a nice view of the protein. We will save this viewpoint using VMD ViewMaster.

2 In the VMD Main window, select Plugins → Visualization → ViewMaster. This will open the VMD ViewMaster window.

3 In the VMD ViewMaster window, click on the Create New button. Now you have saved your OpenGL Display view point.

4 Go back to your OpenGL Display window, use your mouse to find another nice view. If you want you can also add/delete/modify a representation in the Graphical Representations window. When you have found a good view, you can again save it by returning to the VMD ViewMaster window and clicking on the Creat New button.

5 Create as many views as you like by repeating the previous step. You can see that in the VMD ViewMaster window, all of your viewpoints are displayed as thumbnails. You can go to a previously-saved viewpoint by clicking on its thumbnail.

6 Let's now save the entire VMD session. In the VMD Main window, choose the File → Save Visualization State menu item. Write an appropriate name (e.g., `myfirststate.vmd`) and save it. The VMD state file `myfirststate.vmd` contains all the information you need to restore your VMD session, including the viewpoints and the representations.

To load a saved VMD state, start a new VMD session and in the VMD Main window choose File → Load State.

7 Quit VMD.

1.6 The Basics of VMD Figure Rendering

One of VMD's many strengths is its ability to render high-resolution, publication-quality molecule images. In this section we will introduce some basic concepts of figure rendering in VMD.

1.6.1 Setting the display background

Before you render a figure, you want to make sure you set up the OpenGL Display background the way you want. Nearly all aspects of the OpenGL Display are user-adjustable, including background color.

1 Start a new VMD session. Load the `1ubq.pdb` file in the `vmd-tutorial-files` directory by following the steps in Section 1.1.

2 In the VMD Main window, choose the Graphics tab. Clicking the icon (overlapping shaded circles) above the Colors menu (Fig. 7(c)) should open the Color Controls window. Look through the Categories list. All display colors, for example, the colors of different atoms when colored by name, are set here.

3 Now we will change the background color. In Categories, select Display. In Names, select Background. Finally, choose 8 white in Colors. Your OpenGL Display now should have a white background.

4 When making a figure, we often don't want to include the axes. To turn off the axes, select Display → Axes → Off in the VMD Main window.

1.6.2 Increasing resolution

Most VMD graphical representations are drawn with an adjustable resolution, allowing users to balance geometric detail with interactive drawing speed.

5 Open the Graphics tab in the VMD Main menu. Modify the default representation to show just the protein, and display it using the VDW drawing method.

6 Zoom in on one or two of the atoms, either by using the scroll wheel on your mouse, or by using Mouse → Scale Mode (shortcut `s`).

OpenGL clipping. You might notice that as you zoom or move the view close up into an atom, it might be cut open or chopped away by an invisible clipping plane that lies just ahead or coincident with the position of the camera. The so-called "near" clipping plane in OpenGL is a detail of its implementation, and it can be made nearly (but not quite) zero to minimize its impact. You can move the molecule toward or away from you (the camera position) so that the clipping doesn't impact the molecular scene in a detrimental way by doing the following: switch your mouse mode to the Translate mode, either by pressing the shortcut key "t" in the OpenGL window or by selecting Mouse → Translate Mode, and drag the mouse in the OpenGL window while holding down the right mouse key. You can now move the molecule toward or away from you (the camera). If moving the molecular scene toward or away from the camera is ineffective, the default OpenGL "near" clipping plane can be adjusted as follows: in the VMD Main window, choose Display → Display Settings...; in the Display Settings window that shows up, you can see many OpenGL display options you can adjust; decrease the value for Near Clip, this will move the OpenGL clipping plane closer, allowing you to zoom or translate the camera into individual atoms without clipping them off.

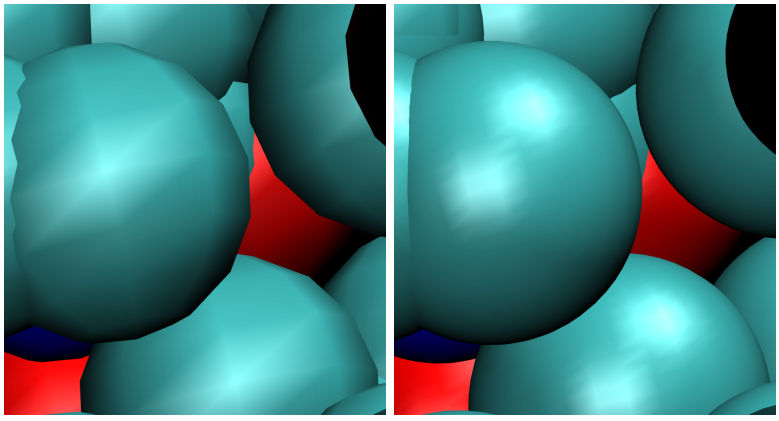
7 Notice that with the default resolution setting, the "spherical" atoms aren't looking very spherical. In the Graphics tab, click on the representation you set up before for the protein to highlight it. Adjust the Sphere Resolution, setting it a higher value, and see what a difference it can make. (See Fig. 12.)

Many of the drawing methods have a resolution setting. Try a few different drawing methods and see how you can easily increase their resolutions. When producing images, you can raise the resolution until it stops making a visible difference.

1.6.3 Colors and materials

8 You may have noticed the Material menu in the Graphics tab (which by default is drawn with Opaque material). Choose the protein representation you made before, and experiment with the different materials in the Material menu.

9 Besides the pre-defined materials in the Material menu, VMD also allows users to create their own materials. To begin making a new material, click the icon next to the Materials menu (blue weave pattern). In the Materials window that appears, you'll see a list of the materials you just tried out, and their adjustable settings. Click the Create New button.



(a) Low resolution: Sphere Resolution set to 8

(b) High resolution: Sphere Resolution set to 28

Figure 12: The effect of the resolution setting.

A new material, e.g. Material23, will be created. Give it the following settings:

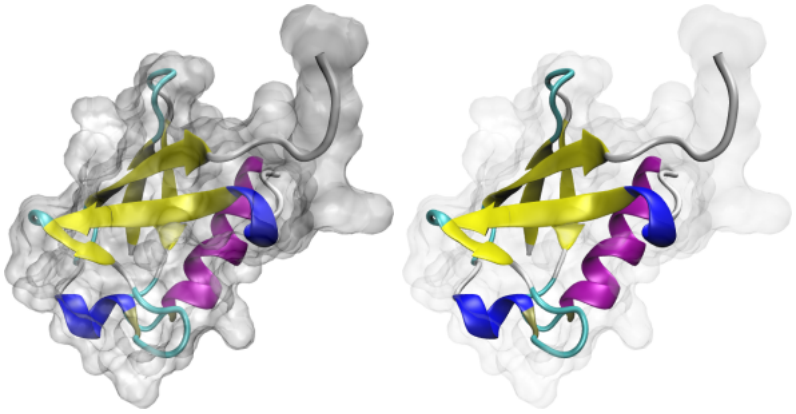
Setting	Value
Ambient	0.30
Diffuse	0.30
Specular	0.90
Shininess	0.50
Opacity	0.95

Table 4: Example user-defined material.

10 In the Material menu of the Graphics tab, you can see that Material23 is now on the list. Try using Material23 for a representation and see what it looks like.

GLSL. Now is a good time to try out the GLSL Render Mode, if your computer supports it. In the VMD Main window, choose Display → Rendermode → GLSL. This mode uses your 3D graphics card to render the scene with real-time raytracing of spheres and alpha-blended transparency; see Fig. 13 for an example.

11 If your computer supports GLSL Render Mode, you can try to reproduce Fig. 13(b). First turn on the GLSL rendering mode by selecting Display → Rendermode → GLSL in the VMD Main window.



(a) The default transparent material. (b) A user-defined material.

Figure 13: Examples of different material settings.

12 Modify Material23 to be more transparent by entering the following values in the Materials window:

Setting	Value
Ambient	0.30
Diffuse	0.50
Specular	0.87
Shininess	0.85
Opacity	0.11

Table 5: Example of a more transparent material.

13 Hide all of your current representations and create the following two representations:

Selection	Coloring Method	Drawing Method	Material
`protein`	Structure	NewCartoon	Opaque
`protein`	ColorID → 8 white	QuickSurf	Material23

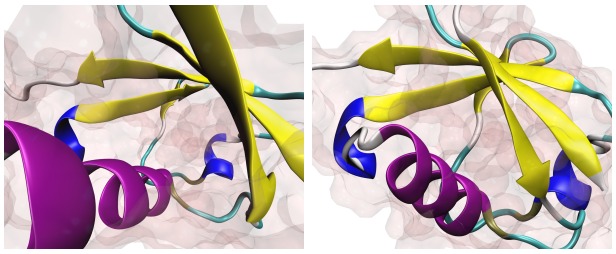
Table 6: Example of representations drawn with different materials.

1.6.4 Depth perception

Since the systems we are dealing with are three-dimensional, VMD has multiple ways of representing the third dimension. In this section, we will explore how to use VMD to enhance or hide depth perception.

14 The first thing to consider is the projection mode. In the VMD Main window, click the Display menu. Here we can choose either Perspective or Orthographic in the drop-down menu. Try switching between Perspective and Orthographic projection modes and see the difference (Fig. 14).

In perspective mode, things nearer the camera appear larger. Although perspective projection provides strong size-based visual depth cues, the displayed image will not preserve scale relationships or parallelism of lines, and objects very close to the camera may appear distorted. Orthographic projection preserves scale and parallelism relationships between objects in the displayed image, but greatly reduces depth perception.



(a) Perspective (b) Orthographic

Figure 14: Comparison of perspective and orthographic projection modes.

Orthographic versus Perspective Mode. Orthographic mode tends to be more useful for analysis, because alignment is easy to see, while perspective mode is often used for producing figures and stereo images.

Another way VMD can represent depth is through the so-called "depth cueing". Depth cueing is used to enhance three-dimensional perception of molecular structures, particularly with orthographic projections.

15 Choose Display → Depth Cueing in the VMD Main window. When depth cueing is enabled, objects further from the camera are blended into the background. Depth cueing settings are found in Display → Display Settings.... Here you can choose the functional dependence of the shading

on distance, as well as some parameters for this function. To see the effect better, you might want to hide the representation with the QuickSurf drawing method.

16 Finally, VMD can also produce stereo images. In the VMD Main window, look at the Display → Stereo menu, showing many different choices. Choose SideBySide (remember to return to Perspective mode for a better result). You should get something like Fig. 15

17 Turn off stereo image by selecting Display → Stereo → Off in the VMD Main window. Also turn off depth cueing by unselecting the Display → Depth Cueing checkbox in the VMD Main window.

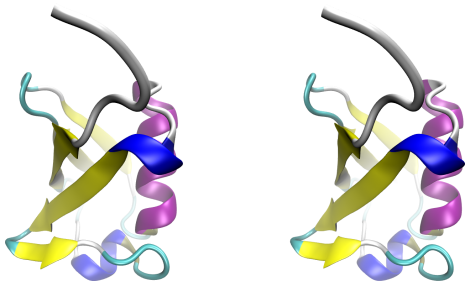


Figure 15: Stereo image of the ubiquitin protein. Shown here with Cue Mode = Linear, Cue Start = 1.5, and Cue End = 2.75.

1.6.5 Rendering

By now we've seen some techniques for producing nice views and representations of the molecule loaded in VMD. Now, we'll explore the use of the VMD built-in snapshot feature and external rendering programs to produce high quality images of your molecule. The "snapshot" renderer saves the on-screen image in your OpenGL window and is often adequate for use in drafts of presentations, movies, and small figures. When one desires higher quality images for publication, renderers such as Tachyon (and its hardware-accelerated variants) and POV-Ray are much better choices due to their improved rendering quality and support for advanced lighting and shading. One of the key benefits of Tachyon and POV-Ray vs. the "snapshot" renderer is that they can directly render curved geometric primitives such as spheres and cylinders, eliminating the need to be concerned with the "resolution" parameters for most graphical representations as described in Sec. 1.6.2.

Sophisticated renderers such as Tachyon and POV-Ray perform challenging rendering calculations in exchange for the high quality images they produce. When rendering very high resolution images in combination with large numbers of transparent surfaces, shadows, ambient occlusion lighting, and other advanced features, the rendering time required by Tachyon or POV-Ray can start to become a noteworthy consideration, particularly in the context of movie rendering. VMD includes an I/O optimized Tachyon rendering path called TachyonInternal that completely avoids writing VMD scene files and constituent geometry to disk, instead rendering directly from the in-memory VMD scene and writing out only the final rendered image. The use of TachyonInternal provides a large performance benefit when rendering large geometrically complex VMD scenes and is the basis for even higher performance hardware-optimized Tachyon variants. VMD optionally incorporates two hardware-optimized "light weight" versions of the Tachyon rendering engine that use GPU-acceleration and/or CPU vectorization to achieve higher performance than the fully-general cross-platform version of Tachyon. These hardware-optimized variants of Tachyon are referred to as TachyonLOptiX and TachyonLOptiXInternal (NVIDIA GPU-acceleration), and TachyonLOSPRay and TachyonLOSPRayInternal (Intel CPU vectorization), each of which offer substantial performance gains (frequently 2× up to as much as 10× faster) over the full-featured cross-platform Tachyon renderer.

18 Hide or delete all your previous representations, and create the new representations shown in Table 7.

Selection	Coloring Method	Drawing Style	Material
`protein and not resid 72 to 76`	Structure	NewCartoon	Opaque
`protein and helix and name CA`	ColorID → 8	QuickSurf	Material 23 (or Material with another number)
`resname GLY and not resid 72 to 76`	ColorID → 7	VDW	Opaque
`resname LYS`	ColorID → 18	Licorice	Opaque

Table 7: Example representations.

19 Rendering is very simple in VMD. Once you have the scene set the way you like it in the OpenGL window, simply choose File → Render... in the VMD Main window. The File Render Controls window will appear on your screen.

20 The File Render Controls allows you to choose which renderer you want to use and the file name for your image. For our first try, let's select snapshot

for the rendering method, type in a filename of your choice, and click Start Rendering.

21 If you are using a Mac or a Linux machine, an image-processing application might open automatically that shows you the molecule you have just rendered using snapshot. If this is not the case, use any image-processing application to take a look at the image file. Close the application when you are done to continue using VMD.

22 Try to render again using different rendering method, particularly TachyonInternal and POV3. If you're using a GPU-accelerated hardware platform, try using one of the TachyonL-OptiX renderers, and if running on appropriate Intel CPU hardware, try TachyonLOSPRay. Compare the quality of the images created by different renderers.

Renderers. The snapshot renderer saves exactly what is already showing in your display window — in fact, if another window overlaps the display window, it may distort the overlapped region of the image. The other renderers (e.g. POV3 and Tachyon) reprocess everything, so it may not look exactly as it does in the OpenGL window. In particular, they don't "clip", or hide, objects very near the camera. If you select Display → Display Settings... in the VMD Main window, you can set Near Clip to 0.01 to get a better idea of what will appear in your rendering.

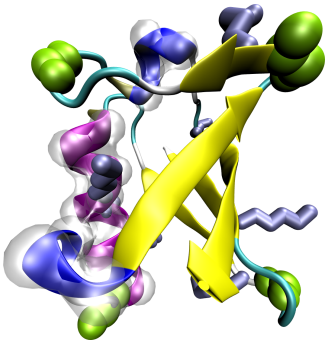


Figure 16: Example of a POV-Ray 3.6 (POV3) rendering.

23 You have learned the basics of VMD. Quit VMD.

2 Trajectories and Movie Making

The time-evolving coordinates of a system are called a trajectory. They are most commonly obtained in simulations of molecular systems, but can also be generated by other means and for different purposes. Upon loading a trajectory, VMD can animate movies of system evolving with time and analyze various structural features throughout the trajectory. This section will introduce the basics of working with trajectory data in VMD. You will also learn how to analyze trajectory data in Section 4.

2.1 Loading Trajectories

Trajectory files are typically large binary files that contain the time varying atomic coordinates for the system. Each set of coordinates corresponds to one frame in time. An example of a trajectory file is a DCD file. Trajectory files usually do not contain information structural information as found in protein structure files (PSF). Therefore, we must first load the structure file, and then add the trajectory data to the same molecule, so that VMD has access to both the structure and trajectory information.

1 Start a new VMD session. In the VMD Main window, select File → New Molecule.... The Molecule File Browser window should appear on your screen.

2 Use the Browse... button to find the file `ubiquitin.psf` in `vmd-tutorial-files` in the tutorial directory. When you select the file, you will be back in the Molecule File Browser window. Press the Load button to load the molecule.

3 In the Molecule File Browser window, make sure that `ubiquitin.psf` is selected in the `Load files for:` pull-down menu on the top, and click on the Browse button. Browse for `pulling.dcd`. Note the options available in the Molecule File Browser window: one can load trajectories starting and finishing at chosen frames, and adjust the stride between the loaded frames. Leave the default settings so that the whole trajectory is loaded.

4 Click on the Load button in the Molecule File Browser window. You will be able to see the frames as they are loaded into the molecule. After the trajectory finishes loading, you will be looking at the last frame of your trajectory. To go to the beginning, use the animation tools at the lower part of the VMD Main menu (see Fig. 17). You can close the Molecule File Browser window.

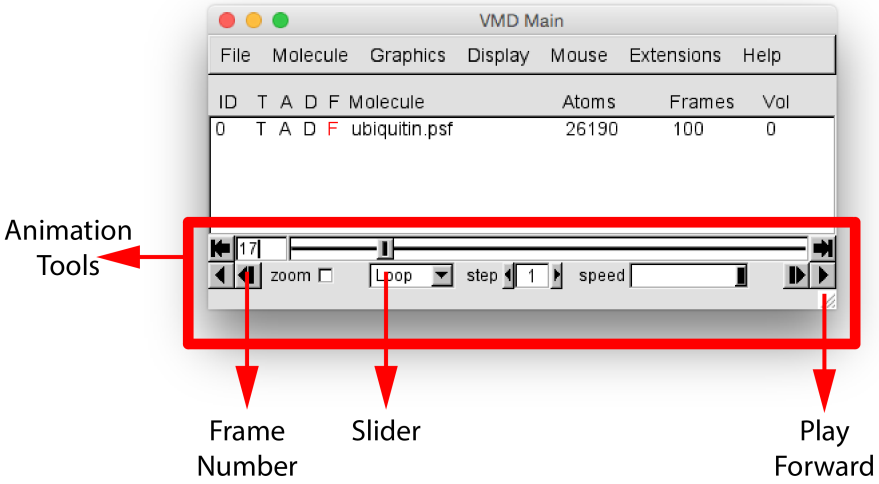


Figure 17: Animation tools in the main menu of VMD. The tools allow one to go over frames of the trajectory (e.g., using the slider) and to play a movie of the trajectory in various modes (Once, Loop, or Rock) and at an adjustable speed.

5 For a convenient visualization of the protein, choose Graphics → Representations in the VMD Main menu. In the Selected Atoms field, type `protein` and hit Enter on your keyboard; in the Drawing Method, select NewCartoon; in the Coloring Method, select Structure.

The trajectory you just loaded is a simulation of an AFM (Atomic Force Microscopy) experiment pulling on a single ubiquitin molecule, performed using the Steered Molecular Dynamics (SMD) method (Israelewitz et al., *Curr. Opin. Struct. Biol.*, **11**:224, 2001). We are looking at the behavior of the protein as it unfolds while being pulled from one end, with the other end being constrained to its original position. Each frame step corresponds to 10 ps. Ubiquitin has many functions in the cell, and it is currently believed that some of these functions depend on the protein's elastic properties. Such elastic properties are usually due to hydrogen bonding between residues in β strands of the protein molecules.

2.2 Main Menu Animation Tools

You can now play the movie of the loaded trajectory back and forth, using the animation tools in Fig. 17.

1 By dragging the slider one navigates through the trajectory. The buttons

to the left and to the right from the slider panel allow one to jump to the end of the trajectory or go back to the beginning.

2 For example, create another representation for water in the Graphical Representations window: click on the Create Rep button; in the Selected Atoms field, type `water` and hit enter; in Drawing Method, choose Lines; in the Coloring Method, select Name. This shows the water droplet present in the simulation to mimic the natural environment for the protein. Using the slider, observe the behavior of the water around the protein. The shape of the water droplet changes throughout the simulation, because water molecules follow the protein as it unfolds, due to interactions with the protein surface.

3 When playing animations, you can choose between three looping styles: Once, Loop and Rock. You can also jump to a frame in the trajectory by entering the frame number in the window on the left of the slider panel.

2.3 Trajectory Visualization

We will now learn some basic visualization tricks that are useful for working with trajectories.

2.3.1 Smoothing trajectories

1 For clarity, turn off the water representation by double-clicking on it in the Graphics tab. As you might have noticed, when we play the animation, the protein movements are not very smooth due to thermal fluctuations (as the simulation is performed under the conditions that mimic a thermal bath).

2 VMD can smooth the animation by averaging some number of frames. In the Graphics tab, select your protein representation and click the Trajectory tab. At the bottom, you see Trajectory Smoothing Window Size set to zero. As your animation is playing, increase this setting. Notice that the motion gets smoother and smoother as the setting goes up.

2.3.2 Displaying multiple frames

We will learn now how to display many frames of the same trajectory at once.

3 In the Graphics tab, highlight your protein representation by clicking on it and press the Create Rep button (green plus). This creates an identical representation, but note that smoothing is set to zero. Hide the old protein representation.

4 Highlight the new protein representation and click the Trajectory tab. Above the smoothing control, notice the Draw Multiple Frames control. It is set to `now` by default, which is simply the current frame. Enter `0:10:99`, which selects every tenth frame from the range 0 to 99.

5 Go back to the Graphics tab, and select Color → Trajectory → Timestep. This will draw the beginning of the trajectory in red, the middle in white, and the end in blue.

6 We can also use smoothing to make the large-scale motion of the protein more apparent. Go back to the Trajectory tab, set the smoothing window to 20. The result should be similar to Fig. 18.

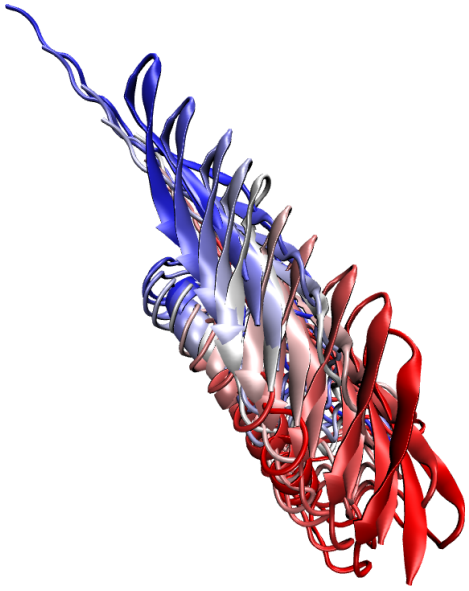


Figure 18: Image of every tenth frame showed at once, smoothed with a 20-frame window.

2.3.3 Updating selections

Now we will see how to make VMD update the selection each frame.

7 Hide the current representation showing all frames, and display only the water representation by double-clicking on it. Change the text in the Selected Atoms from `water` to `water and within 3 of protein` and hit enter. This will show all water atoms within 3 Å of the protein.

8 Play the trajectory. Although the displayed water atoms may be near the protein for a little while, they soon wander off, and are still shown despite no longer meeting the selection criteria. The Update Selection Every Frame option in the Trajectory tab of the Graphical Representations window remedies this. If the option box is checked, the selection is updated every frame. See Fig. 19.

9 Quit VMD.

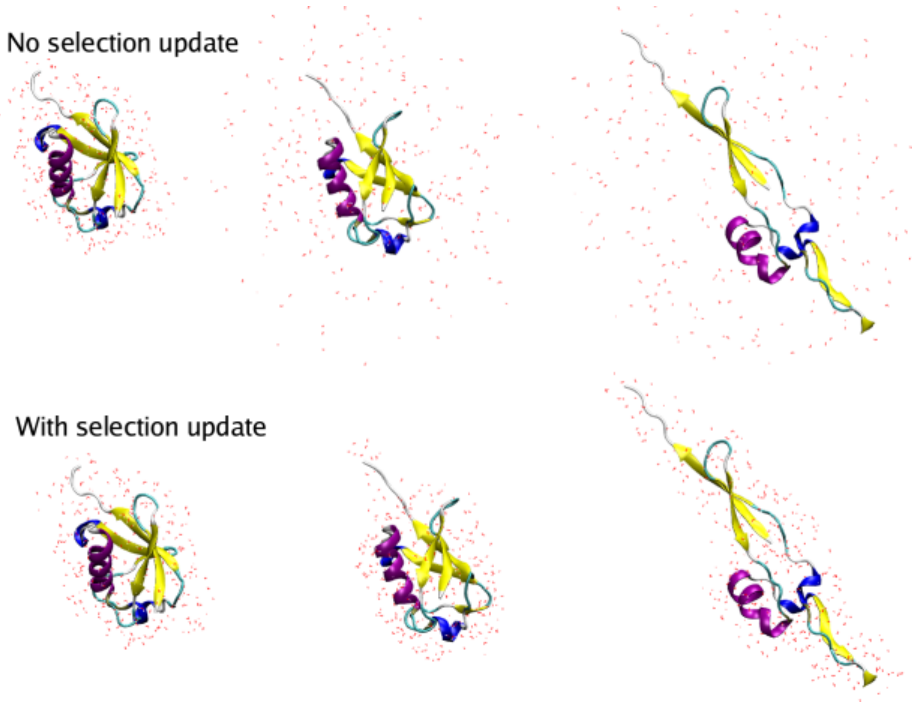


Figure 19: Water within 3 Å of the protein, shown for a selection that is not updated and for the one that is updated each frame. The snapshots shown are (from left to right) for frames 0, 17, and 99.

2.4 The Basics of Move Making in VMD

We will now learn how to make a basic movie.

1 Start a new VMD session. Repeat steps 1-5 in Section 2.1 to load the ubiquitin trajectory into VMD and display the protein in a secondary structure representation.

2 To make movies, we will use the VMD Movie Maker plugin. In the VMD Main window, go to menu item Plugins → Visualization → Movie Maker. The VMD Movie Generator window will appear (Fig. 20).

2.4.1 Making single-frame movies

3 First, let us look at some of the options for making a movie. Click on the Movie Settings menu in the VMD Movie Generator window. You can see that in addition to a trajectory movie, Movie Maker can also make a movie by rotating the view point of a single frame. In the Renderer menu, one can choose the type of renderer for making the movie. We will use the default option, Snapshot. One can also choose the output file format for the movie in menu item Format.

Movie rendering. While renderers other than Snapshot, such as Tachyon, generally provide more visually appealing images, they also require longer time for rendering. The rendering time is also affected by the size of the OpenGL window, since it takes more computing time to render a larger image. Likewise, the size of the movie file is determined by the size (resolution) of that window.

4 We will first make a movie of just one frame of the trajectory. For that purpose, select Rock and Roll option in the Movie Settings menu in the VMD Movie Generator window. Set the working directory to any convenient directory of your choice, give your movie a name, and click Make Movie.

5 Once rendering is finished, open and view the movie with your favorite application. This movie setting is good for showing one side of your system primarily.

Software requirements. If you cannot successfully make movies with VMD, it's possible that you're missing some softwares required for generating movies. All the required softwares are freely available, and to find what software you need, please see the VMD Movie Plugin page at http://www.ks.uiuc.edu/Research/vmd/plugins/vmdmovie/.

2.4.2 Making trajectory movies

6 Now we will make a movie of the trajectory. In the VMD Movie Generator window, select Movie Settings → Trajectory, give this one a different name, and click Make Movie. Note that the length of the movie is automatically set 24 frames per second. For a trajectory, duration of the movie can be decreased, but cannot be increased.

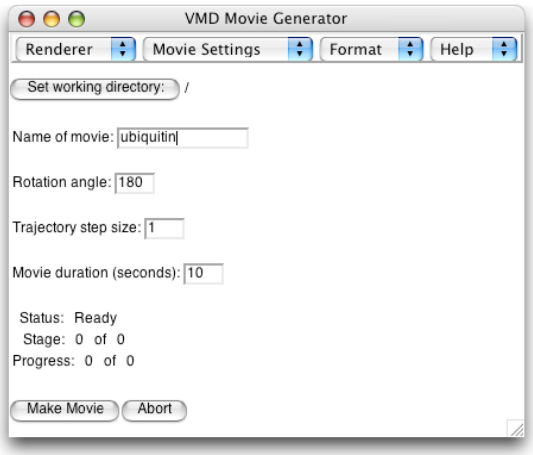


Figure 20: The VMD Movie Generator window.

7 Try out different options in the VMD Movie Generator window. Once you are done, quit VMD.

3 Scripting in VMD

VMD provides embedded scripting languages (Python and Tcl) for the purpose of user extensibility. In this section we will discuss the basic features of the Tcl scripting interface in VMD. You will see that everything you can do in VMD interactively can also be done with Tcl commands and scripts, and how the extensive list of Tcl text commands can help you investigate molecule properties and perform analysis.

The Tcl/Tk scripting language. Tcl is a rich language that contains many features and commands, in addition to the typical conditional and looping expressions. Tk is an extension to Tcl that permits the writing of graphical user interfaces with windows and buttons, etc. More information and documentations about the Tcl/Tk language can be found at http://www.tcl.tk/doc.

3.1 The Basics of Tcl Scripting

To execute Tcl commands, you will be using a convenient text console called Tk Console.

1 Start a new VMD session. In the VMD Main menu select Plugins → Tk Console to open the VMD TkConsole window (Fig. 21). You can now start entering Tcl/Tk commands here.

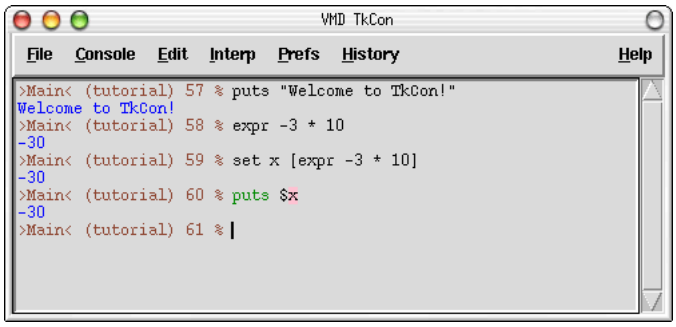


Figure 21: The VMD Tk Console window.

Let's start with the very basics of Tcl/Tk. Here are Tcl's `set` and `puts` commands:

`set` *variable value*	– sets the value of *variable*
`puts` $*variable*	– prints out the value of *variable*

2 Try entering the following commands in the VMD TkConsole window. Remember to hit enter after each line and take a look at what you get after each input.

```
set x 10
puts "the value of x is:$x"
set text "some text"
puts "the value of text is:$text."
```

As you can see, `$`*variable* refers to the value of *variable*.
Here is a command that performs mathematical operations:

`expr` *expression* – evaluates a mathematical expression

3 Try the `expr` command by entering the following lines in the VMD TkConsole window:

```
expr 3 - 8
set x 10
expr - 3 * $x
```

One of the most important aspects of Tcl is that you can embed Tcl commands into others by using brackets. A bracketed expression will automatically be substituted by the return value of the expression inside the brackets:

`[`*expr.*`]` – represents the result of the expression inside the brackets

4 Create some commands using brackets and test them. Try entering the following example in the VMD TkConsole window:

```
set result [ expr -3 * $x ]
puts $result
```

Often, one needs to execute a block of codes for many times. For this purpose, Tcl provides an iterated loop similar to the `for` loop in C. The `for` command in Tcl requires four arguments: an initialization, a test, an increment, and the block of code to evaluate. The syntax of the `for` command is:

`for` *{initialization} {test} {increment} {commands}*

5 Now let's calculate the values of $-3 * x$ for integers x from 0 to 10 and output the results into a file named `myoutput.dat`. Please also pay attention the way of writing the output to a file on disk.

```
set file [open "myoutput.dat" w]
for {set x 0} {$x <= 10} {incr x} {
puts $file [ expr -3 * $x ]
}
close $file
```

6 Take a look at the output file `myoutput.dat`, either by a text editor of your choice, or the command `less` in a terminal window on a Mac or Linux Machine.

3.2 VMD scripting

Anything that can be done in the VMD graphical interface can be done with text commands. This allows scripts to be written that can automatically load molecules, create representations, analyze data, make movies, etc. Here, we will go through some simple examples of what can be done using the scripting interface in VMD.

3.2.1 Loading molecules with text commands

1 In the VMD TkConsole window, type the command `mol new 1ubq.pdb` and hit enter. As you can see, this command performs the same function as described at the beginning of Section 1.1, namely, loading a new molecule with file name `1ubq.pdb`.

Navigating directories in TkConsole. If you see the error message `Unable to load file '1ubq.pdb' using file type 'pdb'`, you might not be in the `vmd-tutorial-files` directory. You can use the standard Unix commands in the VMD TkConsole window to move to the correct directory where 1ubq.pdb is located.

When you open VMD, by default a vmd console window appears. The vmd console window tells you what's going on within the VMD session that you are working on.

2 Take a look at the vmd console window (Fig. 22). It should tell you a molecule has been loaded, as well as some of its basic properties like number of atoms, bonds, residues and etc. The Tcl commands that you enter in the VMD TkConsole window can also be entered in the vmd console window. If you are using a Mac, your vmd console window is the terminal window that shows up when you open VMD.

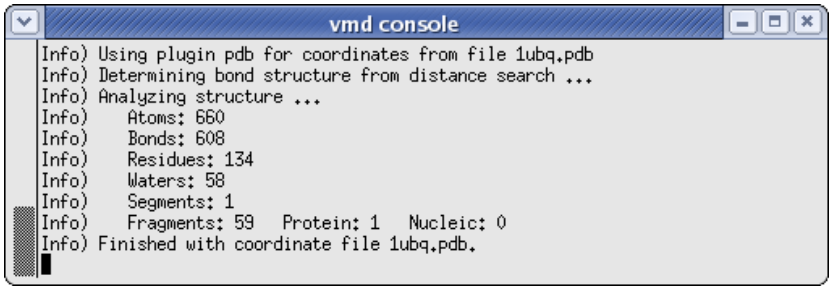


Figure 22: The VMD Console window.

3.2.2 The `atomselect` command

Many times you might want to perform operations on only a specific part a molecule. For this purpose, VMD's `atomselect` command is very useful.

`atomselect` *molid selection* – creates a new atom selection

This command allows you to select a specific part of a molecule. The first argument to `atomselect` is the molecule ID (shown to the very left of the VMD Main window), the second argument is a textual atom selection like what you have been using to describe graphical representations in Section 1.3. The selection returned by `atomselect` is itself a command which you will learn to use.

3 Type `set crystal [atomselect top "all"]` in the Tk Console window. This creates a selection, `crystal`, that contains all the atoms in the molecule and assigns it to the variable `crystal`. Instead of a molecule ID (which is a number), we have used the shortcut "`top`" to refer to the top molecule. A `top` molecules means that it is the target for scripting commands. This concept is particularly important when multiple molecules are loaded at the same time (see Section 4 for dealing with multiple molecules in VMD).

The result of `atomselect` is a function. Thus, `$crystal` is now a function that performs actions on the contents of the "all" selection.

3.2.3 Obtaining and changing molecule properties with text commands

After you have defined an atom selection, you have many commands that you can use to operate on it. For example, you can use commands to learn about the properties (number of atoms, coordinates, total charge, etc) of your atom

selection. You can also use commands to change its coordinates and other properties. See VMD User's Guide[2]for an extensive list of commands.

4 Type `$crystal num` in the Tk Console window. Passing `num` to an atom selection returns the number of atoms in that selection. Check that this number matches the number of atoms for your molecule displayed in the VMD Main window.

5 We can also use commands to move our molecule on the screen. You can use these commands to change atom coordinates.

```
$crystal moveby {10 0 0}
$crystal move [transaxis x 40 deg]
```

The following examples will show you how to edit atomic properties using VMD's `atomselect` command.

6 Open the Graphical Representation window by selecting Graphics → Representations... in the VMD Main window. Type in `protein` as the atom selection, change its Coloring Method to Beta and its Drawing Method to VDW. Your molecule should now appear as a mostly red and blue assembly of spheres.

The PDB B-factor field. The "B" field of a PDB file typically stores the "temperature factor" for a crystal structure and is read into VMD's "Beta" field. Since we are not currently interested in this information, we can recycle this field to store our own numerical values. VMD has a "Beta" coloring method, which colors atoms according to their B-factors. By replacing the Beta values for various atoms, you can control the color in which they are drawn. This is very useful when you want to show a property of the system that you have computed.

7 Return to the Tk Console window, and type `$crystal set beta 0`. This resets the "beta" field (which is displayed) to zero for all atoms. As you do this, you should observe that the atoms in your OpenGL window will suddenly change to a uniform color (since they all have the same beta values now).

Examples of atomic properties. You can obtain and set many atomic properties using atom selections, including segment, chain, residue, atom name, position (x, y and z), charge, mass, occupancy and radius, just to name a few.

[2]VMD documentation can be found at http://www.ks.uiuc.edu/Research/vmd/current/docs.html

Atom selections are just references to the atoms in the original molecule. When you change a property (e.g. beta value) of some atoms through a selection, that change is reflected in all the other selections that contain those atoms.

8 In the Tk Console window, type `set sel [atomselect top "hydrophobic"]`. This creates a selection, `sel`, that contains all the atoms in the hydrophobic residues.

9 Let's label all hydrophobic atoms by setting their beta values to 1. You probably know how to do this by now: type `$sel set beta 1` in the Tk Console window. If the colors in the OpenGL Display do not get updated, go to the Graphical Representations window and click on the Apply button at the bottom.

10 You will now change a physical property of the atoms to further illustrate the distribution of hydrophobic residues. In the Tk Console window type `$crystal set radius 1.0` to make all the atoms smaller and easier to see through, and then `$sel set radius 1.5` to make atoms in the hydrophobic residues larger. The radius field affects the way that some representations (e.g., VDW, CPK) are drawn.

You have now created a visual state that clearly distinguishes which parts of the protein are hydrophobic and which are hydrophilic. If you have followed the instructions correctly, your protein should resemble Fig. 23.

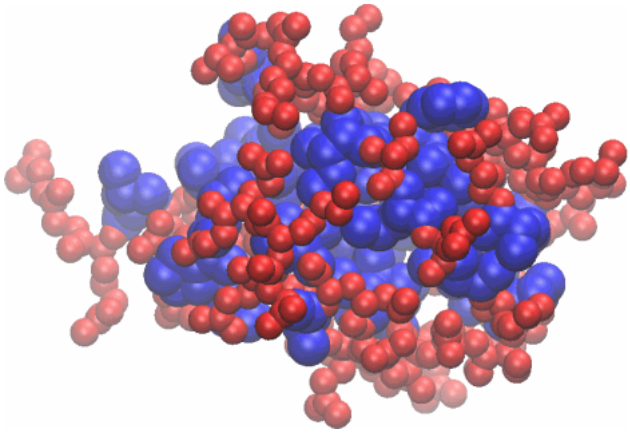


Figure 23: Ubiquitin in the VDW representation, colored according to the hydrophobicity of its residues.

Identifying hydrophobic residues. Many times in studies of proteins it is important to identify the locations of the hydrophobic residues, as they often have a functional implication. The method you have just learned is useful in this task. For example, you can see easily that in ubiquitin, the hydrophobic residues are almost exclusively contained in the inner core of the protein. This is a typical feature for small water-soluble proteins. As the protein folds, the hydrophilic residues will have a tendency to stay at the water interface, while the hydrophobic residues are pushed together and play a structural role. This helps the protein achieve proper folding and increases its stability.

Atom selections are useful not only for setting atomic data, but also for getting atomic information. Let's say that you wish to communicate which residues are hydrophobic, all you need to do is to create a hydrophobic selection and use `get` command.

11 Try to use `get` command with your `sel` atom selection to obtain the names of hydrophobic residues:

```
$sel get resname
```

But there is a problem! Each residue contains many atoms, resulting in multiple repeated entries. Can you think of a way to circumvent this? We know that each amino acid has the same backbone atoms. If you pick only one of these atoms per residue, each residue will be present only once in your selection.

12 Let's try this solution. Each residue has one and only one α-carbon (`name CA` = `alpha`), so type the following in the Tk Console window:

```
set sel [atomselect top "hydrophobic and alpha"]
$sel get resname
```

This should give you the list of hydrophobic residues.

13 You can also get multiple properties simultaneously. Try the following:

```
$sel get resid
$sel get {resname resid}
$sel get {x y z}
```

If you want to obtain some of the structural properties, e.g., the geometric center or the size of a selection, the command `measure` can do the job easily.

14 Let's try using `measure` with the `sel` selection:

```
measure center $sel
measure minmax $sel
```

The first command above returns the geometric center of atoms in `sel`. And the second command returns two vectors, the first containing the minimum x, y, and z coordinates of all atoms in `sel`, and the second containing the corresponding maxima.

15 Once you are done with a selection, it is always a good ideal to delete it to save memory:

```
$sel delete
```

3.2.4 Sourcing scripts

We have learned many useful commands in VMD. When performing a task that requires many lines of commands, instead of typing each line in the Tk Console window, it is usually more convenient to write all the lines into a script file and load it in VMD. This is very easy to do. Just use any text editor to write your script file, and in a VMD session, use the command `source` *filename* to execute the file. In the `vmd-tutorial-file`, you will find a simple script file `beta.tcl`, which we will execute in VMD as an example. The script `beta.tcl` sets the colors of residues LYS and GLY to a different color from the rest of the protein by assigning them a different beta value, a trick you have also learned in Section 3.2.3.

16 In the Tk Console window, type `source beta.tcl` and observe the color change. You should see that the protein is mostly a collection of red spheres, with some residues shown in blue. The blue residues are the LYS and GLY residues in the ubiquitin.

17 Let's take a quick look at the `beta.tcl`. Use any text editor of your choice, open the file `beta.tcl`. You can see that there are six lines in this file, and each line represents a Tcl command line that you have used before. Close the text editor when you are done.

A vmd saved state is a script file. The .vmd file you saved in Section 1.5 is actually a series of commands. You are encouraged to take a look at that file using a text editor. Hopefully, by the end of this section, you'll understand many of those commands. In fact, you can execute the file at Tk Console the same way as you execute other script files, i.e., by typing `source myfirststate.vmd` in the Tk Console window.

The `logfile console` command. Many times you might want to look up the command for an interactive VMD feature. You can either find it in the VMD User's Guide[3], or use the `logfile console` command. Try typing `logfile console` in your Tk Console window. This creates a logfile for all your actions in VMD and writes them in the Tk Console window as command lines. If you execute those command lines you can repeat the exact same actions you have performed interactively. To turn off logfile, type `logfile off`.

vmdrc. Many times, you would like to automatically load certain scripts or packages upon starting VMD. VMD has a preferences file `.vmdrc` in your home directory (Windows uses the file `vmd.rc`) for this purpose. You can also change the default behavior of VMD here (e.g., change the background color) and add atom selection macros. VMD will look for this file upon startup and will recognize all your scripts, macros, etc.. For more information on the VMD startup files, refer to the VMD user's guide.

3.3 Drawing shapes

VMD offers a way to display user-defined objects built from graphics primitives such as points, lines, cylinders, cones, spheres, triangles, and text. The command that can realize those functions is `graphics`, the syntax of which is

`graphics` *molid command*

Where *molid* is a valid molecule ID and *command* is one of the commands shown below. Let's try drawing some shapes with the following examples.

1 Hide all representations in the Graphics tab.

2 Let's draw a point. Type the following command in your Tk Console window:

```
graphics top point {0 0 10}
```

Somewhere in your OpenGL window there should be a small dot.

3 Let's draw a line. Type the following command in your Tk Console window:

```
graphics top line {-10 0 0} {0 0 0} width 5 style solid
```

This will give you a solid line.

[3] VMD documentation can be found at http://www.ks.uiuc.edu/Research/vmd/current/docs.html

4 You can also draw a dashed line:

```
graphics top line {10 0 0} {0 0 0} width 5 style dashed
```

5 All the objects drawn so far are all in blue. You can change the color of the next graphics object by using the command `graphics top color` *colorid*. The *colorid* for each color can be found in Colors → ColorID from the Graphics tab. For example, the color for orange is "3". Type `graphics top color 3` in the Tk Console window, and the next object you draw will appear in orange.

6 Try the following commands to draw more shapes:

```
graphics top cylinder {15 0 0} {15 0 10} radius 10 resolution 60 filled no
graphics top cylinder {0 0 0} {-15 0 10} radius 5 resolution 60 filled yes
graphics top cone {40 0 0} {40 0 10} radius 10 resolution 60
graphics top sphere {65 0 5} radius 10 resolution 80
graphics top triangle {80 0 0} {85 0 10} {90 0 0}
graphics top text {40 0 20} "my drawing objects"
```

7 On your OpenGL window, there are a lot of objects now. To find the list of objects you've drawn, use the command `graphics top list`. You'll get a list of numbers, standing for the ID of each object.

8 The detailed information about each object can be obtained by typing `graphics top info` *ID*. For example, type `graphics top info 0` to see the information on the point you drew.

9 You can also delete some of the unwanted objects using the command `graphics top delete` *ID* .

10 Using these basic shape-drawing commands, you can create geometric objects, as well as texts, to be displayed in your OpenGL window. When you render an image (as discussed in Section 1.6.5), these objects will be included in the resulting image file. You can hence use geometric objects and texts to point or label interesting features in your molecule. When you are done, quit VMD.

4 Data Analysis in VMD

VMD is a powerful tool for analysis of structures and trajectories. Numerous tools for analysis are available under the VMD Main menu item Extensions →Analysis. In addition to these built-in tools, VMD users often use custom-written scripts to analyze desired properties of the simulated systems. VMD Tcl scripting capabilities are very extensive, and provide boundless opportunities for analysis. In this following section, we will learn how to use built-in VMD features for standard analysis, as well as consider a simple example of scripting.

4.1 Labels

Labels can be placed in VMD to get information on a particular selection, to be used during visualization and quantitative analysis. Labels are selected with the mouse and can be accessed in Graphics → Labels menu. We will cover labels that can be placed on atoms and bonds, although angle and dihedral labeling are also available. In this context, labels for "bonds" or "angles" actually mean distances between two atoms or angles between three atoms; the atoms do not have to be physically connected by bonds in the molecule.

1 Start a new VMD session. Load the ubiquitin trajectory into VMD (using the files `ubiquitin.psf` and `pulling.dcd`). For graphical representation, display protein only, using New Cartoon for drawing method and Structure for coloring method. If you need help, check Section 2.1, steps 1-5.

2 Choose the Mouse → Labels → Atoms menu item from the VMD Main menu. The mouse is now set to the mode for displaying atom labels. You can click on any atom on your molecule and a label will be placed for this atom. Clicking again on it will erase the label.

3 We will now try the same for bonds. Choose the Mouse → Label → Bonds menu item from the VMD Main menu. This selects the "Display Label for Bond" mode.

We will consider the distance between the α carbon of Lysine 48 and of the C terminus. In the pulling simulation, the former is kept fixed, and the latter is pulled at a constant force of 500 pN. In reality, polyubiquitin chains can be linked by a connection between the C terminus of one ubiquitin molecule and the Lysine 48 of the next. The simulation then mimics the effect of pulling on the C terminus with this kind of linkage.

4 We will make a VDW representation for the α carbons of Lysine 48 and of the C terminus. To find out the index of these atoms, make a selection including these two atoms, by typing in the Tk Console window (to open

the Tk Console window, select Extensions → Tk Console in the VMD Main menu.):
`set sel [atomselect top "resid 48 76 and name CA"].`

5 Get the indices by typing the following line in the Tk Console window:
`$sel get index`
This command should give the indices `770 1242`.

PDB and VMD atom numbering. Note that the atom number of these atoms in the pdb file is 771 and 1243. VMD starts counting the `index` from zero, so the text in the Representation should be the numbers that VMD understands. This is only the case for index, since VMD does not read them from the PDB file. Other keywords, such as residue, are consistent with the PDB file.

6 In the Graphical Representations window, create a representation for the selection `index 770 1242`, with VDW as drawing method.

7 Now that you can see the two α carbons, choose the Mouse → Label → Bonds menu item from the VMD Main menu. Click on each atom one after the other. You should get a line connecting the two atoms (Fig. 24). The number appearing next to the line is the distance between the two atoms in Ångstroms. Note that the appearance of the line (its color), as well as appearance of essentially all other objects in VMD, can be changed in Graphics → Colors in the VMD Main menu.

Labels. The shortcut keys for labels are 1: Atoms and 2: Bonds. You can use these instead of the Mouse menu. Be sure the Open GL Display window is active when using these shortcuts.

8 The value of the distance displayed corresponds to the current frame. Try playing the trajectory - you will see that the label is modified automatically as the distance between the atoms changes.

9 The labels can be used not only for display, but also for obtaining quantitative information. In VMD Main menu, select Graphics → Labels. On the top left-hand side of the window, there is a pull-down menu where you can choose the type of label (Atoms, Bonds, Angles, Dihedrals). For now, keep it in Atoms. You can see the list of atoms for which you have made a label.

10 Click on one of the atoms. You can see all the information of the atom displayed on the bottom half of the Labels window. This information is useful to make selections; it corresponds to the current frame, and is updated as the frame is changed.

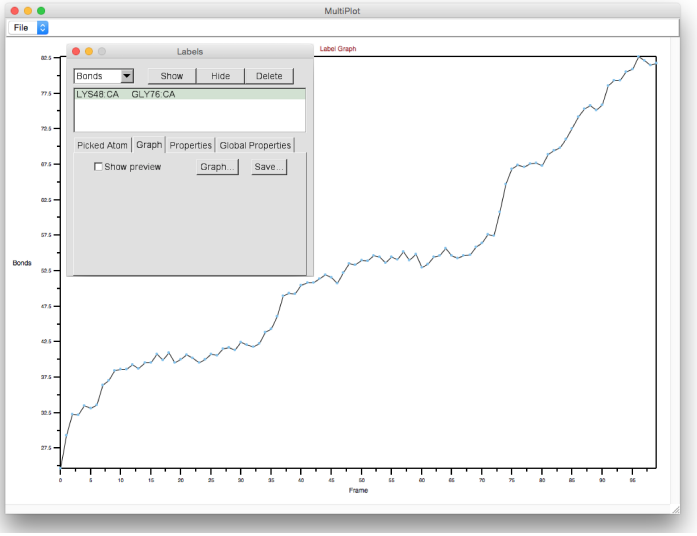


Figure 24: Labels in VMD. Label control is available in the Graphics → Labels menu item, using which, one can, e.g., plot the labeled distance as a function of time.

11 You can also delete, hide, or show the atom label by clicking on the corresponding button on the top of the Labels window.

12 Now, in the Labels window, choose the label type Bonds, and select the "bond" (distance) you labeled (Fig. 24). The information given corresponds to only the first atom in the bond, but the number in the Value field corresponds to the length of the bond in Ångstroms.

13 Click on the Graph tab. Select the bond you labeled between atoms 770 and 1242. Click on the Graph button. This will create a plot of the distance between these two atoms over time (Fig. 24). You can also save this data to a file by clicking on the Save button, and then use an external plotting program to visualize the data.

14 Quit VMD.

4.2 Example of a built-in analysis tool: the RMSD Trajectory Tool

The built-in analysis tools in VMD are available under the menu item Extensions →Analysis. These tools each features a GUI window that allow one to enter parameters and customize the quantities analyzed. In addition, all tools

can be invoked in a scripting mode, using the TkConsole window. We will learn how to work with one of the most frequently used tools, the RMSD Trajectory Tool.

Root Mean Square Deviation. The Root Mean Squared Deviation (RMSD) is defined as :

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N_{atoms}} (r_i(t_1) - r_i(t_2))^2}{N_{atoms}}} \quad (1)$$

where N_{atoms} is the number of atoms whose positions are being compared, and $r_i(t)$ is the position of atom i at time t.

In this example, we will analyze RMSD for two trajectories for the same system, `ubiquitin.psf`. One of them is the already familiar pulling trajectory, `pulling.dcd`, and the other is the trajectory of a simulation in which no force was applied to the protein, `equilibration.dcd`.

1 Start a new VMD session. Load the ubiquitin equilibration trajectory into VMD (using the files `ubiquitin.psf` and `equilibration.dcd`).

2 Choose Extensions → Analysis → RMSD Trajectory Tool in the VMD Main window (Fig. 25). The RMSD Trajectory Tool window will show up.

3 In the RMSD Trajectory Tool window, you can see many customizations can be made. For the default values, the molecule to be analyzed is `ubiquitin.psf` (the only one loaded). The selection for which RMSD will be computed is all of the `protein` atoms, excluding hydrogens (since thenoh checkbox is on). The RMSD will be calculated for each frame (time t_1 in Eq. 1) with the reference to frame 0 (time t_2 in Eq. 1). Make sure the Plot checkbox is selected.

4 Click the Align button. This will align each frame of the trajectory with respect to the reference frame (in this case, frame 0) to minimize RMSD, by applying only rigid-body translations and rotations. This step is not necessary, but is desirable in most cases, because we are interested only in RMSD that arises from the fluctuations of the structure and not from the displacements and rotations of the molecule as a whole. The result of the alignment can be seen in the OpenGL display.

5 Now, click the RMSD button in the RMSD Trajectory Tool window. The protein RMSD (in Ångstroms) vs. frame number is displayed in a plot (Fig. 25).

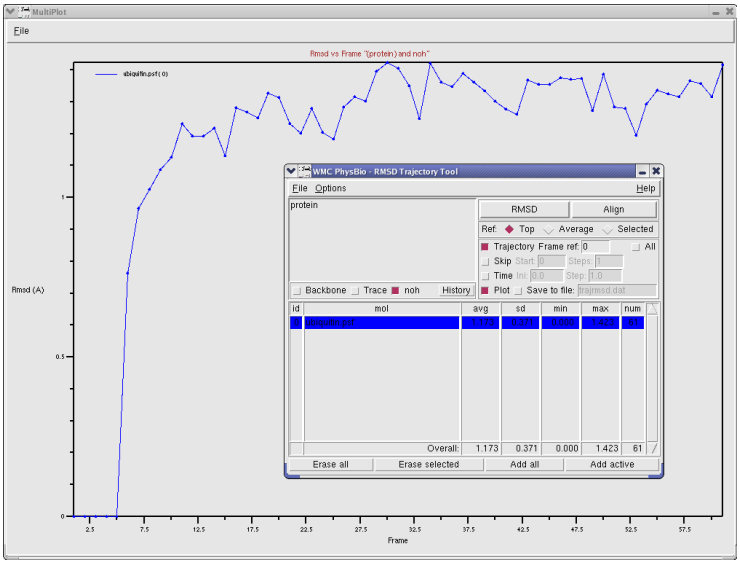


Figure 25: RSMD Trajectory Tool. The RMSD is plotted for the equilibration of ubiquitin.

RMSD plot. Over the several initial frames, RMSD= 0 because positions of the protein atoms are fixed during that time in the simulation, to allow water molecules around the protein to adjust to the protein surface. After that, the protein is released, and RMSD grows quickly up to around 1.5 Å. At that point, RMSD levels off and remains at ∼1.5 Å further on. This is a typical behavior for MD simulations. Leveling of the RMSD means that the protein has relaxed from its initial crystal structure (which is affected by crystal packing and usually misses some atoms, e.g., hydrogens) to a more stable one. Production MD simulations are usually preceded by such equilibration runs, where the protein is allowed to relax; the process is monitored by checking RMSD vs. time, and equilibration is assumed to be sufficient when RMSD levels off. The RMSD of 1.5 Å is quite a good value, totally acceptable for most protein simulations. Usually, the deviations from the crystal structure in a simulation are due to the thermal motion and to the relaxation process mentioned; imperfections of the simulation force-fields can contribute as well.

6 We will now work with the other trajectory. Load the pulling trajectory into VMD using the files `ubiquitin.psf` and `pulling.dcd`. Make sure you load `ubiquitin.psf` as a new molecule. You can change the names of the molecules by double-clicking on them in the VMD Main menu (see

Section 5.1.2).

7 In the RMSD Trajectory Tool window, hit the button Add all to update the list of molecules.

8 Click the Align button, and then click RMSD. The new graph (Fig. 26) displays two RMSD plots vs. time, one for the equilibration trajectory, and the other for the pulling trajectory. The pulling RMSD does not level off and is much higher than the equilibration RMSD, since the protein is stretched in the simulation.

9 Quit VMD.

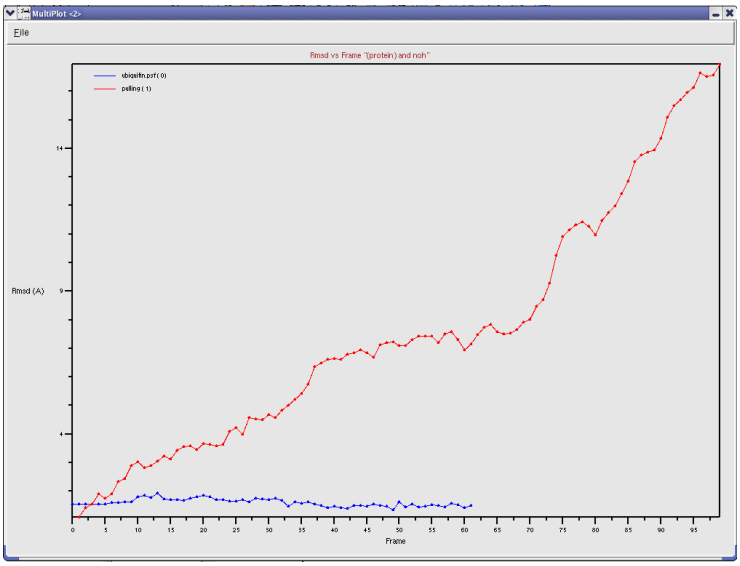


Figure 26: RMSD vs. time for the equilibration (blue) and pulling (red) trajectories of ubiquitin.

4.3 Example of an analysis script

In many cases, one requires special types of trajectory analysis that are tailored for certain needs. The Tcl scripting in VMD provides opportunities for such custom tasks. Users commonly write their own scripts to analyze the features of interest, and a very extensive library of VMD scripts, contributed by many users, is available online, at `http://www.ks.uiuc.edu/Research/vmd/script_library/`.

We will investigate a very simple exemplary script, `distance.tcl`, which computes the distance between two atom selections vs. time and also distribution of the distance.

1 Start a new VMD session. Load the ubiquitin equilibration trajectory (files `ubiquitin.psf` and `equilibration.dcd`).

2 Open the TkConsole window by selecting Extensions → Tk Console in the VMD Main menu.

3 In the the TkConsole window, load the script into VMD by typing: `source distance.tcl` (make sure that the file `distance.tcl` is in the current folder). This will load the procedure defined in `distance.tcl` into the VMD.

4 One can now invoke the procedure by typing `distance` in the the TkConsole window. In fact, the correct usage is
`distance seltext1 seltext2 N_d f_r_out f_d_out`
where `seltext1` and `seltext2` are the selection texts for the groups of atoms between which the distance is measured, `N_d` is the number of bins for the distribution, and `f_r_out` and `f_d_out` are the file names to where the output distance vs. time and distance distribution will be written.

5 Open the script file `distance.tcl` with a text editor. You can see that the script does the following:

- Choose atom selections
```
set sel1 [atomselect top "$seltext1"]
set sel2 [atomselect top "$seltext2"]
```

- Get the number of frames in the trajectory and assign this value to the variable `nf`
```
set nf [molinfo top get numframes]
```

- Open file specified by the variable `f_r_out`
```
set outfile [open $f_r_out w]
```

- Loop over all frames
```
for {set i 0} {$i < $nf} {incr i} {
```

- Write out the frame number and update the selections to the current frame

```
puts "frame $i of $nf"
$sel1 frame $i
$sel2 frame $i
```

- Find the center of mass for each selection (`com1` and `com2` are position vectors)
```
set com1 [measure center $sel1 weight mass]
set com2 [measure center $sel2 weight mass]
```

- At each frame `i`, find the distance by subtracting one vector from the other (command `vecsub`) and computing the length of the resulting vector (command `veclength`), assign that value to an array element `simdata($i.r)`, and print a frame-distance entry to a file
```
set simdata($i.r) [veclength [vecsub $com1 $com2]]
puts $outfile "$i $simdata($i.r)"
}
```

- Close the file
```
close $outfile
```

- The second part of the script is for obtaining the distance distribution. It starts from finding the maximum and minimum values of the distance
```
set r_min $simdata(0.r)
set r_max $simdata(0.r)
for {set i 0} {$i < $nf} {incr i} {
set r_tmp $simdata($i.r)
if {$r_tmp < $r_min} {set r_min $r_tmp}
if {$r_tmp > $r_max} {set r_max $r_tmp}
}
```

- The step over the range of distances is chosen based on the number of bins `N_d` defined in the beginning, and all values for the elements of the distribution array are set to zero
```
set dr [expr ($r_max - $r_min) /($N_d - 1)]
for {set k 0} {$k < $N_d} {incr k} {
set distribution($k) 0
}
```

- The distribution is obtained by adding 1 (`incr ...`) to an array

element every time the distance is within the respective bin
```
for {set i 0} {$i < $nf} {incr i} {
set k [expr int(($simdata($i.r) - $r_min) / $dr)]
incr distribution($k)
}
```

- Write out the file with the distribution
```
set outfile [open $f_d_out w]
for {set k 0} {$k < $N_d} {incr k} {
puts $outfile "[expr $r_min + $k * $dr] $distribution($k)"
}
close $outfile
```

6 Now run the script by typing in the TkConsole window
`distance "protein" "protein and resid 76" 10 res76-r.dat res76-d.dat`
This will compute the distance between the center of the protein and center of the terminal residue 76, and write the distance vs. time and its distribution to files `res76-r.dat` and `res76-d.dat`.

7 Repeat the same for the protein's residue 10 by typing in the TkConsole window
`distance "protein" "protein and resid 10" 10 res10-r.dat res10-d.dat`
The data in files produced by the script `distance.tcl` are in two-column format. Compare the outputs for residue 76 and 10 using your favorite external plotting program (Fig. 27).

Residue 76 is at the protein's C-terminus, which is extended towards the solvent and is quite flexible, while residue 10 is at the surface of the globular part of ubiquitin. The difference in their dynamics with respect to the rest of the protein is immediately obvious when our newly obtained data are plotted (Fig. 27): the distance of residue 76 from the protein's center is substantially greater than that of residue 10, and the distribution of the distance is noticeably wider due to the flexibility of the C-terminus. This is just a simple example of scripting for the analysis of a trajectory. Similar, but usually much more complex, customized scripts are routinely employed by VMD users to perform many kinds of analysis.

8 Quit VMD.

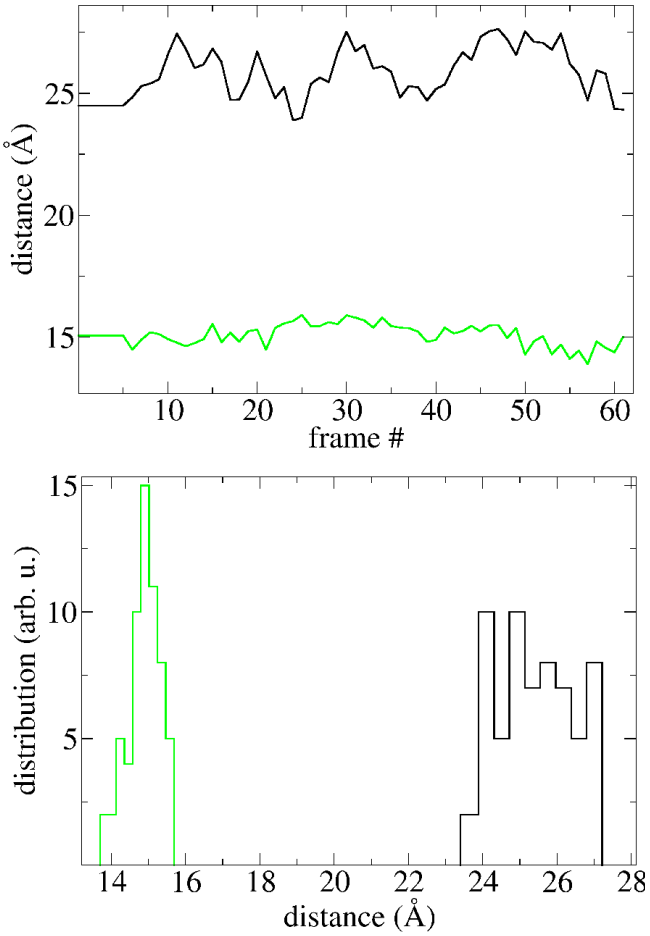


Figure 27: Distance between a residue and the center of ubiquitin. The distances analyzed are those for residue 76 (black) and residue 10 (green).

5 Working with Multiple Molecules

In this section you will learn to deal with multiple molecules within one VMD session. We will use the water transporting channel protein, aquaporin, as an example.

5.1 Main Menu Molecule List Browser

Aquaporins are membrane channel proteins found in a wide range of species, from bacteria to plants to human. They facilitate water transport across the cell membrane, and play an important role in the control of cell volume and transcellular water traffic. Many aquaporin protein structures are available in the Protein Data Bank, including the human aquaporin (PDB code 1FQY; Murata et al., *Nature*, **407**:599, 2000) and *E. coli* aquaporin (PDB code 1RC2; Savage et al., *PLoS Biology*, **1**:E72, 2003). To practice dealing with multiple proteins with VMD, let's load both aquaporin structures.

5.1.1 Loading multiple molecules

1 Start a new VMD session. In the VMD Main window, choose File → New Molecule.... The Molecule File Browser window should appear on your screen.

2 Use the Browse... button to find the file `1fqy.pdb` in `vmd-tutorial-files` in the tutorial directory. When you select the file, you will be back in the Molecule File Browser window. Press the Load button to load the molecule. The coordinate file of human aquaporin AQP1 should now be loaded and can be seen in the OpenGL window.

3 The Molecule File Browser window should still be open; if not open it through File → New Molecule... again. Make sure you choose `New Molecule` in the `Load files for:` pull-down menu on the top. Use the Browse... button to find the file `1rc2.pdb` in `vmd-tutorial-files` directory and press Load. Close the Molecule File Browser window.

You have just loaded a second molecule; any number of molecules may be loaded and displayed in VMD simultaneously by repeating the previous step. VMD can load as many molecules as the memory of your computer allows.
Take a look at your VMD Main window, which should look like Fig. 28. Within the VMD Main menu you can find the Molecule List Browser (circled in Fig. 28), which shows the global status of the loaded molecules. The Molecule List Browser displays information about each molecule, including Molecule ID (`ID`), the four Molecule Status Flags (`T`, `A`, `D`, and `F`, which stand for Top, Active, Drawn, and Fixed), name of the molecule (`Molecule`), number of atoms in the

molecule (`Atoms`), number of frames loaded in the molecule (`Frames`), and the volumetric data loaded (`Vol`).

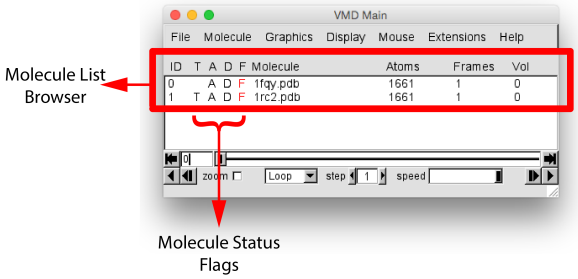


Figure 28: The Molecule List Browser.

5.1.2 Changing molecule names

Let's first start with the `Molecule` column. By default the `Molecule` column displays file names of the molecules loaded in VMD, but you can change the molecule names to recognize them more easily.

4 In the VMD Main menu, double-click on `1fqy.pdb` in the `Molecule` column. A window will pop up with the message "Enter a new name for molecule 0:" (Fig. 28a). Type in "`human aquaporin`", and click `OK` (or press enter). In the VMD Main menu, the first molecule now has the name "`human aquaporin`".

5 Repeat the previous step for the *E. coli* aquaporin by double-clicking the `1rc2.pdb` molecule name, and changing it to "`E. coli aquaporin`" in the pop-up window. Your VMD Main window should now look like Fig. 29b.

5.1.3 Drawing different representations for different molecules

Before we continue exploring other features in the Molecule List Browser, take a look at your OpenGL Display window. You have two aquaporin structures, but since they are both shown in the same default representation, it is difficult to distinguish them. To tell them apart, you can assign them different representations.

6 Open the Graphical Representations window via Graphics → Representations... from the VMD Main menu. Make sure 0:human aquaporin is

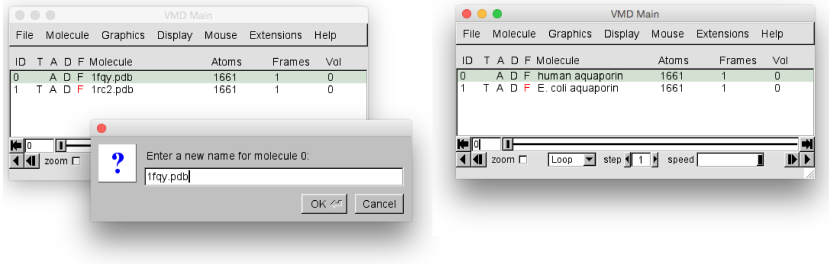


Figure 29: Changing molecule names.

selected in the Selected Molecule pull-down menu on top. Select New Cartoon for Drawing method, and ColorID → 1 red for Coloring Method.

7 In the Graphical Representations window, select 1:E. coli aquaporin in the Selected Molecule pull-down menu on top. Select New Cartoon for Drawing method, and ColorID → 4 yellow for Coloring Method. Close the Graphical Representations window.

Now your OpenGL Display window should show a human aquaporin colored in red and an *E. coli* aquaporin colored in yellow (Fig. 30).

5.1.4 Molecule Status Flags

In your OpenGL Display window, try moving the aquaporins around with your mouse in different mouse modes (rotating, scaling, and translating). You can see that both aquaporins move together. You can fix any molecule by double-clicking the *F* (fixed) flag in the Molecule List Browser on the left of the molecule name.

8 In the Molecule List Browser, double-click on the `F` flag on the left of `human aquaporin` to fix the human aquaporin molecule. Return to the OpenGL Display window and toggle your mouse around. You can see that only the yellow *E. coli* aquaporin moves. Double-click on the `F` flag for human aquaporin again to release it.

One thing to notice about the `F` flag is that, although it may seem that one molecule has been moved relative to another when one of the molecules is fixed, the difference is only apparent. The internal coordinates of molecules are not

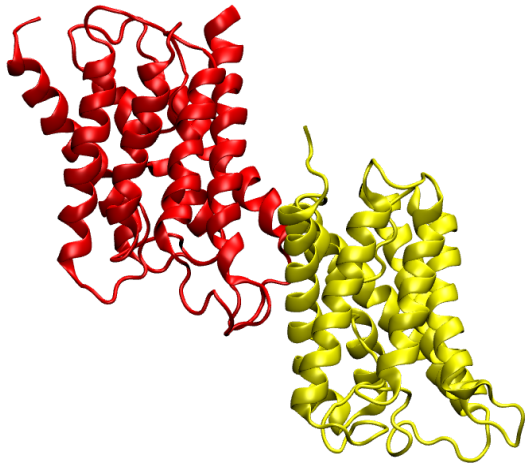


Figure 30: The two aquaporins drawn in different representations.

changed by the rotation, translation and scaling motions. To change the coordinates of atoms in a molecule you need to use the text command interface (discussed in Section 3.2.3), and by using the atom move picking modes (by choosing Mouse → Move in the VMD Main menu).
Other features in the Molecule List Browser includes the Molecule ID (`ID`), Top (`T`), Active (`A`), and Drawn (`D`). Molecule ID is a number (starting from `0`) assigned to each molecule when it's loaded into VMD, and is how VMD recognizes each molecule internally. You also refer to molecules by their Molecule IDs in text command interface. Top flag (`T`) indicates the default molecule in VMD operations, for example when resetting the VMD OpenGL view and when playing molecule trajectories. There can be only one top molecule at a time. Active flag (`A`) indicates if the trajectory of the given molecule is updated when using animation tools described in Section 2. Finally, Drawn flag (`D`) indicates if the given molecule is displayed in the OpenGL window. Let's try out the Top and Drawn flags.

9 Make sure no molecule is fixed. By default the last molecule loaded in the VMD is the top molecule, so you can check and see that there is a `T` displayed for the *E. coli* aquaporin in the VMD Main menu. Reset the view by pressing "=" in the OpenGL Display window. Note that the yellow *E. coli* aquaporin is now placed in the center of the OpenGL Display window.

10 Switch the top molecule by double-clicking on the empty `T` flag for the human aquaporin molecule in the VMD Main menu. A `T` should appear

for the human aquaporin, while the `T` for *E. coli* disappears. Go to the OpenGL Display window and reset the view again. You can see that this time the red human aquaporin is placed in the center of the OpenGL Display window.

11 In the VMD Main menu, try hiding a molecule by double-clicking on its `D` flag. You can display the molecule again by double-clicking its `D` flag again.

5.2 Aligning Molecules with the `measure fit` Command

When you look at your OpenGL Display window, you can see that the two aquaporins are very similar in structure. But it is difficult to detect their slight structural differences as the two proteins are placed apart. We will now try out a very useful Tcl command `measure fit` to align two molecules.

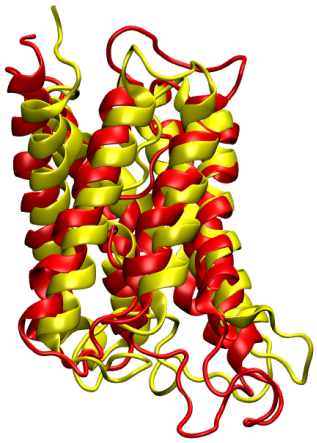


Figure 31: Result of the alignment between the two aquaporins using the `measure fit` command.

1 Open the VMD TkConsole window by choosing Extension → TkConsole from the VMD Main menu, and input the following commands (hit Enter after each line):

```
set sel0 [atomselect 0 all]
set sel1 [atomselect 1 all]
set M [measure fit $sel0 $sel1]
$sel0 move $M
```

> `measure fit` *selection1 selection2*
> – measures the transformation matrix that best aligns the coordinates of selection1 with the coordinates of selection2

As soon as you enter the last command line, you can see that the two aquaporins are now overlapping (Fig. 31). The α-helical regions of the aquaporins agree very well, with bigger deviations in the loop regions. Note the `measure fit` command can only work if two molecules have the same number of atoms. In this case it's a pure coincidence that the human aquaporin and *E. coli* aquaporin PDB files have the same number of atoms. The `measure fit` command is hence most useful in aligning the same protein in different conformations or different frames of a molecular dynamics simulation trajectory. Generally, to compare the structures of different proteins, one needs to use a different method. A good tool is the MultiSeq VMD plugin, which we will discuss in the following section.

2 Quit VMD.

6 Comparing Structures and Sequences with MultiSeq

MultiSeq (Roberts et al., *BMC Bioinformatics*, **7**:382, 2006) is a bioinformatics analysis environment developed in the Luthey-Schulten Group at the University of Illinois in Urbana-Champaign. MultiSeq allows users to organize, display, and analyze both sequence and structure data for proteins and nucleic acids[4], and has been incorporated in VMD as a plugin tool starting with VMD version 1.8.5. In this section you will learn how to compare protein structures and sequences with the VMD MultiSeq plugin. We will again use the water transporting channel protein, aquaporin, as an example.

6.1 Structure Alignment with MultiSeq

Very often comparing structures of different proteins reveal many important information. For example, proteins with similar functions tend to be found with similar structural features. MultiSeq structure alignment is useful for this reason. We will compare the structures of four aquaporin proteins, whose coordinate files can all be found in the `vmd-tutorial-files` directory.

PDB code	Description
`1fqy` (Murata et al., *Nature*, **407**:599, 2000)	Human AQP1
`1rc2` (Savage et al., *PLoS Biology*, **1**:E72, 2003)	*E. coli* AqpZ
`1lda` (Tajkhorshid et al., *Science*, **296**:525, 2002)	*E. coli* Glycerol Facilitator (GlpF)
`1j4n` (Sui et al., *Nature*, **414**:872, 2001)	Bovine AQP1

Table 8: The four aquaporins used in this section.

6.1.1 Loading aquaporin structures

1 Start a new VMD session. Open the Molecule File Browser window by choosing the File → New Molecule... menu item in the VMD Main window. In the Molecule File Browser window, use the Browse... button to find and select the file `1fqy.pdb` in the directory `vmd-tutorial-files`. Press Load to load the molecule.

2 Load the remaining aquaporins, `1rc2`, `1lda`, and `1j4n`. Make sure that each PDB is loaded into a new molecule. Close the Molecule File Browser window when you finished loading all four molecules. Your VMD Main menu should look like Fig. 32 when all four aquaporins are loaded.

[4]More information on MultiSeq can be found in its plugin documentation page http://www.ks.uiuc.edu/Research/vmd/plugins/multiseq/

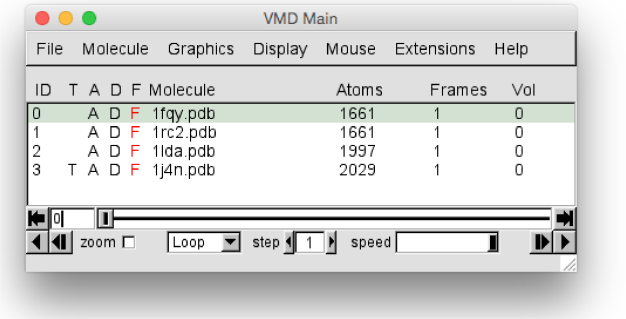


Figure 32: VMD Main menu after loading the four aquaporins.

6.1.2 Aligning the molecules

3 Within the VMD main window, choose the Extensions menu and select Analysis → MultiSeq.

The Multiseq window (with window name `untitled.multiseq` showing on top) should now be open. You may be asked to update some databases in a pop-up window if this is the first time you use MultiSeq. If this is the case, simply click Yes and wait for MultiSeq to finish downloading. When MultiSeq starts, your Multiseq window should look like Fig. 33, with a list of the four aquaporin protein structures and a list of two non-protein structures. The non-protein structures are detergent molecules used in crystallizing the aquaporin proteins, and will not be needed for structure or sequence alignment. You can tell MultiSeq to throw away molecules you are not interested in.

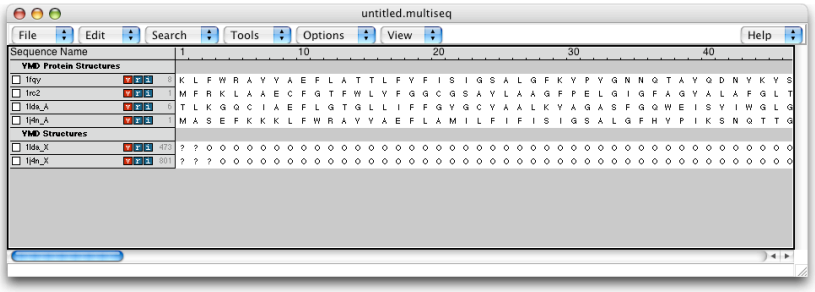


Figure 33: The MultiSeq window.

4 In the Multiseq window, select the `1lda_X` detergent molecule by clicking on it. This will highlight the entire row of `1lda_X`. Remove it from MultiSeq by pressing the delete or Backspace key on your keyboard. Do the same to remove the `1j4n_X` detergent molecule.

MultiSeq uses the program STAMP (Russell et al., *Proteins: Struct., Func., Gen.*, **14**:309, 1992) to align protein molecules. STAMP (Structural Alignment of Multiple Proteins) is a tool for aligning protein sequences based on a three-dimensional structure. Its algorithm minimizes the C_α distance between aligned residues of each molecule by applying globally optimal rigid-body rotations and translations. Note that you can only perform alignments on molecules that are structurally similar; if you try to align proteins that have no common structures, STAMP will have no means of aligning them.

5 In the Multiseq window, select Tool → Stamp Structural Alignment. This will open the Stamp Alignment Options window.

6 In the Stamp Alignment Options window, choose Align the following: All Structures and go to the bottom of the menu and press OK.

The molecules have been aligned. You can see the alignment both in the OpenGL window and in the MultiSeq window (Fig. 34). Your alignment in OpenGL window will not immediately resemble Fig. 34. When MultiSeq completes an alignment, it creates a new representation for all the aligned protein in the NewCartoon representation with the same default coloring method and hides all other representations created previously. Let's give different colors to different aquaporins to distinguish them.

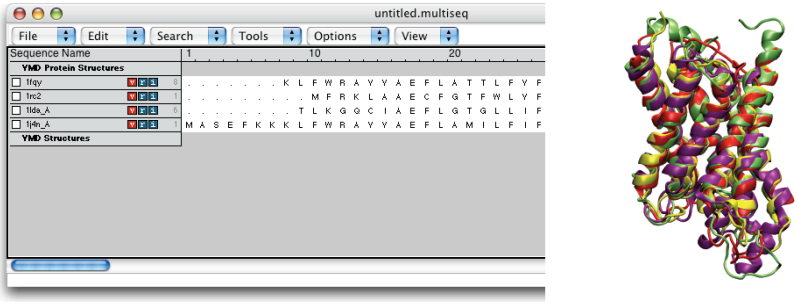


Figure 34: The four aquaporins aligned according to their structural similarity.

7 Open your Graphical Representations window, and you should see two representations for each molecule, one on top created when VMD loaded the

molecule (which is now hidden), and one on the bottom created automatically by MultiSeq. Select 0:1fqy.pdb in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Change the color for this representation by selecting ColorID → 1 red for Coloring Method.

8 In the Graphical Representations window, select 1:1rc2.pdb in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Select ColorID → 4 yellow for Coloring Method.

9 In the Graphical Representations window, select 2:1lda.pdb in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Select ColorID → 11 purple for Coloring Method.

10 In the Graphical Representations window, select 3:1j4n.pdb in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Select ColorID → 12 lime for Coloring Method. Close the Graphical Representations window.

Now your OpenGL window should look similar to Fig. 34, and you can see that the alignment was pretty good as the four aquaporin structures are very similar. You can also get more information about the alignment in the MultiSeq window by highlighting the molecules you wish to compare.

11 In the MultiSeq window, highlight `1fqy` by clicking on it. To highlight another molecule without unhighlighting `1fqy`, you need to Ctrl-click (or command-click on Mac) on that molecule. Highlight `1rc2` by clicking on it while holding down the Ctrl key on the keyboard (or the command key on Mac). When both `1fqy` and `1rc2` are highlighted, you should see on the lower left corner in the MultiSeq window a line of text: `QH:0.6442, RMSD:2.3043, Percent Identity:30.28` as shown in Fig. 35. Note, the values you obtain might be a little different depending on if your MultiSeq database is updated, but they should be close to the ones in Fig. 35.

The Q_H value is a metric for structural homology. It's an adaptation of the Q value that measures structural conservation[5]. Q=1 implies that structures are identical. When Q has a low score (0.1-0.3), structures are not aligned well, i.e., only a small fraction of the C_α atoms superimpose. Along with RMSD and Percent Identity, these numbers tell you that the `1fqy` and `1rc2` structures are pretty well aligned. You can repeat the previous step to compare the alignment of other molecules. To unselect a highlighted molecule, Ctrl-click on it again (or command-click on Mac).

[5] Eastwood, M.P., C. Hardin, Z. Luthey-Schulten, and P.G. Wolynes. "Evaluating the protein structure-prediction schemes using energy landscape theory." IBM J. Res. Dev. 45: 475-497, 2001. URL: http://www.research.ibm.com/journal/rd/453/eastwood.pdf

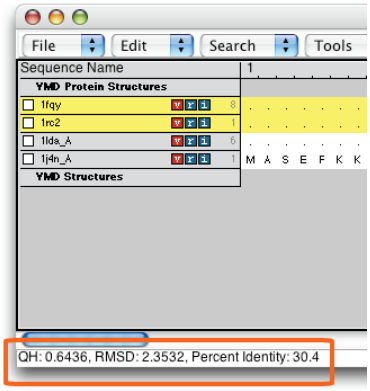


Figure 35: The Q_H, `RMSD`, and `Percent Identity` values can be used to determine how good an alignment is between two molecules, and how similar they are in structure and sequence.

6.1.3 Coloring molecules by their structural identity

You can also color the molecules according to the value of Q per residue (Q_{res}) obtained in the alignment. Q_{res} is the contribution from each residue to the overall Q value of aligned structures.

12 In the MultiSeq window, choose View → Coloring → Qres.

13 Look at the OpenGL window to see the impact this selection has made on the coloring of the aligned molecules (Fig. 36). The blue areas indicate that the molecules are structurally conserved at those points. If there is no correspondence in structural proximities at these points, the points appear red. As you can see the α-helices that form the pore are well conserved structurally among the four aquaporins, while there are more structural difference in the less functionally relevant loops.

6.2 Sequence Alignment with MultiSeq

Besides studying structural similarities, MultiSeq also allows protein comparison based on their sequence information. Sequence alignment is often used to identify conserved residues among similar proteins, as such residues are likely functionally important.

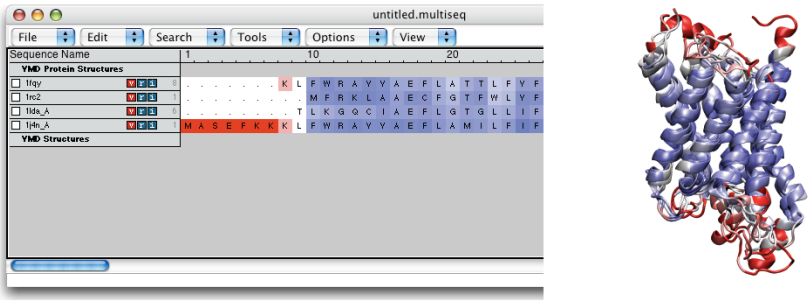


Figure 36: Result of a structural alignment of the four aquaporins, colored by Q_{res}.

6.2.1 Aligning molecules and coloring molecules by degree of conservation

1 In the MultiSeq window, select Tools → Sequence Alignment window. In the Sequence Alignment Options window, choose ClustalW under Alignment Program and make sure the Align All Sequences option is checked, and go to the bottom of the window and select OK. Now the four aquaporins have been aligned according to their sequence using the ClustalW tool (Thompson et al., *Nucl. Acids Res.*, **22**:4673, 1994).

2 Let's color the aligned molecules by their sequence similarity. In the Multiseq window, choose View → Coloring → Sequence identity. Now each amino acid is colored according to the degree of conservation within the alignment: blue means highly conserved, whereas red means very low or no conservation. Your MultiSeq window and OpenGL window should now resemble Fig. 37.

You have now aligned the four aquaporins by their sequence and identified the conserved residues, which tend to locate inside the pore (Fig. 38). Since aquaporin facilitates water transport across the membrane, these conserved residues are most likely the ones that carry out this important function.

6.2.2 Importing FASTA files for sequence alignment

Many times the structure of a protein might not be available, but its sequence is. You can analyze a protein in MultiSeq without its structure by loading its sequence information in the FASTA file format. If you don't have the FASTA file of a protein but you have its sequence, you can create a FASTA file easily with any text editor of your choice.

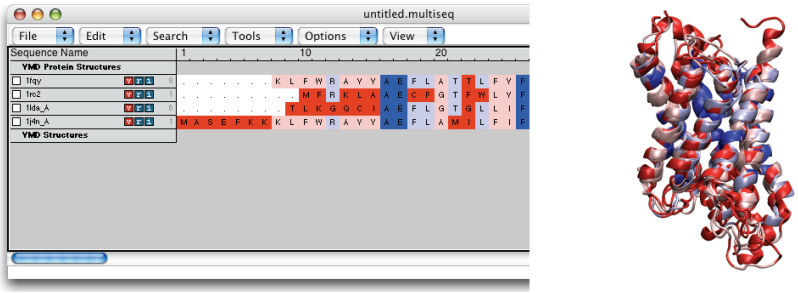


Figure 37: Result of a sequence alignment of the four aquaporins, colored by sequence identity.

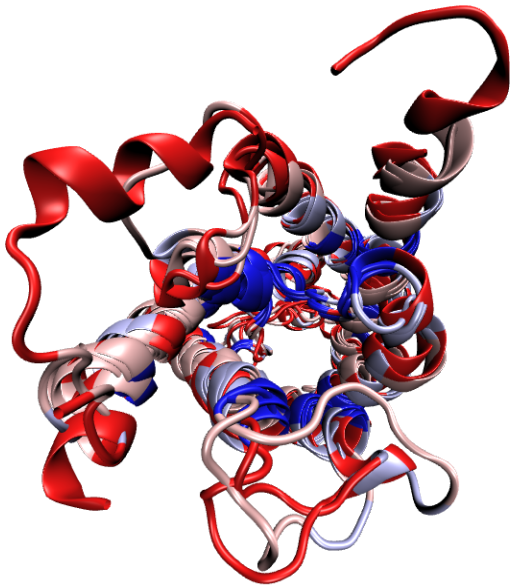


Figure 38: Top view of the aligned aquaporins colored by sequence conservation. The conserved residues locate mostly inside the aquaporin pore.

3 In the `vmd-tutorial-files` directory, find the provided FASTA sequence file `spinach_aqp.fasta` and open it with a text editor (Fig. 39a). A FASTA file contains a header that starts with ">", followed by the name of the protein. In the next line is the protein sequence in one-letter amino acid code. You can create FASTA files similarly in this format. When you create a FASTA file, remember to save it in plain text, and use `.fasta`

as the file extension. Close the text editor when you finish examining `spinach_aqp.fasta`.

Figure 39: The content of `spinach_aqp.fasta`.

4 In the MultiSeq window, select File → Import Data.... Select From File in the Import Data window, and press the top Browse button to select the file `spinach_aqp.fasta`. Press OK on the bottom of the Import Data window.

You have now loaded the sequence information of a spinach aquaporin into MultiSeq (Fig. 39b). You can now preform sequence alignment on the spinach aquaporin protein with other loaded aquaporin molecules. Let's try a sequence alignment between spinach and human aquaporins.

5 Click on the checkbox on the left of `spinach_aqp`, and click on the checkbox on the left of `1fqy.pdb`. Open the Sequence Alignment Options window by selecting Tools → Sequence Alignment. Choose ClustalW as the Alignment Program and under the Multiple Alignment options on the top, check Align Marked Sequences. Go to the bottom of the window and select OK.

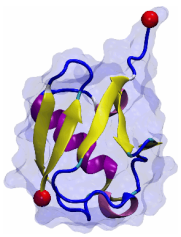
The sequence of spinach aquaporin is now aligned with the sequence of human aquaporin, and you can check how good the alignment is by obtaining its Q_H and Sequence Identity values. If you feel that the two molecules are listed too far apart in the MultiSeq window, you can move the molecules by dragging them with your mouse. Also, as you might have noticed, in MultiSeq molecules can be "Marked" by checking their checkboxes. They can also be "Selected" by highlighting them. You can align only the molecules of your choice by selecting Align Marked Sequences or Align Selected Sequences, depending if you have marked or highlighted your molecules. This option is available for both structural alignment and sequence alignment.
The structures of spinach aquaporin are actually available (Törnroth-Horsefield et al., *Nature*, **439**:688, 2006), but now that you have learned how to import

FASTA sequence data, you can compare the sequences of proteins even if their structures are not yet published.

6 When you finish comparing the sequence of spinach aquaporin with other aquaporins, delete it by clicking on `spinach_aqp` and press delete or Backspace on your keyboard.

6.3 Phylogenetic Tree

The Phylogenetic Tree feature in MultiSeq elucidates the structure-based and/or sequence-based relationships between different proteins. Structure-based phylogenetic trees can be constructed according to the RMSD or Q values between the molecules after alignment; sequence-based phylogenetic trees can be constructed according to the percent identity or ClustalW values.



The Phylogenetic Tree. A phylogenetic tree is a dendogram, representing the succession of biological form by similarity-based clustering. Classical taxonomists use these methods to infer evolutionary relationships of multicellular organisms based on morphology. Molecular evolutionary studies use DNA, RNA, protein sequences or protein structures to depict the evolutionary relationships of genes and gene products. In this tutorial we employ Q_H and RMSD to depict evolution of protein structure. For a comprehensive explanation of phylogenetic trees, see *Inferring Phylogenies* by Joseph Felsenstein[6].

1 Align the structures again, by going to the Multiseq window and selecting Tools → Stamp Structural Alignment.

2 In the Stamp Structural Alignment window, select All Structures, and keep the default values for the rest of the parameters. Press the OK button to align the structures.

3 In the Multiseq program window choose Tools → Phylogenetic Tree. The Phylogenetic tree window will open.

4 Select Structural tree using Q_H, and press the OK button. A phylogenetic tree based on the Q_H values will be calculated and drawn as shown in Fig. 40. Here you can see the relationship between the four aquaporin, e.g., how the *E.Coli* AqpZ (1r2c) is related to human AQP1 (1fqy).

5 You can also construct the phylogenetic tree of the four aquaporins based on their sequence information. Close the Tree Viewer window

[6] J. Felsenstein *Inferring Phylogenies*. Sinauer Associates, Inc.: 2004.

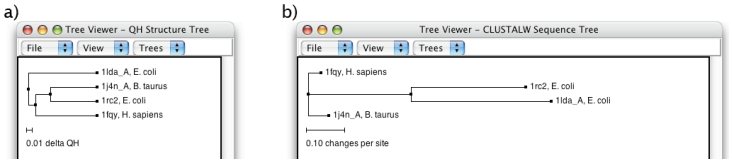


Figure 40: a) A structure-based phylogenetic tree generated by Q_H values. b) A sequence-based phylogenetic tree generated by ClustalW.

6 You need to perform the sequence alignment again for the four aquaporin proteins. In your MultiSeq window, choose Tools → Sequence Alignment, check ClustalW under Alignment Program and make sure the Align All Sequences option is checked, and press OK.

7 In the Multiseq program window choose Tools → Phylogenetic Tree to open Phylogenetic tree window again.

8 Select Sequence tree using ClustalW, and press the OK button. A phylogenetic tree based on ClustalW will be calculated and drawn as shown in Fig. 40.

9 Quit VMD.

7 Running VMD on Supercomputers

As the number and speed of state-of-the-art CPUs and GPUs in modern HPC systems have increased the size and time scale of molecular dynamics simulations, the size of the trajectory data to be analyzed has increased commensurately. This has led molecular scientists to perform an increasing fraction of their routine VMD analysis tasks on the same HPC systems where simulations are performed, where high performance processors, GPUs, and storage systems make fast work of large trajectory analysis and visualization tasks. To facilitate usage on large scale HPC systems, VMD can be run in a non-interactive mode of execution, either serially or in parallel using MPI, with and without OpenGL graphics, all while retaining support for high fidelity ray tracing and support for GPU acceleration of computationally demanding analysis tasks.

7.1 Running VMD in Text Mode

All graphics-capable VMD builds can also be run in text mode any time there is a need to perform long-running non-interactive analysis tasks. When run in text mode, VMD does not allow use of the "snapshot" renderer or other OpenGL rendering features, however non-OpenGL renderers such as Tachyon, TachyonLOptiX, and TachyonLOSPRay are still available since they don't depend on OpenGL. When run in text mode, VMD prints no status messages related to interactive graphics during startup, but availability of optional features such as GPU-acceleration are still reported. To launch VMD in text mode, the `-dispdev text` flag is appended to the command line flags, as shown below:

```
% vmd -dispdev text -e myscript.vmd

Info) VMD for LINUXAMD64, version 1.9.3 (November 30, 2016)
Info) http://www.ks.uiuc.edu/Research/vmd/
Info) Email questions and bug reports to vmd@ks.uiuc.edu
Info) Please include this reference in published work using VMD:
Info)    Humphrey, W., Dalke, A. and Schulten, K., 'VMD - Visual
Info)    Molecular Dynamics', J. Molec. Graphics 1996, 14.1, 33-38.
Info) ------------------------------------------------------------
Info) Multithreading available, 64 CPUs detected.
Info)   CPU features: SSE2 AVX FMA
Info) Free system memory: 61GB (96%)
Info) No CUDA accelerator devices available.
Info) Dynamically loaded 2 plugins in directory:
Info) /Projects/vmd/pub/linux64/lib/vmd193/plugins/LINUXAMD64/molfile
vmd >
```

7.2 Running VMD with Off-Screen OpenGL Graphics

VMD can be compiled with support for off-screen OpenGL rendering, enabling the "snapshot" renderer to be used in non-interactive VMD sessions, e.g., for in-situ rendering. VMD supports off-screen rendering via so-called OpenGL pixel buffers or "Pbuffers", using either the OpenGL GLX windowing system interface, or through the OpenGL EGL embedded graphics interface.

The GLX-based Pbuffer feature is normally available in standard graphically-enabled compilations of VMD for Linux/Unix. VMD GLX off-screen rendering requires that an X-Windows server be running and that the DISPLAY environment variable is set to the correct server hostname and display. The GLX pbuffer feature is most appropriate when running VMD on a user's own desktop workstation, since the same user would likely control the active windowing system.

To use VMD with off-screen graphics, the `-dispdev openglpbuffer` flag is added to the VMD launch command, as shown below:

```
% vmd -dispdev openglpbuffer -e myscript.vmd

Info) VMD for LINUXAMD64, version 1.9.3 (November 30, 2016)
Info) http://www.ks.uiuc.edu/Research/vmd/
Info) Email questions and bug reports to vmd@ks.uiuc.edu
Info) Please include this reference in published work using VMD:
Info)    Humphrey, W., Dalke, A. and Schulten, K., 'VMD - Visual
Info)    Molecular Dynamics', J. Molec. Graphics 1996, 14.1, 33-38.
Info) ------------------------------------------------------------
Info) Multithreading available, 40 CPUs detected.
Info)   CPU features: SSE2 AVX AVX2 FMA
Info) Free system memory: 411GB (81%)
Info) Creating CUDA device pool and initializing hardware...
Info) Detected 3 available CUDA accelerators:
Info) [0] Quadro M6000 24GB  24 SM_5.2 @ 1.11 GHz, 24GB RAM, AE2, ZCP
Info) [1] Quadro M6000 24GB  24 SM_5.2 @ 1.11 GHz, 24GB RAM, AE2, ZCP
Info) [2] Quadro M6000 24GB  24 SM_5.2 @ 1.11 GHz, 24GB RAM, AE2, ZCP
Info) OpenGL Pbuffer size: 4096x2400
Info) OpenGL renderer: Quadro M6000 24GB/PCIe/SSE2
Info)   Features: MSAA(4) MDE CVA MTX NPOT PP PS GLSL(OVFGS)
Info)   Full GLSL rendering mode is available.
Info)   Textures: 2-D (16384x16384), 3-D (4096x4096x4096), Multitexture (4)
Info) Created GLX OpenGL Pbuffer for off-screen rendering
Info) Detected 3 available TachyonL/OptiX ray tracing accelerators
Info)   Compiling 1 OptiX shaders on 3 target GPUs...
```

In cases where it is inconvenient or not possible to launch a windowing system such as on large-scale HPC systems, VMD supports the use of EGL-based Pbuffer rendering through compilation with a special OpenGL runtime dispatch library. As a result of the current need for special compilation, EGL-enabled versions of VMD are made available separaely from conventional VMD builds, and in many cases the user may need to compile VMD from source since EGL is often used in conjunction with MPI-enabled builds of VMD for parallel rendering, as outlined below.

```
% vmd -dispdev openglpbuffer -e myscript.vmd

Info) VMD for LINUXAMD64, version 1.9.3 (December 1, 2016)
Info) http://www.ks.uiuc.edu/Research/vmd/
Info) Email questions and bug reports to vmd@ks.uiuc.edu
Info) Please include this reference in published work using VMD:
Info)    Humphrey, W., Dalke, A. and Schulten, K., 'VMD - Visual
Info)    Molecular Dynamics', J. Molec. Graphics 1996, 14.1, 33-38.
Info) ------------------------------------------------------------
Info) Multithreading available, 40 CPUs detected.
Info)   CPU features: SSE2 AVX AVX2 FMA
Info) Free system memory: 411GB (81%)
Info) Creating CUDA device pool and initializing hardware...
Info) Detected 3 available CUDA accelerators:
Info) [0] Quadro M6000 24GB  24 SM_5.2 @ 1.11 GHz, 24GB RAM, AE2, ZCP
Info) [1] Quadro M6000 24GB  24 SM_5.2 @ 1.11 GHz, 24GB RAM, AE2, ZCP
Info) [2] Quadro M6000 24GB  24 SM_5.2 @ 1.11 GHz, 24GB RAM, AE2, ZCP
Info) EGL: node[0] bound to display[0], 3 displays total
Info) EGL version 1.4
Info) OpenGL Pbuffer size: 4096x2400
Info) OpenGL renderer: Quadro M6000 24GB/PCIe/SSE2
Info)   Features: STENCIL MSAA(4) MDE CVA MTX NPOT PP PS GLSL(OVFGS)
Info)   Full GLSL rendering mode is available.
Info)   Textures: 2-D (16384x16384), 3-D (4096x4096x4096), Multitexture (4)
Info) Created EGL OpenGL Pbuffer for off-screen rendering
Info) Detected 3 available TachyonL/OptiX ray tracing accelerators
Info)   Compiling 256 OptiX shaders on 3 target GPUs...
```

Both GLX- and EGL-based off-screen OpenGL Pbuffer rendering support all of the advanced OpenGL features used by VMD, such as programmable shading, multisample antialiasing, and 3-D texture mapping. One area where they behave differently from traditional windowed-OpenGL is that they have a fixed maximum framebuffer resolution, which defaults to 4096×2400. The maximum framebuffer size can be increased beyond this resolution by setting the

`VMDSCRSIZE` environment variable to the maximum framebuffer resolution that might be required during a VMD run. In Bourne/bash shells, this would be done with the command: `export VMDSCRSIZE="8192 4096"` In C-shell/tcsh shells, this would be done with the command: `setenv VMDSCRSIZE "8192 4096"`

7.3 Using VMD with MPI

When VMD has been compiled with support for MPI it can be run in parallel on thousands of compute nodes at a time. There are presently a few noteworthy considerations that affect how VMD is compiled, launched, and how it behaves in a parallel environment as compared with conventional interactive desktop usage.

When compiled with MPI support and launched with the platform-dependent `mpirun` or site-specific launch commands (e.g., `aprun`, `jsrun`, or similar), VMD will automatically initialize MPI internally, and each parallel VMD instance will be assigned a unique MPI rank. During startup, a parallel launch of VMD will print hardware information about each of the participating compute nodes from node 0. When a parallel VMD run exits, all nodes are expected to call `exit` at the same time so that they shutdown MPI together.

Due to the fact that MPI does not (yet) include a standarized binary interface, MPI support requires that VMD be compiled from source code on the target platform for each MPI implementation to be supported. For example, VMD would have to be compiled separately for each MPI to be supported on the system, e.g., MPICH, OpenMPI, and/or other MPI versions. This means that in the general case, unlike the approach taken by the VMD development team where binary VMD distributions are provided for all mainstream computer and operating system platforms, this is not possible in the context of MPI. Users wishing to use VMD with MPI must compile VMD from source code.

Some MPI implementations require special interactions with batch queueing systems or storage systems, and in such cases it is necessary to modify the standard VMD launcher scripts to perform any extra steps or to invoke any platform-specific parallel launch commands. By modifying the VMD launch script, users can continue to use familiar VMD launch syntax while gaining the benefits of parallel analysis with MPI. The VMD launch script has been modified so that it can automatically recognize cases where VMD has been launched within batch schedulers used on Cray XK and XC supercomputers such as NCSA Blue Waters, ORNL Titan, CSCS Piz Daint, and related systems, where the VMD executable must be launched using the 'aprun' or 'srun' utilities, depending on the scheduling system in use.

MPI-enabled builds of VMD can often also be run on login nodes or interactive visualization nodes that may not managed by the batch scheduler and/or may not support MPI, so long as MPI and other shared libraries used on the

compute nodes are also available on the login or interactive visualization nodes. To run an MPI-enabled VMD build outside of MPI, i.e., without 'mpirun', the environment variable VMDNOMPI can be set, which will prevent VMD from calling any MPI APIs during the run, allowing it to behave like a normal non-MPI build for convenience. In most cases, this makes it possible to use a single VMD build on all of the different compute node types on a system.

At present, the interactive console handling used in interactive text interpreters conflicts with the behavior of some mainstream MPI implementations, so when VMD is run in parallel using MPI, the interactive console is disabled and VMD instead reads commands only from script files specified with the "-e" command line argument.

Aside from the special launch behavior and lack of the interactive text console, MPI runs of VMD support high performance graphics with full support for OpenGL via GLX or EGL, and ray tracing with Tachyon, OptiX, and OSPRay.

Here is an example session showing a VMD run performed on the CSCS Piz Daint Cray XC50 supercomputer with NVIDIA Tesla P100 GPU accelerators:

```
stonej@daint103> srun -C gpu -n 256 --ntasks-per-node=1 \
   /users/stonej/local/bin/vmd193 -dispdev text -e rendermovie.tcl
on daint103
srun: job 50274 queued and waiting for resources
srun: job 50274 has been allocated resources
Info) VMD for CRAY_XC, version 1.9.3 (December 15, 2016)
Info) http://www.ks.uiuc.edu/Research/vmd/
Info) Email questions and bug reports to vmd@ks.uiuc.edu
Info) Please include this reference in published work using VMD:
Info)    Humphrey, W., Dalke, A. and Schulten, K., 'VMD - Visual
Info)    Molecular Dynamics', J. Molec. Graphics 1996, 14.1, 33-38.
Info) ------------------------------------------------------------
Info) Creating CUDA device pool and initializing hardware...
Info) Initializing parallel VMD instances via MPI...
Info) Found 256 VMD MPI nodes containing a total of 6144 CPUs and 256 GPUs:
Info)    0:  24 CPUs, 60.8GB (96%) free mem, 1 GPUs, Name: nid03072
Info)    1:  24 CPUs, 60.8GB (96%) free mem, 1 GPUs, Name: nid03073
Info)    2:  24 CPUs, 60.8GB (96%) free mem, 1 GPUs, Name: nid03074
[...example output omitted...]
Info)  253:  24 CPUs, 60.9GB (96%) free mem, 1 GPUs, Name: nid03375
Info)  254:  24 CPUs, 60.9GB (96%) free mem, 1 GPUs, Name: nid03376
Info)  255:  24 CPUs, 60.9GB (96%) free mem, 1 GPUs, Name: nid03377
```

7.4 VMD parallel commands

The parallel command enables large scale parallel scripting when VMD has been compiled with MPI support. In absence of MPI support, the parallel command is still available, but it operates the same way it would if an MPI-enabled VMD would when run on only a single node. The parallel command enables large analysis scripts to be easily adapted for execution on large clusters and supercomputers to support simulation, analysis, and visualization operations that would otherwise be too computationally demanding for conventional workstations.

- **nodename**: Return the hostname of the current compute node.
- **noderank**: Return the MPI rank of the current compute node.
- **nodecount**: Return the total number of MPI ranks in the currently running VMD job.
- **allgather** *object*: Perform a parallel allgather operation across all MPI ranks, taking the user defined object as input to each caller. All VMD MPI ranks must participate in the allgather operation.
- **allreduce** *user_reduction_procedure object*: Perform a parallel reduction across all MPI ranks by calling the user-supplied reduction procedure, passing in a user defined object. All VMD MPI ranks must participate in the allreduce operation.
- **barrier**: Perform a barrier synchronization across all MPI ranks.
- **for** *startcount endcount user_worker_procedure object*: Invoke VMD parallel work scheduler to run a computation over all MPI ranks. The VMD work scheduler uses dynamic load balancing to assign work indices to workers, calling the user-defined procedure for each work item.

The `parallel gather` command allows VMD analysis scripts to gather results from all of the nodes, returning the complete set of per-node results in a new Tcl list. Here is a simple example procedure that shows this principle by gathering up all of the MPI node hostnames and printing them. Note that to avoid redundant output, the script always does output only on node rank 0:

```
proc testgather { } {
  set noderank  [parallel noderank]

  # only print messages on node 0
  if {$noderank == 0} {
    puts "Testing parallel gather..."
```

```
  }

  # Do a parallel gather of all node names
  set datalist [parallel allgather [parallel nodename]]

  # only print messages on node 0
  if {$noderank == 0} {
    puts "datalist length: [llength $datalist]"
    puts "datalist: $datalist"
  }
}
```

The `parallel allreduce` command allows VMD to compute a parallel reduction across all MPI ranks, returning the final result to all nodes. Each rank contributes one input to the reduction. The user must provide a Tcl proc that performs the appropriate reduction operation for a pair of data items, resulting in a single item. This approach allows arbitrarily complex reductions on arbitrary data to be devised by the user. The VMD reduction implementation calls the user provided routine in parallel on pairs of arguments exchanged between ranks, with each such call producing a single reduced output value. VMD performs successive parallel reduction operations until it computes the final reduced value that is returned to all ranks.

The example below returns the sum of all of the MPI node ranks:

```
proc sumreduction { a b } {
  return [expr $a + $b]
}

proc testreduction {} {
  set noderank  [parallel noderank]

  # only print messages on node 0
  if {$noderank == 0} {
    puts "Testing parallel reductions..."
  }
  parallel allreduce sumreduction $noderank
}
```

VMD can easily perform parallel rendering of trajectories or other kinds of movies with relatively simple scripting based on the `parallel` commands above. Try running this simple script in an MPI-based VMD session, which just uses the individual MPI node ranks to render one frame per-node. Be sure to replace "somedir" with your own directory:

```
set noderank [parallel noderank]

puts "node $noderank is running ..."
parallel barrier

mol new /somedir/alanin.pdb waitfor all
puts "node $noderank has loaded data"
parallel barrier

rotate y by [expr $noderank * 20]
render TachyonInternal test_node_$noderank.tga
puts "node $noderank has rendered a frame"

parallel barrier
quit
```

A much more sophisticated (but incomplete) example below shows how the `parallel for` command can be used along with a user-defined procedure to do larger scale parallel rendering with dynamic load balancing, passing parameters into the user defined procedure, triggering the VMD movie maker plugin and any user-defined per-frame callback that may be active therein. The `userdata` parameter shown here is used to communicate information necessary for the user-defined worker procedure to interpret the meaning of incoming work indices and take appropriate actions. The `userdata` parameter enables the user-provided procedure to avoid using global variables or hard-coded implementation details.

```
proc render_one_frame { frameno userdata } {
  # retrieve user data rendering workers
  set formatstr [lindex $userdata 0]
  set dir [lindex $userdata 1]
  set renderer [lindex $userdata 2]
  # Set frame, triggering user-defined movie
  # callbacks to update the molecular scene
  # prior to rendering of the frame
  set ::MovieMaker::userframe $frameno
  # Regenerate molecular geometry if not up to date
  display update
  # generate output filename, and render the frame
  set fname [format $formatstr $frameno]
  render $renderer $dir$fname
}

proc render_movie { dir formatstr framecount renderer } {
```

```
  set userdata {}
  lappend userdata $formatstr
  lappend userdata $dir
  lappend userdata $renderer
  set lastframe [expr $framecount - 1]
  parallel for 0 $lastframe render_one_frame $userdata
}
```

Final Remarks

What we are able to present in this tutorial only showcases a small part of VMD's capability. But now that you have learned the basics of VMD, you are ready to explore its many other features most suitable for your research. For this purpose there are many tutorials available that aim to offer a more focused training, either on a specific tool or on a scientific topic. You can find many useful documentations, including the comprehensive VMD User's Guide, in the VMD homepage http://www.ks.uiuc.edu/Research/vmd/. If you have any question on using VMD, we encourage you to subscribe to the VMD mailing list http://www.ks.uiuc.edu/Research/vmd/mailing_list/.

Citing VMD

VMD developement is funded by the National Institutes of Health (NIH 9P41GM104601 - Resource for Macromolecular Modeling and Bioinformatics). Proper citation is a primary way that we demonstrate the value of VMD to the scientific community, and is essential for continued funding. We request that all published work utilizing VMD include the primary VMD citation at a minimum:

Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", J. Molec. Graphics, 1996, vol. 14, pp. 33-38.

Publication images rendered the built-in Tachyon renderers should also cite:

Stone, J. E., "An Efficient Library for Parallel Ray Tracing and Animation", Master's Thesis, University of Missouri-Rolla, 1998.

Stone, J. E., Vandivort, K. L., Schulten, K., "GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms", UltraVis'13: Proceedings of the 8th International Workshop on Ultrascale Visualization, pp. 6:1-6:8, 2013.

Work that uses softwares or plugins incorporated into VMD should also add the proper citations for those tools. For example, works that uses MultiSeq as introduced in Section 6 should cite:

Roberts, E., Eargle, J., Wright, D. and Luthey-Schulten Z., "MultiSeq: Unifying sequence and structure data for evolutionary analysis", BMC Bioinformatics, 2006, 7:382.

Please see http://www.ks.uiuc.edu/Research/vmd/allversions/cite.html for more information on how to cite VMD and its tools.