

# Using VMD with LAMMPS

**Dr. Axel Kohlmeyer**

LAMMPS Core Developer and Mailing List “Bad Cop”

Research Professor, Department of Mathematics

Associate Director, Institute for Computational Science

Assistant Vice President for High-Performance Computing

**Temple University**

Philadelphia PA, USA

**a.kohlmeyer@temple.edu**

# Visual Molecular Dynamics - VMD

- A tool for primarily designed for modeling, visualization and analysis of biological systems, but not limited to that
- Support for all major computing platforms
- Many flexible methods for representing and coloring of data
- Multi-core support and GPU acceleration for selected features
- Flexible and powerful scripting in Tcl (& Python)
- Many extensions and plugins, many user contributed scripts
- Export to raytracing software for publication level images
- Supports OpenGL graphics acceleration and 3D Stereo
- VMD home page: <http://www.ks.uiuc.edu/Research/vmd/>

# Graphical and Text Mode Interface

- VMD can be controlled via its GUI or scripts
- Most GUI dialogs emit text mode commands on entering changes (can be logged to a file)
- Thus the GUI and scripts can be used together
- The visualization status can be saved as a sequence of text commands to a file (Note: this does not include data)
- Tcl scripts can be added to GUI event hooks
- Custom Tk/Tcl plugins can be included



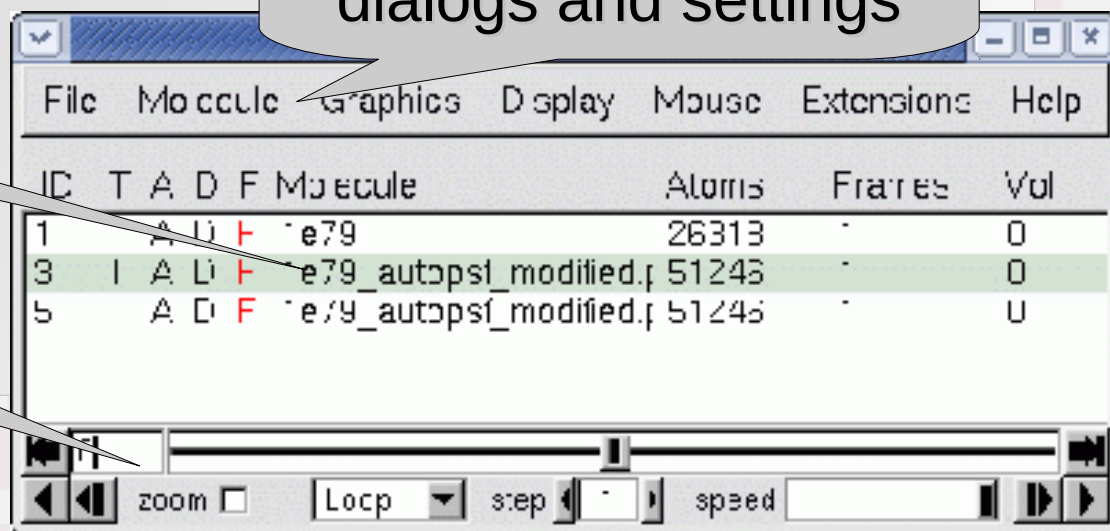
# VMD Main Dialog

- Central hub with access to all dialog windows
- Displays status of current VMD session
- Global “molecule” toggles (active/visible/frozen)
- Animation controls (“media player” like)

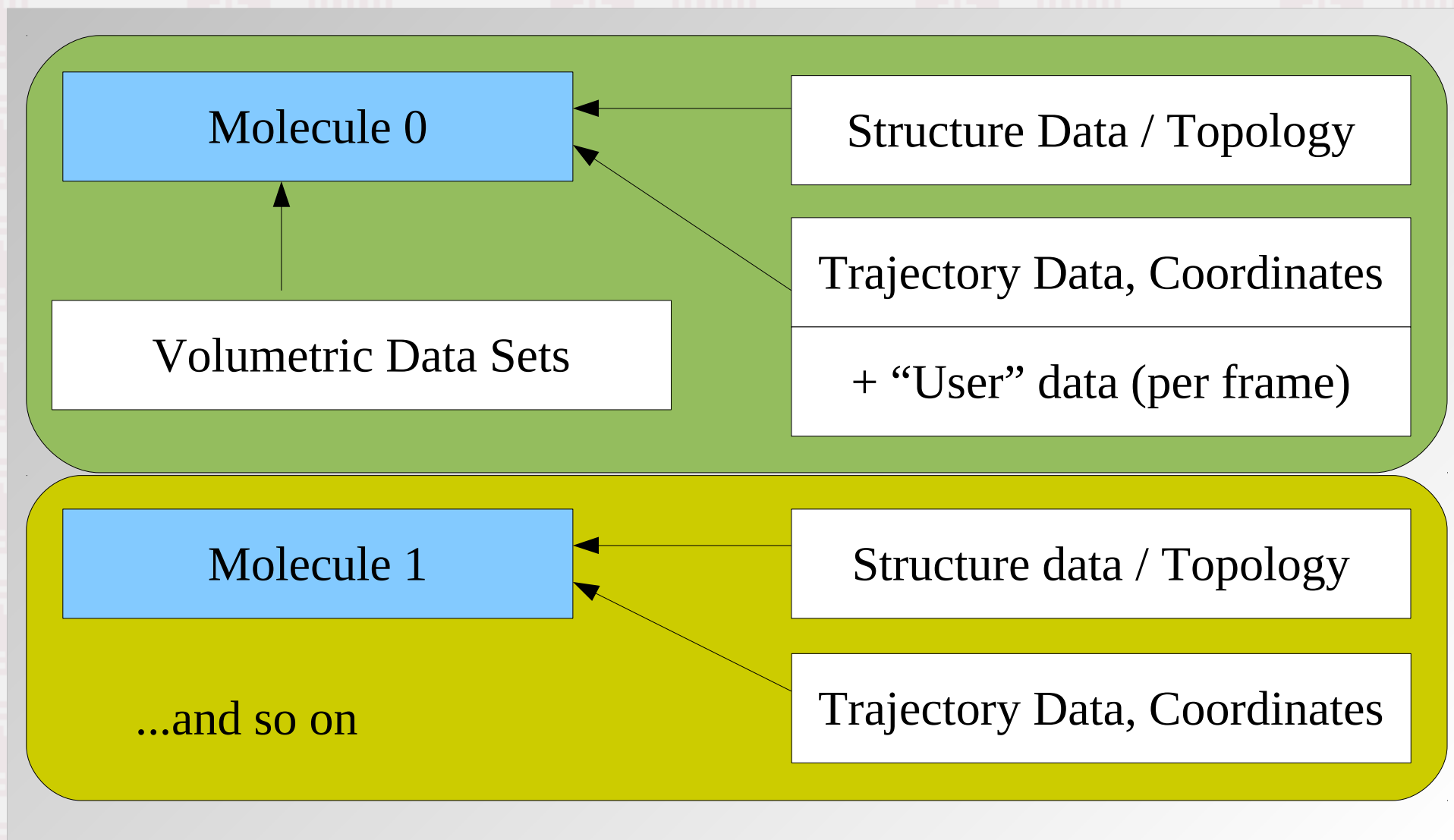
Molecule and  
data set status

Animation controls

Access to additional  
dialogs and settings

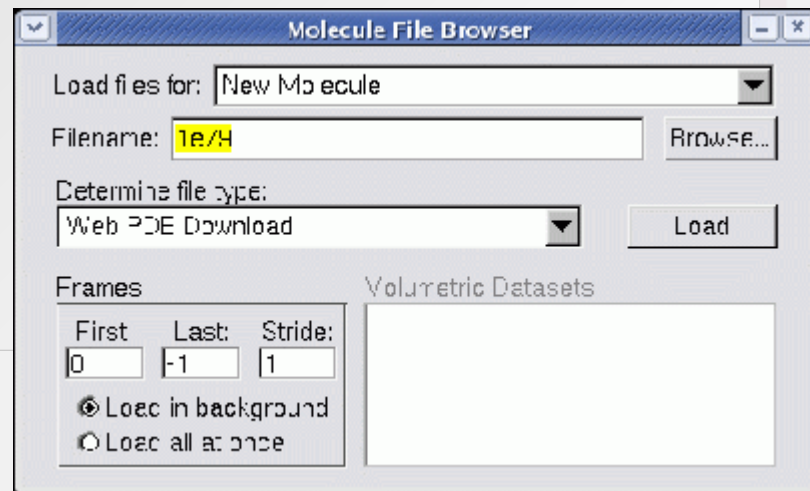


# The VMD Data Model



# Loading Data into VMD

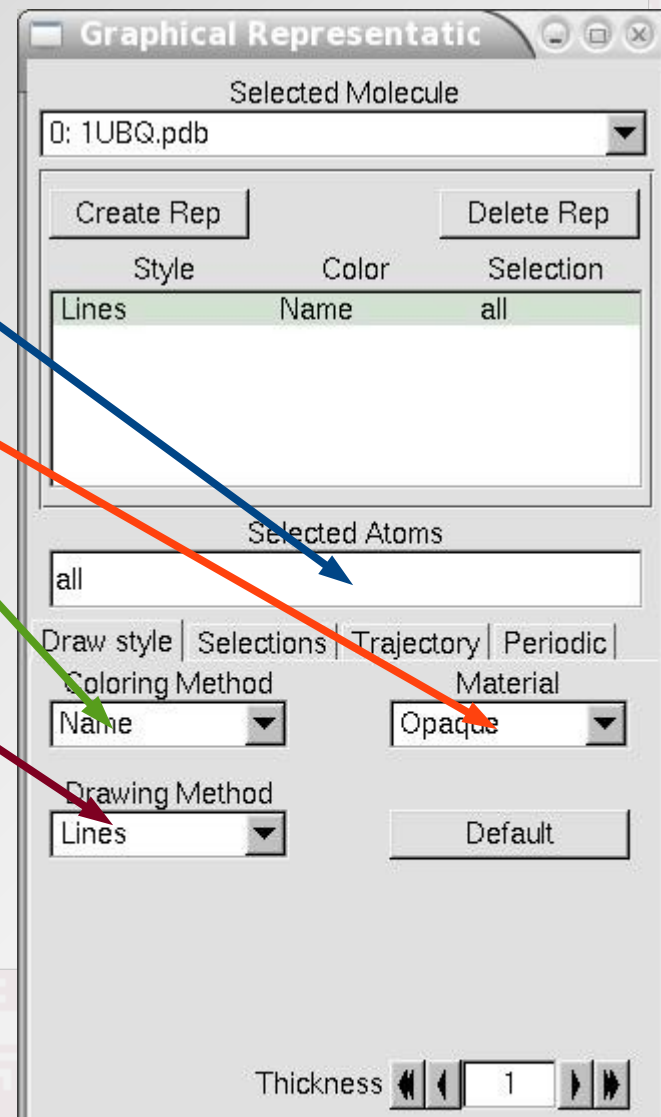
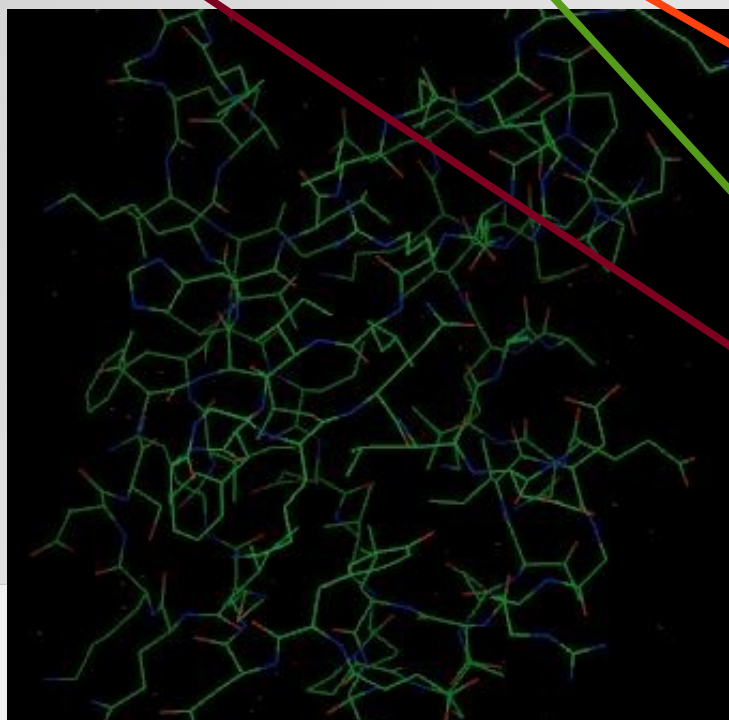
- VMD reads “structure” and “coordinate” data
- VMD tries to guess missing “structure” data; guess based on PDB naming conventions
- The “structure” data is stored only once
- Some files only contain “structure” data (PSF)
- Some files only “coordinate” data (DCD)
- Some both or parts of both
- Multiple files can be read into the same “molecule”





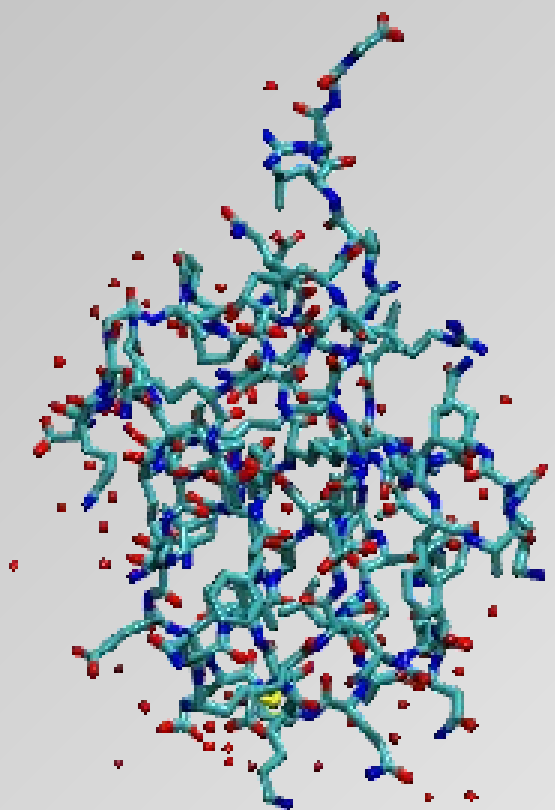
# Displaying a Molecule in VMD

- By default VMD will represent your molecule showing “all” atoms, colored by “Name”, as “Lines”, with an “Opaque” material

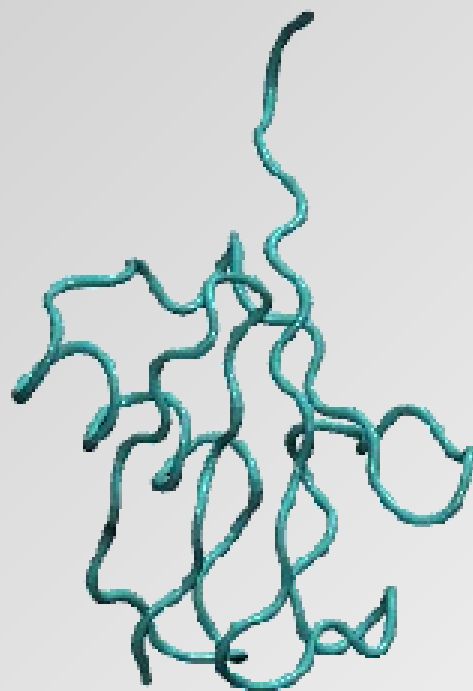




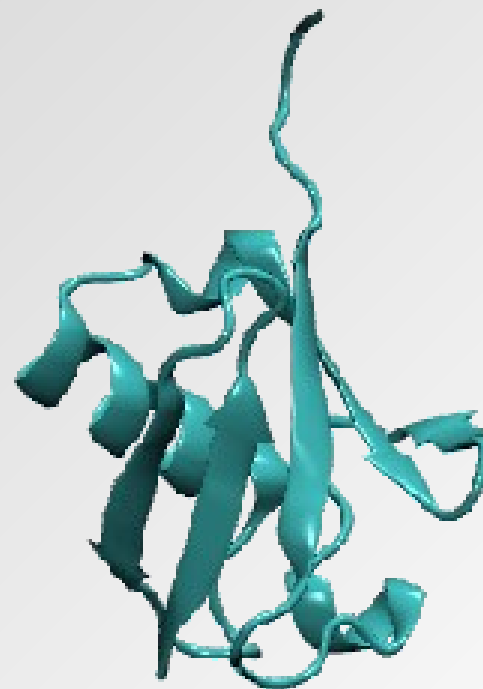
# Representation Options



Licorice

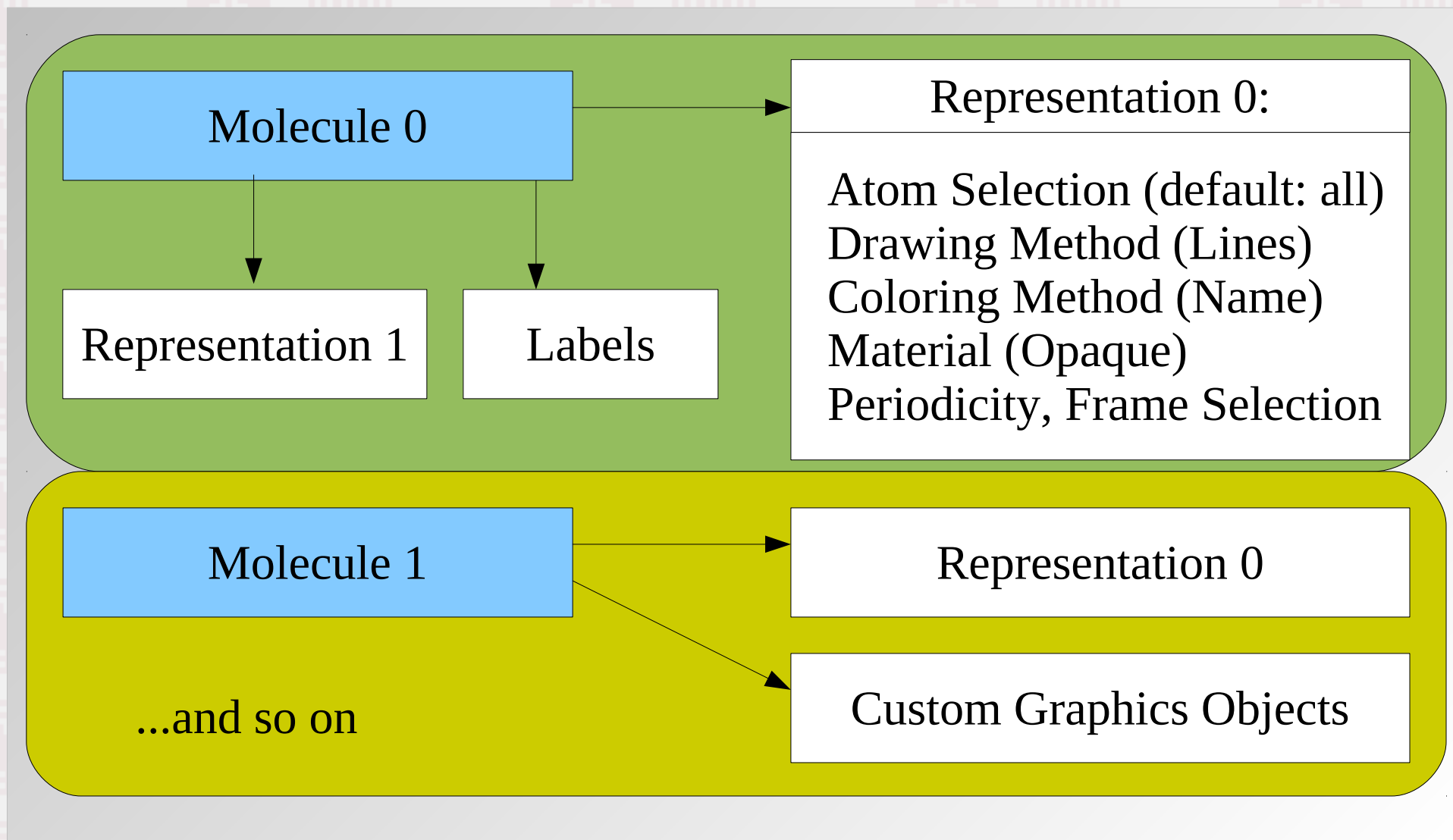


Tube



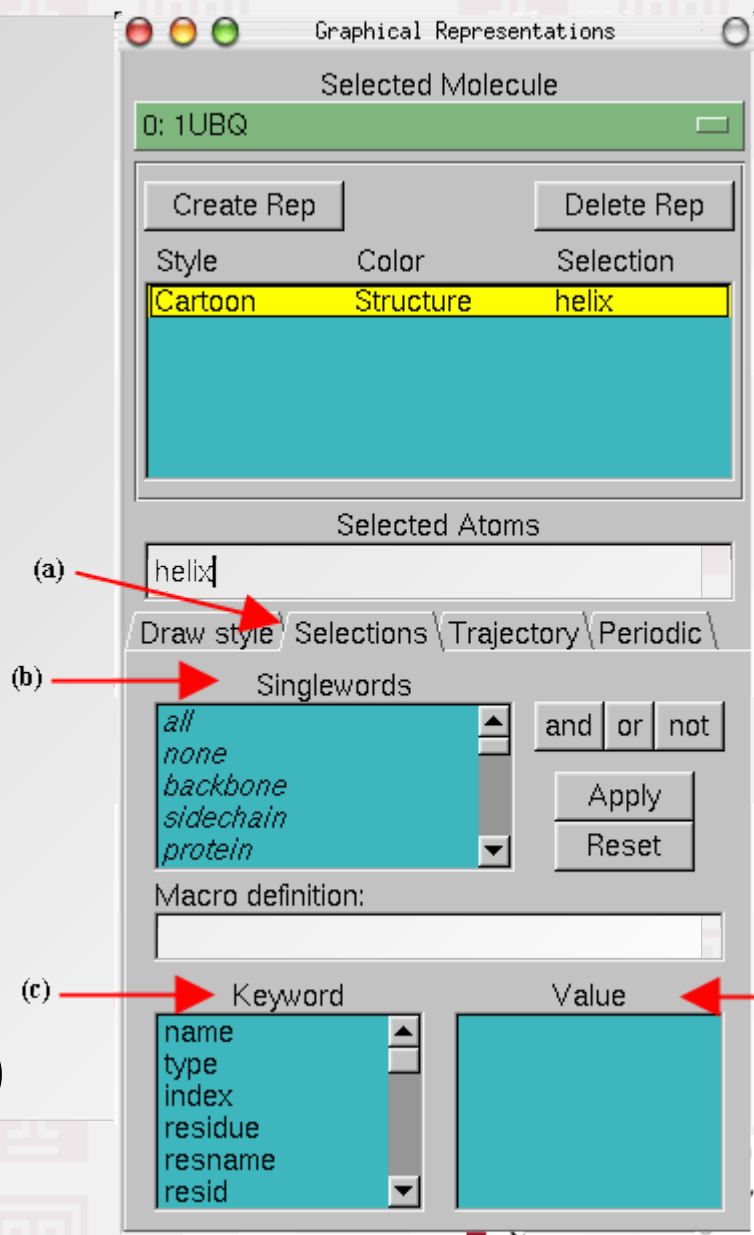
NewCartoon

# The VMD Visualization Model



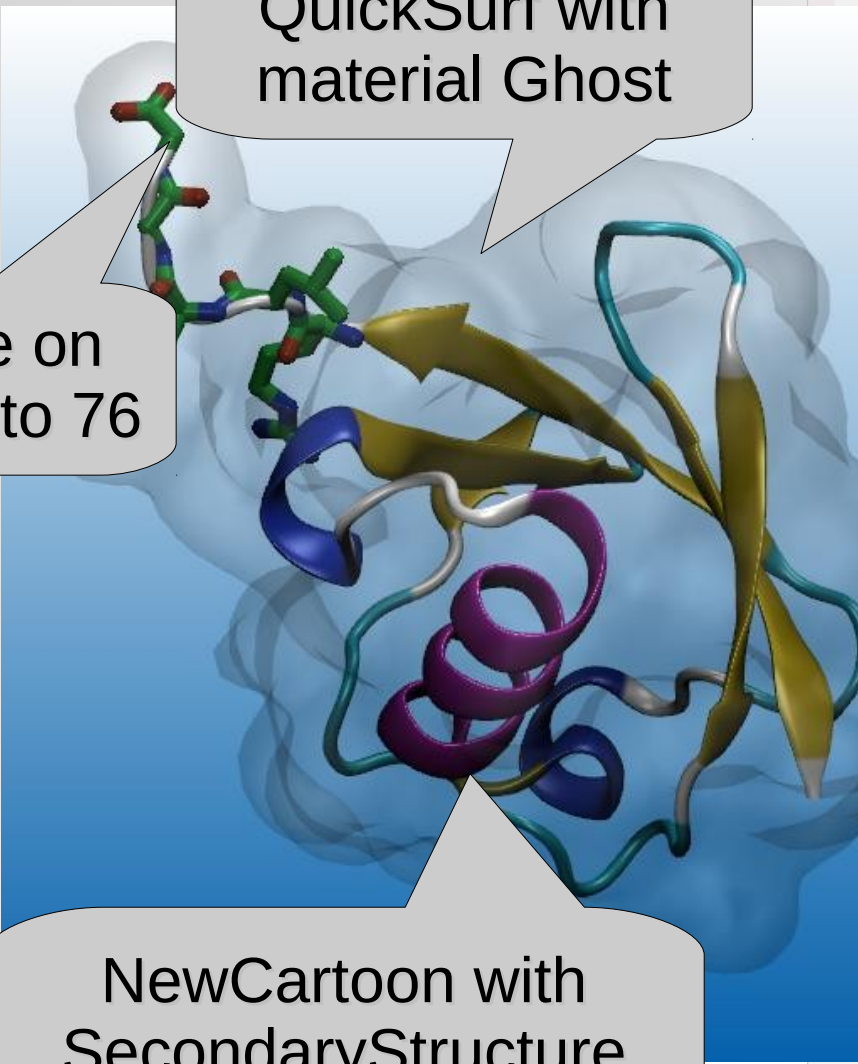
# Atom Selection Expressions

- Selects atoms by properties or comparisons
- “within # of” operations, macros, arithmetic expressions, comparisons, ranges
- Individual selection expressions can be combined with **logical** and/or/not
- Some Examples:
  - protein and not (resid 72 to 76)
  - same fragment as water within 4.0 of resname GLY
  - name CA and chain A and ( $x < 10.0$ )



# Complex Visualizations

- Through using multiple representations with different atom selections, drawing methods, coloring schemes, and materials you can build complex visualization that highlight important information and hide what is distracting



Licorice on resid 74 to 76

The image shows a 3D molecular model of a protein. The protein backbone is represented by a ribbon structure with various colors (yellow, blue, purple, green). A specific region of the protein is highlighted with a licorice representation (stick model) in green and red. The protein is shown within a semi-transparent blue surface representation. Three callout boxes point to different parts of the model: one to the licorice representation, one to the overall protein structure, and one to the ribbon representation.

QuickSurf with material Ghost

NewCartoon with SecondaryStructure coloring scheme

# Saving Your Work

- Via “File...” menu:
  - Visualization state:

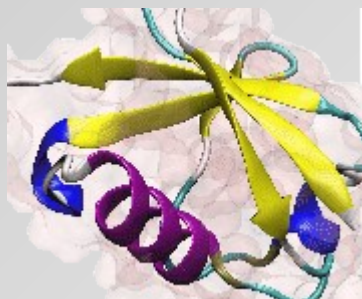
This saves all representations and settings and also the global visualization state (camera, zoom, projection, background, visibility, etc.). This does not include coordinates, only file names are stored.
  - Coordinate data:

This saves the data of the selected (top) molecule to a file in the chosen format
  - Rendered picture of OpenGL graphics window:

This allows to save the visualization as image, either as snapshot or via export to a raytracer

# Graphics Display Settings

- Perspective vs. Orthographic mode



- Anti-aliasing (smoother edges, if GPU support)
- Render mode (use GLSL, if GPU supports it, for better performance and better image quality)
- Depth cueing, Stereo mode, Axis position, ...

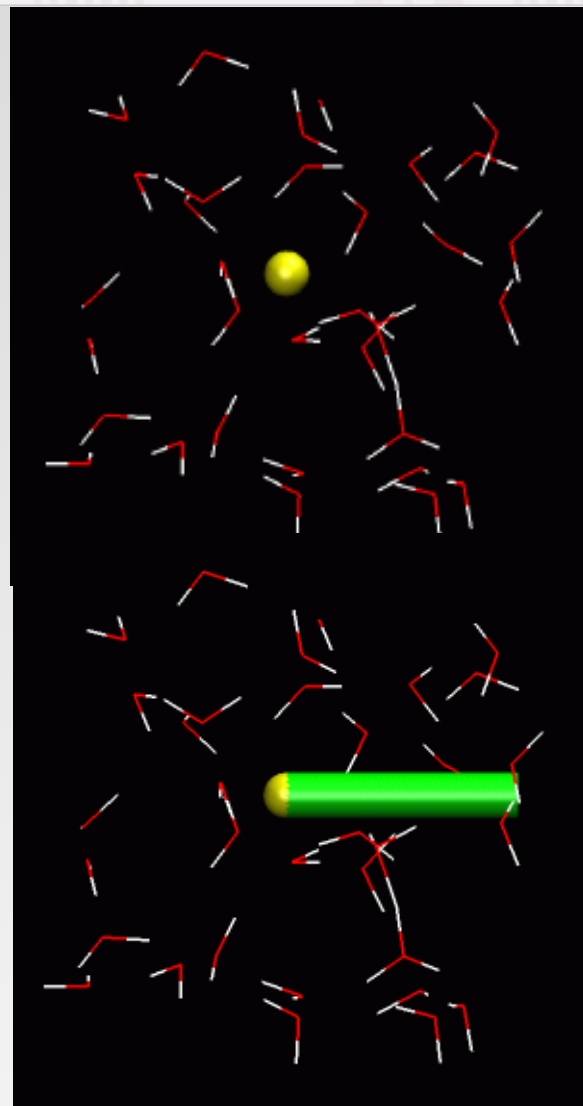


# Graphics <-> Mouse Interactions

- The OpenGL graphics window has multiple mouse modes:
  - Rotate mode (default, short cut: “r” key):
    - click and hold left button rotates scene “in plane”
    - click and hold right button rotates “orthogonal to plane”
    - scroll-wheel zooms “camera” in and out
  - Translate mode (short cut: “t” key):
    - click and hold left button translates “in plane”
    - click and hold right button translates “back and forth” (this is not the same as the camera zoom)
  - Query mode (short cut: “0” key):
    - click on atom displays (some) atom properties in text mode console window (terminal)
  - For more mouse modes (Label, Move) see “Mouse..” menu

# Adding Custom Graphics Objects

- Through the Tcl script interface, you can add custom graphics primitives (sphere, cylinder, cone, triangle) to a visualization (in molecule coordinate system)
- Example:  
draw color yellow  
draw material Diffuse  
draw sphere {0 0 0} radius 0.5  
draw color green  
draw cylinder {0 0 0} {5 0 0} \  
radius 0.5



# Changing Defaults with .vmdrc

Upon startup VMD looks for a file .vmdrc with TCL script code executed when VMD is launched

```
menu main on          ; # almost always needed
menu graphics on     ; # shows rep dialog
# plugins only available after full init
after idle { menu tkcon on } ; # TkConsole

axes location off
display resize 600 600

# default settings
mol default style VDW          ; # VDW not Lines
mol default selection protein ; # not "all"
```

# Changing Defaults with .vmdrc

More .vmdrc customizations:

```
display projection orthographic  
color Display Background silver
```

```
# new keyboard shortcuts
```

```
user add key o {display projection orthographic}
```

```
user add key p {display projection perspective}
```

```
user add key d {\
```

```
  if {[display get rendermode] != "Normal" } \  
    {display rendermode Normal } \  
    {display rendermode GLSL} }
```

```
# color assignments only possible after full init
```

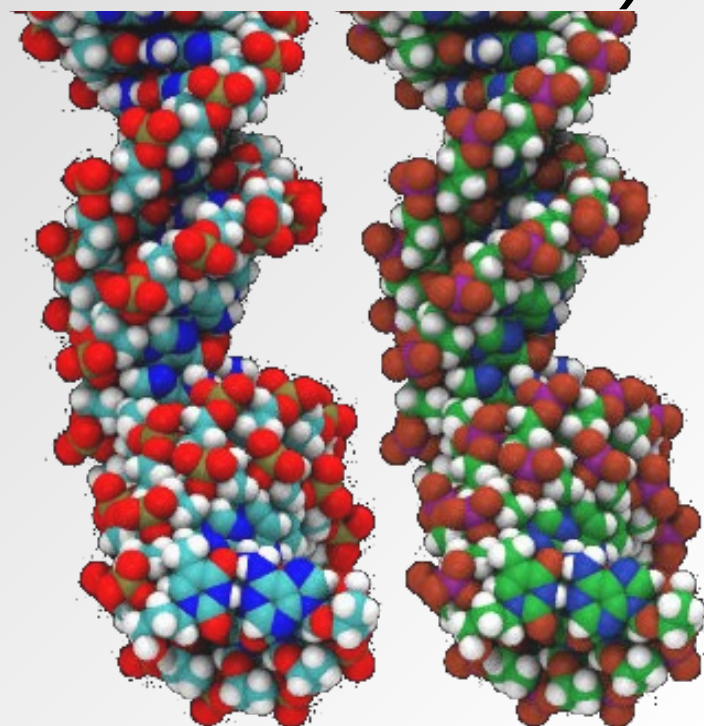
```
# "after idle" delays execution accordingly
```

```
after idle {color Name C green}
```

# Changing Color Palette

VMD by default uses "pure" colors defined in RGB color space, however those have often limited contrast when used with a different color space (pure green when printing, pure red with movies)

```
color change rgb 0 0.1 0.2 0.7
color change rgb 1 0.7 0.2 0.1
color change rgb 3 0.7 0.4 0.0
color change rgb 4 0.8 0.7 0.1
color change rgb 7 0.1 0.7 0.2
color change rgb 10 0.1 0.7 0.8
color change rgb 11 0.6 0.1 0.6
```

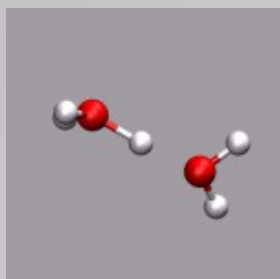


# Custom VMD Command Example

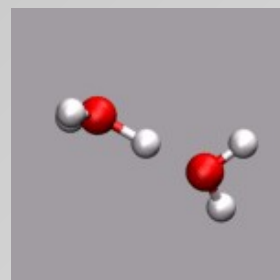
```
proc hardcopy { {renderer "TachyonInternal"}} {  
  # set background to white to save ink  
  set oldbg [colorinfo category Display Background]  
  color Display Background white  
  set tga {/tmp/.hardcopy.tga}  
  set ps {/tmp/.hardcopy.ps}  
  render $renderer $tga  
  exec convert $tga $ps  
  exec lpr $ps  
  exec rm -f $tga $ps  
  # restore saved background color  
  color Display Background $oldbg  
}  
  
user add key H hardcopy ; # link to key "shift-h"
```



# How to Show Bond-Breaking and Bond-Formation



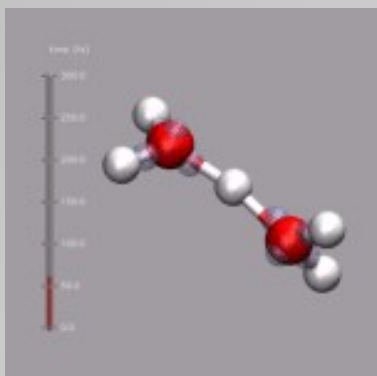
By default VMD assumes that bonds stay the same throughout a trajectory -> heritage in classical MD



Combining a VDW representation (at scale 0.2) with a Dynamic Bonds representation allows to recompute bonds on the fly.

With multiple average bond lengths and atom types, use multiple representations with selections

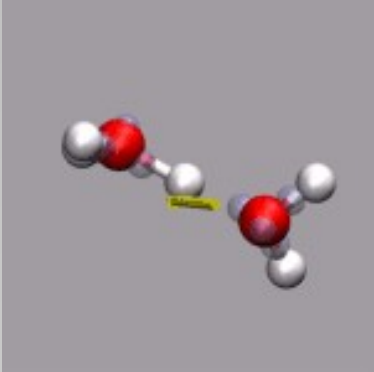
# Adding Dynamical Graphics



TCL allows to monitor objects (e.g. variables) with the **trace** command  
VMD provides several variables that one can connect custom functions to:

```
# use function for custom action commands
proc do_time {args} {
    display_time_bar 1.0 50.0 "fs" 0
}
# connect function to change of frame
set mol [molinfo top]
trace variable vmd_frame($mol) w do_time
```

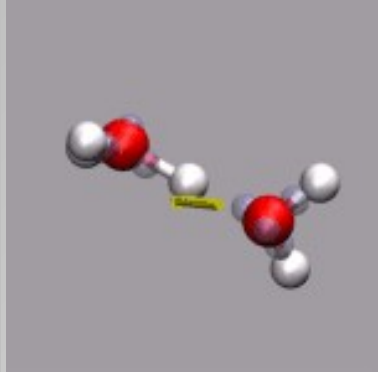
# Using External Data for Graphics



Read dipole vector into array and compute & store center of mass:

```
set dip [open "zundel.dip" r]
set sel [atomselect 0 "not name X"]
set nf [molinfo 0 get numframes]
for {set i 0} {$i < $nf} {incr i} {
  $sel frame $i
  set dipdata($i) \
    [list [measure center $sel] [gets $dip]]
}
close $dip
```

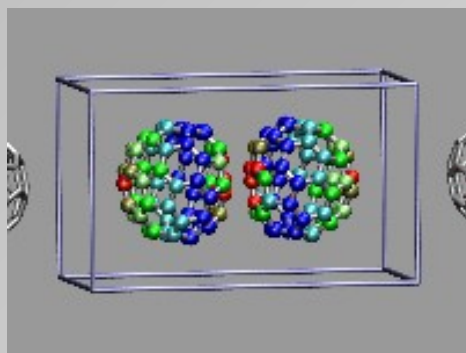
# Using External Data for Graphics



Hook up function to draw dipole vector graphics to **vmd\_frame**:

```
proc do_dipdraw {args} {
  global dipdata dipgraph
  set m [molinfo top]; set f [molinfo $m get frame]
  if {[info exists dipdata($f)]} then {
    if {[info exists dipgraph]} then {
      foreach g $dipgraph {graphics $m delete $g}
    }
    graphics $m color yellow
    lassign $dipdata($f) cnt vec
    set dipgraph [vmd_draw_vector $m \
      $cnt $vec 100000.0 10 0.08]
  }
}
trace variable vmd_frame(0) w do_dipdraw
```

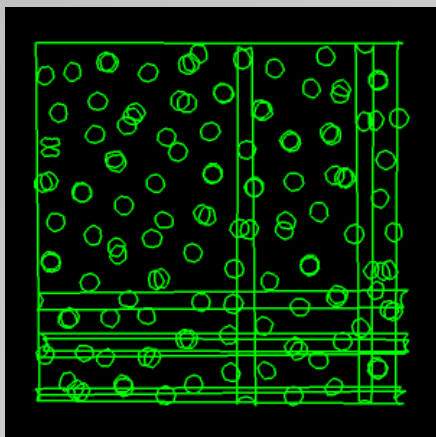
# Using a Computed User Value



Colorization shall indicate deformation of C60 molecules

```
set mol [molinfo top]
set left [atomselect $mol {index < 60}]
set right [atomselect $mol {index >= 60}]
set nf [ molinfo $mol get numframes ]
for {set i 0} {$i < $nf} {incr i} {
    $left frame $i
    set com [measure center $left]
    set dlist ""
    foreach c [$left get {x y z}] {
        lappend dlist [veclength [vecsub $c $com]]
    }
    $left set user $dlist
    # repeat for second molecule...
}
```

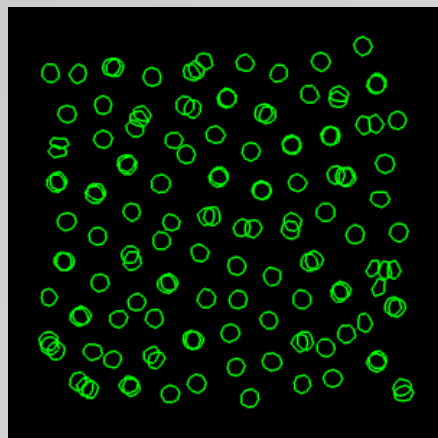
# Using VMD to Tailor Data



Simulations with PBC may have atoms wrapped back into the simulation cell.

Use VMD to make molecules whole:

```
package require pbctools  
pbcc join fragment
```



For trajectories, go to first frame, use:

```
pbcc join fragment  
pbcc unwrap -all  
pbcc wrap -compound fragment
```



# TopoTools Plugin

- [sites.google.com/site/akohlmey/software/topotools](https://sites.google.com/site/akohlmey/software/topotools)
- Complement to **psfgen**, which mostly assumes a system to be a (bio)polymer or solvent
- Easy building of topologies for simple polymers, nanostructures, system replication
- Allow manipulations on subsets of atoms
- Methods to manipulate bonds, angles, dihedrals, impropers and assign types
- Can read and write LAMMPS data files, but does **not** store or write potential parameters

# Common Interface

- Import with: **package require topotools 1.7**
- “**topo**” command is front end for most of API  
syntax: `topo <cmd> <flags> <global flags>`
- **namespace import ::TopoTools::\***  
to import convenience functions
- Common/global flags to “topo”:
  - `-sel <selection function or string>`, defaults to “all”
  - `-molid <molecule id>`, defaults to “top”

# TopoTools examples

- `topo guessatom <what> <from>`  
determine some atom properties heuristically  
`lammps` → data (combination of steps),  
`mass` → element, `name` → element,  
`element` → radius, `element` → mass
- `topo guessangles` or `topo guessdihedrals`  
or `topo guessimpropers <cutoff angle>`  
determine bonded interaction definitions based on  
the existing bond topology (can be guessed itself  
from atom-atom distance and atomic radii)
- `topo retypebonds`: atom types => bond types

# TopoTools: High Level Methods

- `topo readlammpsdata <file> <atom_style>`  
import data files from LAMMPS
- `topo writelammpsdata <file> <atom_style>`  
save coordinates and topology to data file
- `topo readvarxyz <xyz filename>`  
reads trajectories with varying number of atoms  
Visualize by selecting “user > 0.0” and  
activating “Update Selection Every Frame”  
order of atoms (-> bonds) is not preserved

# Convenience Functions

- `::TopoTools::replicatemols <mol> <nx> <ny> <nz>`  
like replicate command in LAMMPS
- `::TopoTools::mergemols {0 1 ...}`  
build new data set from multiple molecules
- `::TopoTools::selections2mols {sel1 sel2 ...}`  
like above, but takes atom selections
- `topo writegmxtop <filename>`  
write partial gromacs topology file for running  
gromacs' analysis tools (version 1.6+ also  
supports full CHARMM/psf to gmxtop export)

# LAMMPS Environment Variables

- **LAMMPSREMAPFIELDS**

allows to alias data fields in a LAMMPS custom dump to VMD supported fields (x,y,z,vx,vy,vz)

to set from within VMD use:

```
set env(LAMMPSREMAPFIELDS) {vx=fx}
```

- **LAMMPSDIPOLE2ATOMS**

if *value* > 0.0, replace point particle for dipole by two points representing the dipole vector and their distance the dipole magnitude times *value*



# LAMMPS Environment Variables

- **LAMMPSMAXATOMS**

Instruct the plugin to always return data for this many atoms. If there are fewer atoms in the dump file, "dummy" atoms are added.

```
set env(LAMMPSMAXATOMS) 10000
```

- **LAMMPSDUMMYPOS**

Set position of inactive or dummy atoms when using **LAMMPSMAXATOMS**. By default, the position (0.0,0.0,0.0) will be used.

```
set env(LAMMPSDUMMYPOS) {0.0 0.0 -10.0}
```