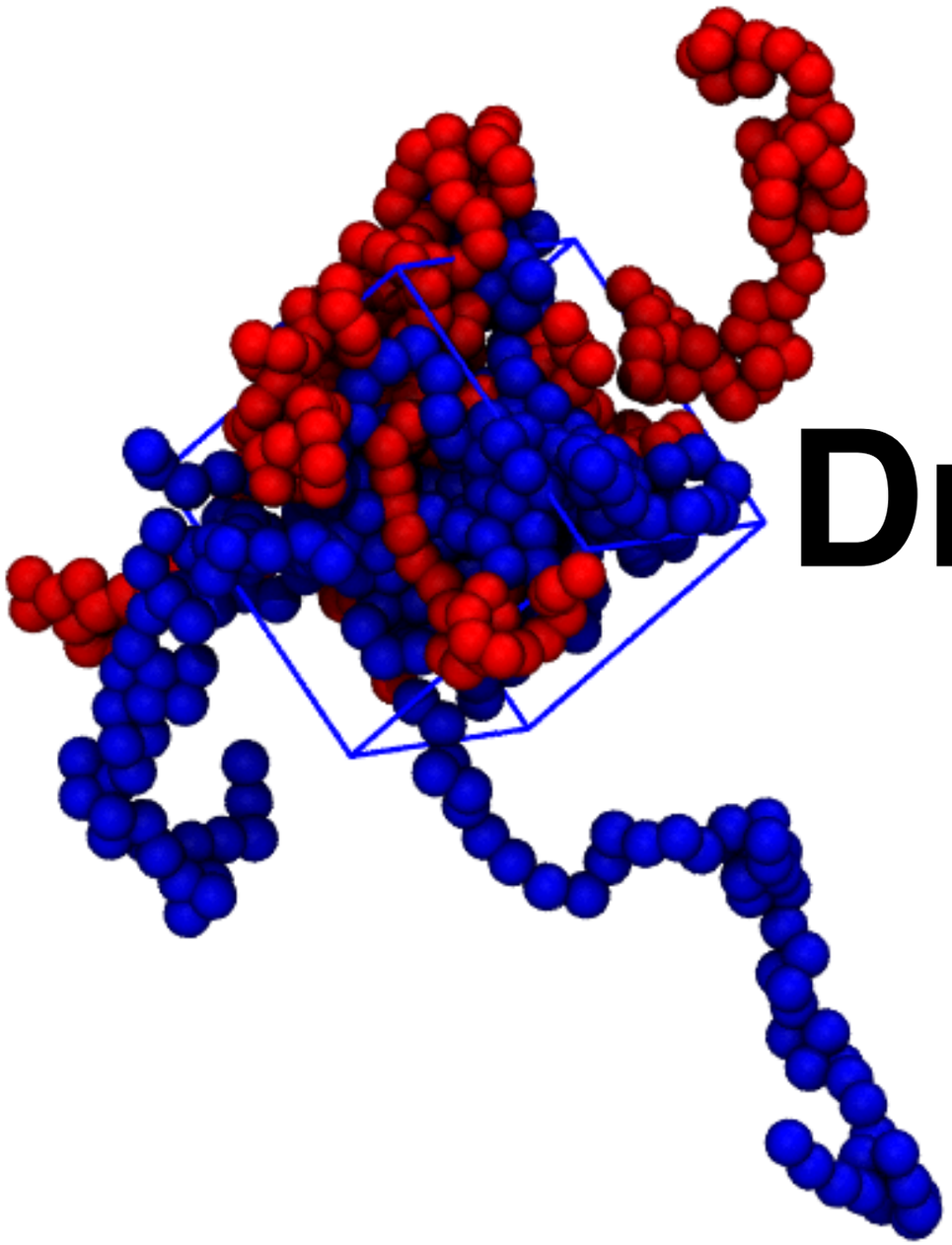


Draw me a picture!

and other cool things
to do with VMD and LAMMPS

Chris MacDermaid

24-March-2014

download companion files

ictp/vmd-viz/

About VMD

- Powerful tool for molecular visualization and analysis.

- Theoretical and Computational Biophysics Group at the University of Illinois Urbana-Champagne.

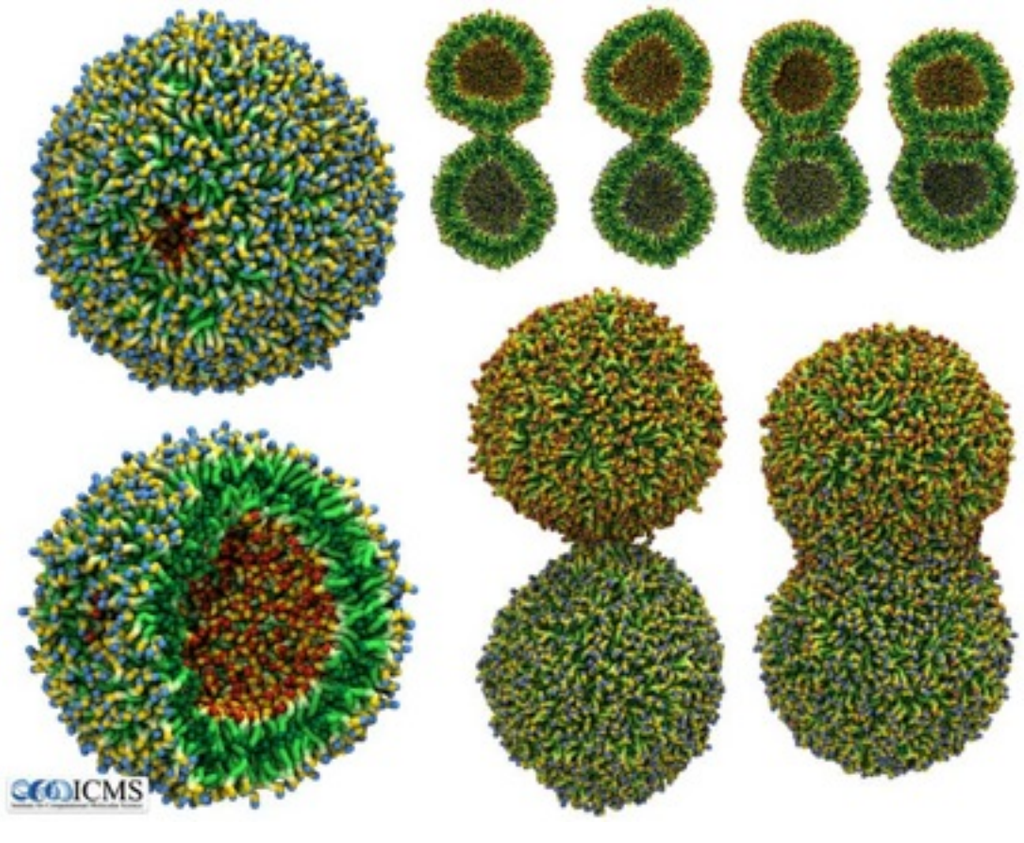
- John Stone, Senior Developer

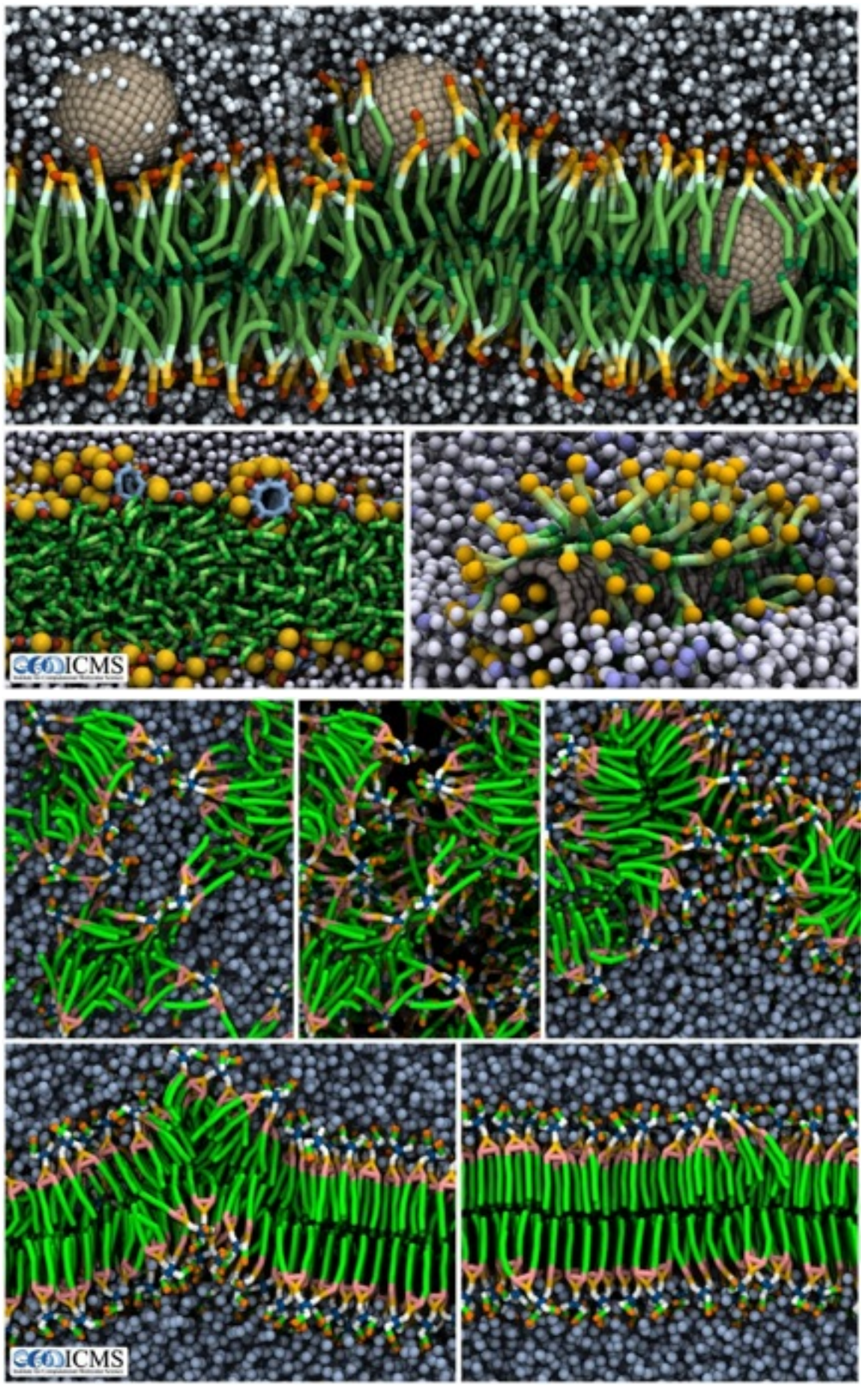
- http://www.ks.uiuc.edu/Research/vmd/

- VMD Mailing List

VMD Features

- Support for all major computer platforms
- Support for multicore processors
- Support for GPU accelerated computation
- No limits on the number of molecules, atoms, residues or number of trajectory frames, **except available memory**
- Many molecular rendering and coloring methods
- Extensive atom selection syntax for choosing subsets of atoms for display (includes boolean operators, regular expressions, and more)
- Support for over 60 molecular file formats and data types through an extensive library of built-in file reader/writer plugins and translators





Some nice images/movies courtesy of Axel

Installing VMD

- On windows/OSX: Install like anything else.
- Linux: (install to root-fs)

  1.Unpack the archive, change to created directory

  2.> ./configure; cd src; make install

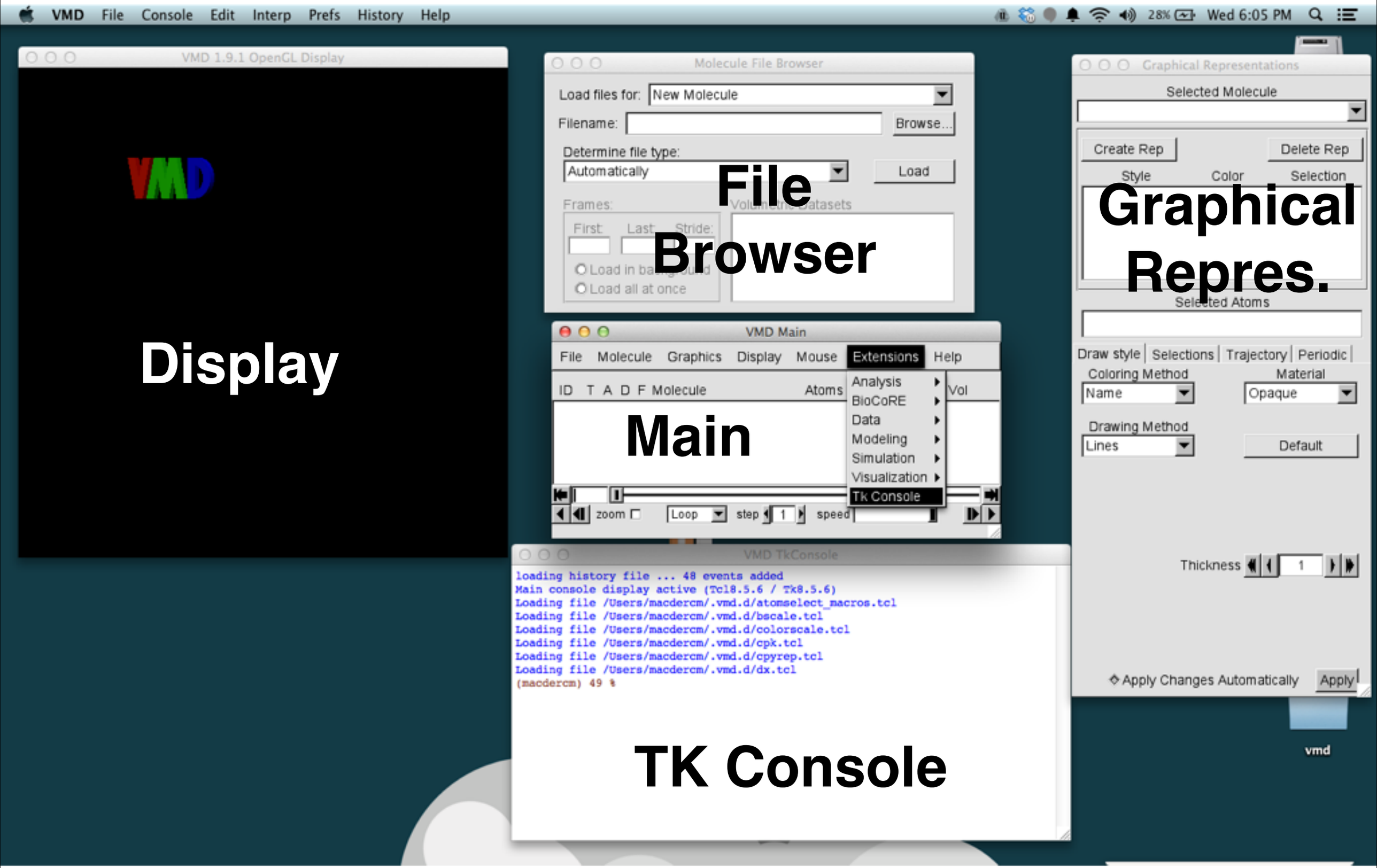
2.or Linux: (install to ${HOME})

  1.Unpack the archive, change to created directory

  2. Edit the configure file: `nano ./configure`

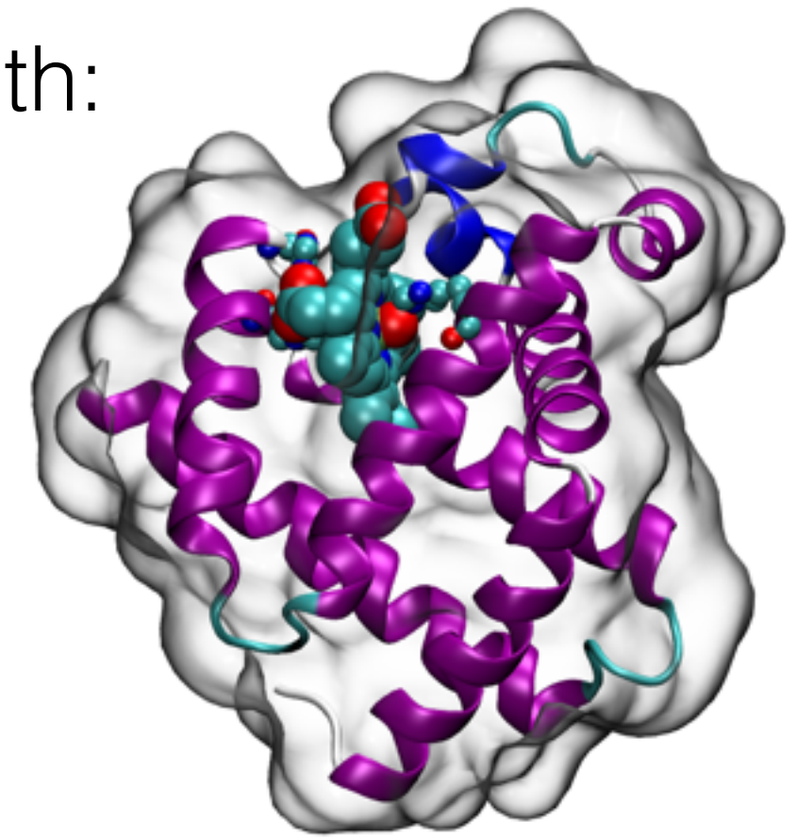
```
$install_bin_dir="\${HOME}/pkg/bin";
$install_library_dir="\${HOME}/pkg/lib/$install_name";
```

  3.> `mkdir -p ${HOME}/pkg/lib ${HOME}/pkg/bin; cd src; make install;`
  `export PATH=${HOME}/pkg/bin:${HOME}`


VMD File Console Edit Interp Prefs History Help
28% Wed 6:05 PM
VMD 1.9.1 OpenGL Display
Display
Molecule File Browser
Load files for: New Molecule
Filename:
Browse...
Determine file type:
Automatically
Load
Frames:
First:
Last:
Stride:
Load in background
Load all at once
Volumetric Datasets
File Browser
VMD Main
File Molecule Graphics Display Mouse Extensions Help
ID T A D F Molecule
Atoms
Vol
Analysis
BioCoRE
Data
Modeling
Simulation
Visualization
Tk Console
Main
zoom
Loop
step
1
speed
Graphical Representations
Selected Molecule
Create Rep
Delete Rep
Style
Color
Selection
Graphical Repres.
Selected Atoms
Draw style
Selections
Trajectory
Periodic
Coloring Method
Name
Material
Opaque
Drawing Method
Lines
Default
Thickness
1
Apply Changes Automatically
Apply
vmd
VMD TkConsole
loading history file ... 48 events added
Main console display active (Tcl8.5.6 / Tk8.5.6)
Loading file /Users/macdercm/.vmd.d/atomselect_macros.tcl
Loading file /Users/macdercm/.vmd.d/bscale.tcl
Loading file /Users/macdercm/.vmd.d/colorscale.tcl
Loading file /Users/macdercm/.vmd.d/cpk.tcl
Loading file /Users/macdercm/.vmd.d/cpyrep.tcl
Loading file /Users/macdercm/.vmd.d/dx.tcl
(macdercm) 49 %
TK Console

10 minute introduction to VMD with Myoglobin

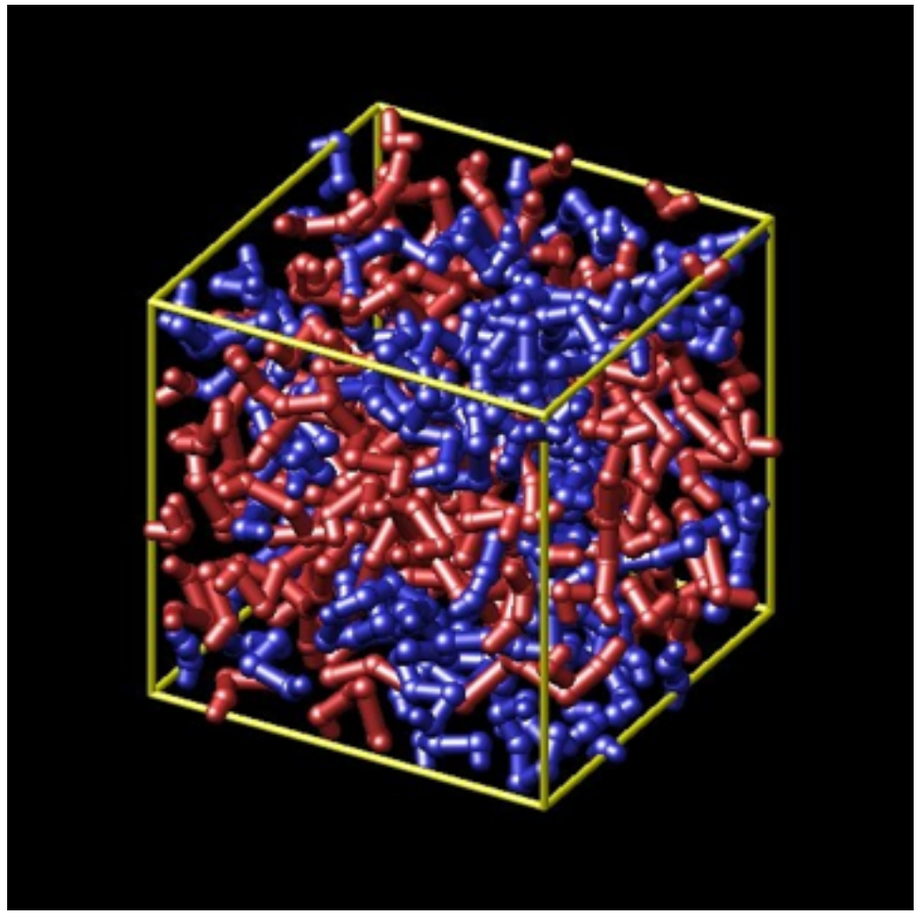
- You should become familiar with:



  - VMD's Main window
  - Loading molecules
  - Changing Representations
    - Coloring Methods, Drawing methods, Materials
  - Highlighting particular groups of atoms

myoglobin VMD Userguide

Image/Move Creation with LAMMPS



- **dump image**

- write an image every 200 steps

```
dump d1 all image 200 file.*.jpg element element
dump_modify d1 pad 5 element N O
```

- Enlarge the image and zoom in a bit. Color by type.

```
dump d1 all image 200 file.*.jpg type element size
1024 1024 zoom 1.2
dump_modify d1 pad 5 element N O
```

03/in.03

03/in.03

```
# in.03
# 3d random polymer

atom_style bond
units lj

read_data data.ab
special_bonds fene

# Step 1: soft potential push-off to remove overlaps

pair_style   soft 1.12246
pair_coeff   * * 0.0 1.12246

# these have to match the data in def.chain.ab
bond_style   harmonic
bond_coeff   1 50.0 0.97
bond_coeff   2 50.0 1.17

# make type 2 particles a bit heavier
mass 2 1.5

velocity all create 0.45 2349852

fix      1 all nve
# use a crude implicit solvent model
fix      2 all langevin 0.45 0.45 0.02 8742
```

LAMMPS input for random polymer

lmp_g++ < in.03

LAMMPS input for random polymer

```
# use this to ramp up the height of the soft core repulsion
variable prefactor equal ramp(1.0,20.0)
fix      3 all adapt 1 pair soft a * * v_prefactor

thermo       50
run          10000

unfix        3
unfix        2
#undump d1

neighbor 0.5 bin

# Main run
pair_style   lj/cut 5.0

pair_coeff   1 1 1.0 1.0
pair_coeff   1 2 1.1 1.122462
pair_coeff   2 2 1.2 1.2
thermo       1000

reset_timestep   0

dump d1 all image 200 snap-03.*.jpg element element
dump_modify d1 pad 5 element N O

run      10000
```

03/in.03

lmp_g++ < in.03

Can you adjust this line to output a movie? Coordinates of the atoms?

Image/**Move** Creation with LAMMPS

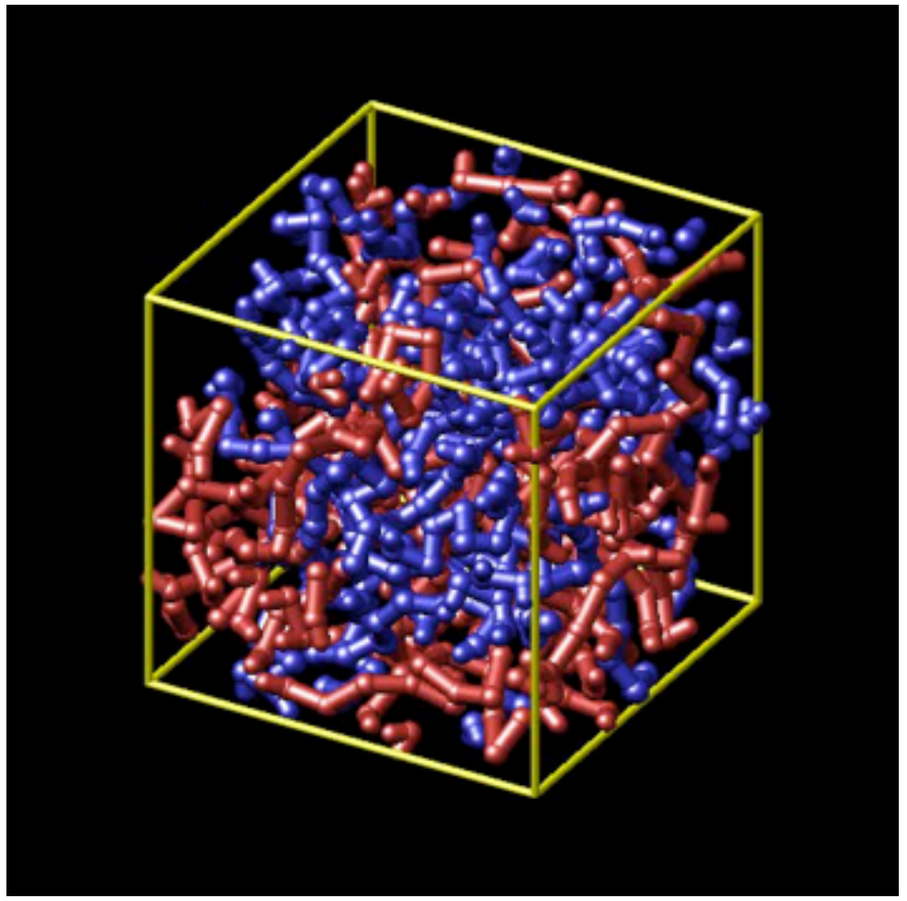
- Piece together individual images to make a movie using ffmpeg

```
> ffmpeg -i file.%05d.jpg movie.mp4
```

- Filenames must be sequential:
  file.00001.jpg
  file.00002.jpg
  file.00003.jpg
  ....

```
sh -c 't=0 ; for s in file.?????.jpg; do mv $s file.$t.jpg ; t=`expr $t + 1`; done'
```

Image/**Move** Creation with LAMMPS



- **dump movie**

- create a movie with 200 steps between frames

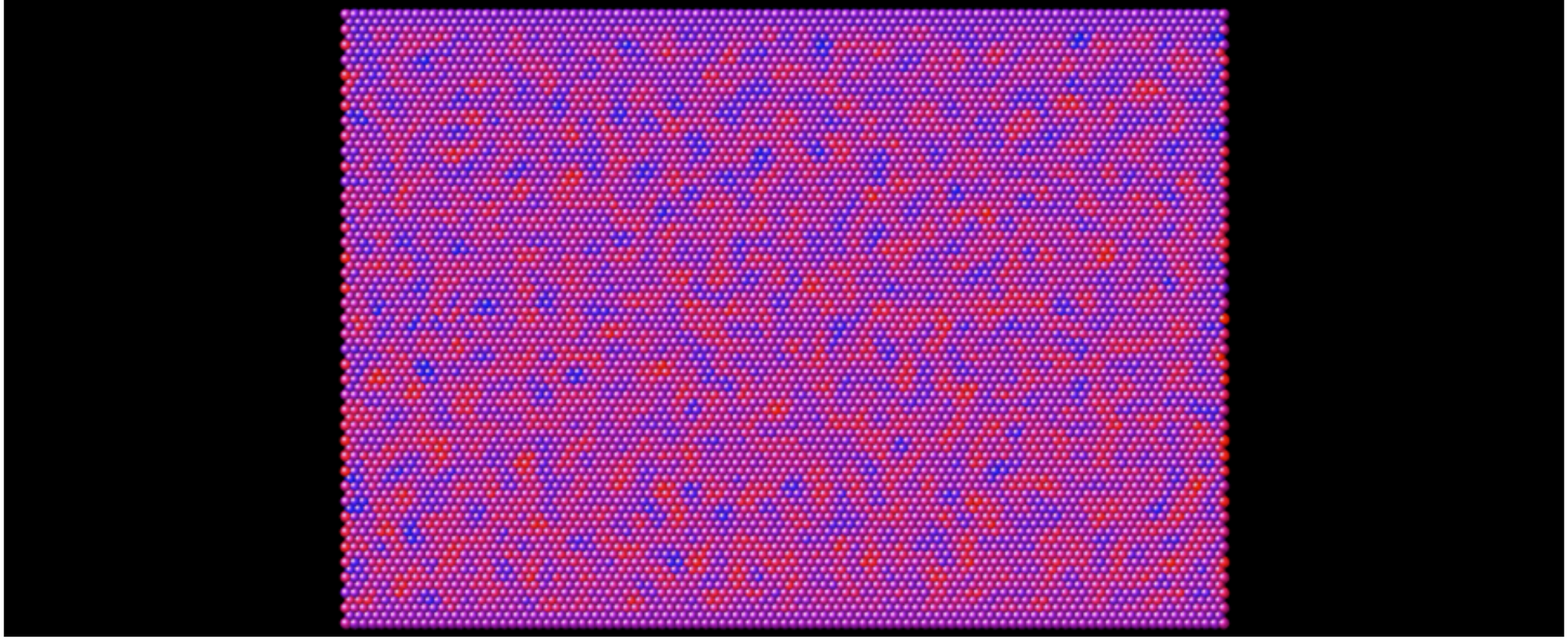
```
dump d1 all movie 200 file.mpg element element
dump_modify d1 element N O
```

- Enlarge the movie and zoom in a bit

```
dump d1 all movie 200 file.mpg element element
size 1024 1024 zoom 1.2
dump_modify d1 element N O
```

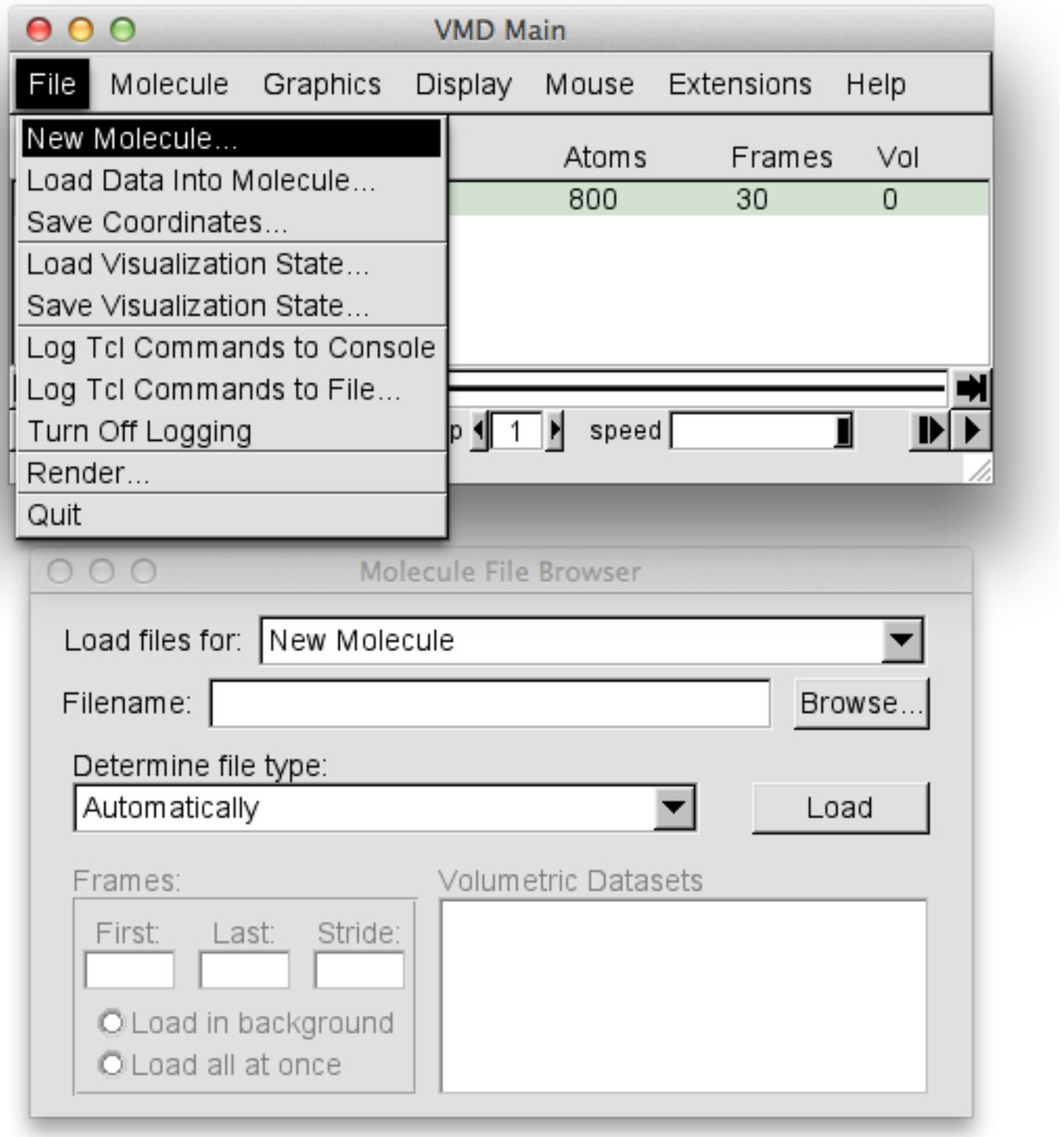
03/in.03

Axel Kohlmeyer



```
compute         strain all stress/atom
variable        mystrain atom c_strain[1]+c_strain[2]
dump            1 all image 50 snap-movie.*.ppm v_mystrain type box no 0.0 center s 0.5 0.7 0.5 &
                zoom 2.0 adiam 1.6 ssao yes 3333333 0.4 size 1920 1080
dump_modify     1 pad 5
run             15000

shell sh -c 't=0 ; for s in snap-movie.?????.ppm; do mv $s snap-movie.$t.ppm ; t=`expr $t + 1`; done'
shell ffmpeg -y -an -i:v snap-movie.%d.ppm -r 24 -b:v 2400k -c:v libx264 crack-strain.mp4
```

Loading a molecule/ trajectory into VMD



- Use the GUI: “New Molecule”
- Use the ‘mol’ command:

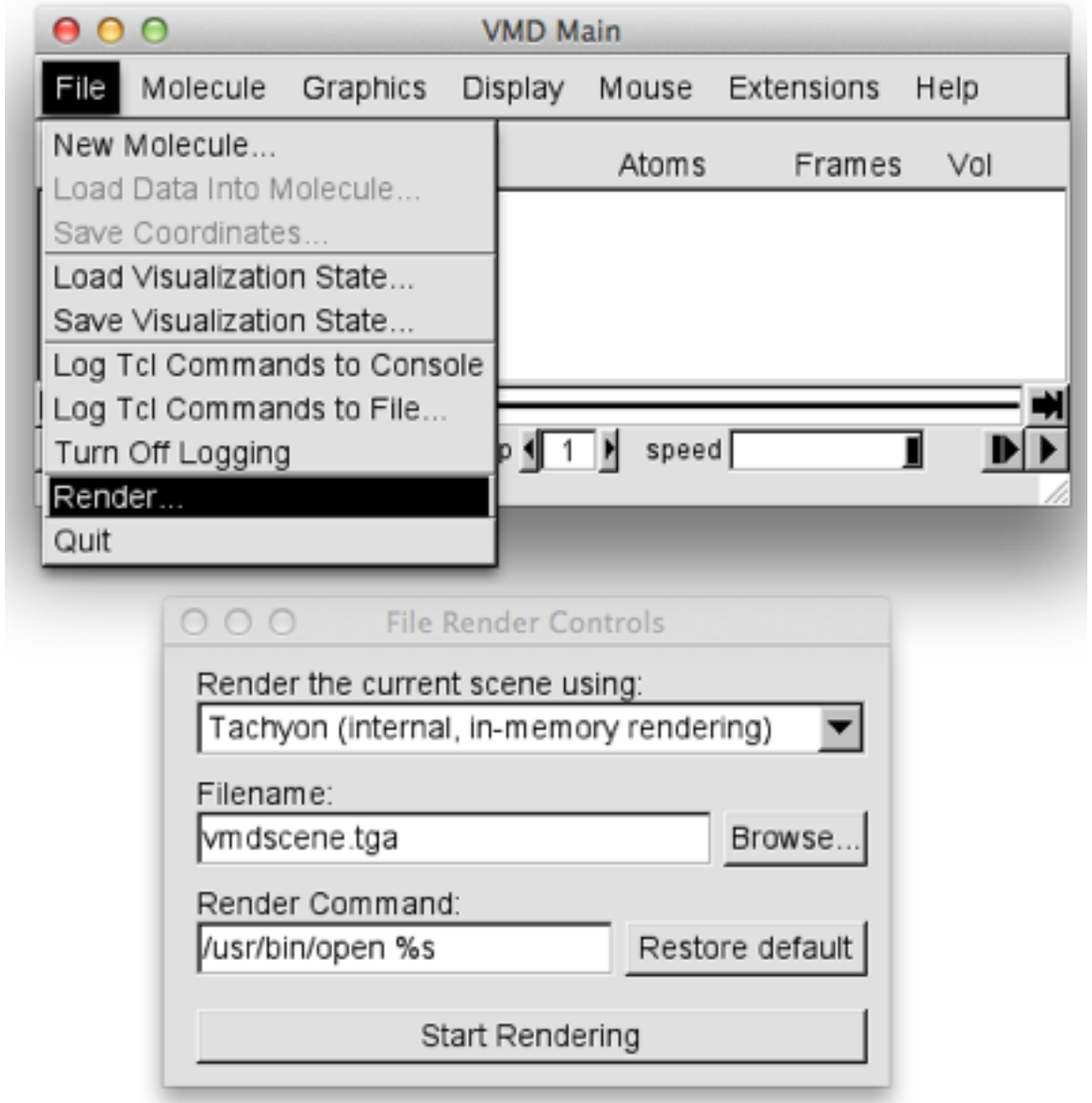
```
mol new myfile.xyz
mol addfile mytraj0.dcd
mol addfile mytraj1.dcd
mol addfile mytraj2.dcd
.....
```

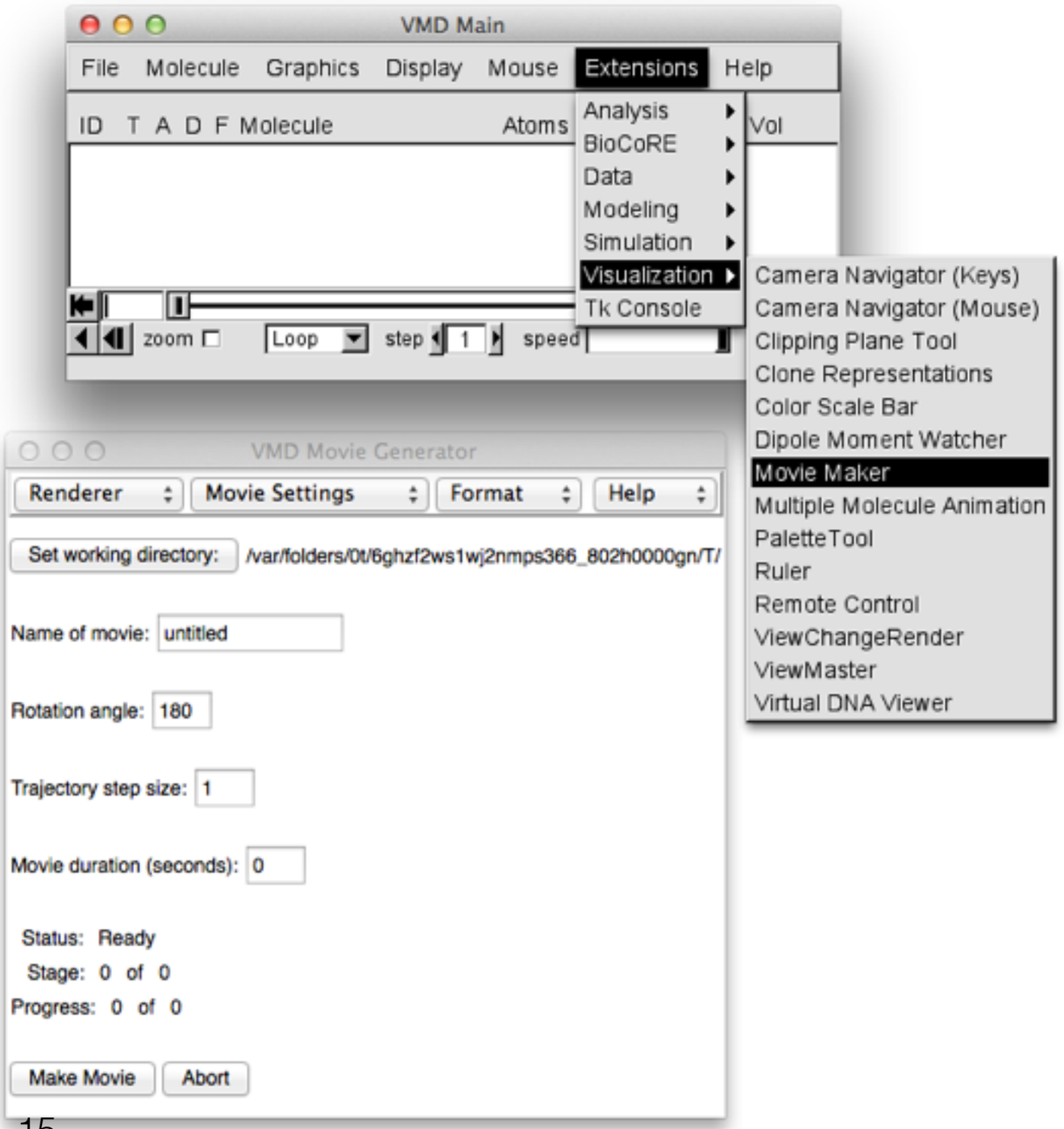
- Lammps data:

```
topo readlammpsdata file.data
```

Image/Move Creation with VMD

- The GUI provides both a render menu and movie-maker tool.





Image/**Move** Creation with VMD

- Use GUI. (What fun is this??)
- Use TCL.
  - Load trajectory and LAMMPS topology (.data) files.

```
topo readlammpsdata my.data
mol addfile my.dcd
```

Requires TopoTools

  - Adjust scene to your liking: orientation, colors, materials, drawing mode, display size. Save a state file for easy reloading.
  - **Movie:** Simple ‘for’ loop over vmd frames, calling your preferred rendering engine in the body of the loop.

```
‘render engine filename options’
```

```
# movie03.tcl
# To run, in the vmd tckon type: 'source movie03.tcl'

# Load a LAMMPS data file, a trajectory,
# set some representations to our liking,
# adjust the display size and rotate/zoom,
# loop over frames and render

# Script requires TopoTools and pbctools
package require topotools
package require pbctools

# Load the data file and trajectory dcd
set molid [topo readlammpsdata data.ab]
mol addfile 03.dcd type dcd waitfor all

# Set the representation, delete the initial representation
# created by vmd
mol delrep 0 top
mol representation CPK 2.0 0.30 10.0 10.0
mol color Name
mol selection {all}
mol material AOEdgy
mol addrep top
```

Load Necessary Packages

Load Files

Create Representation

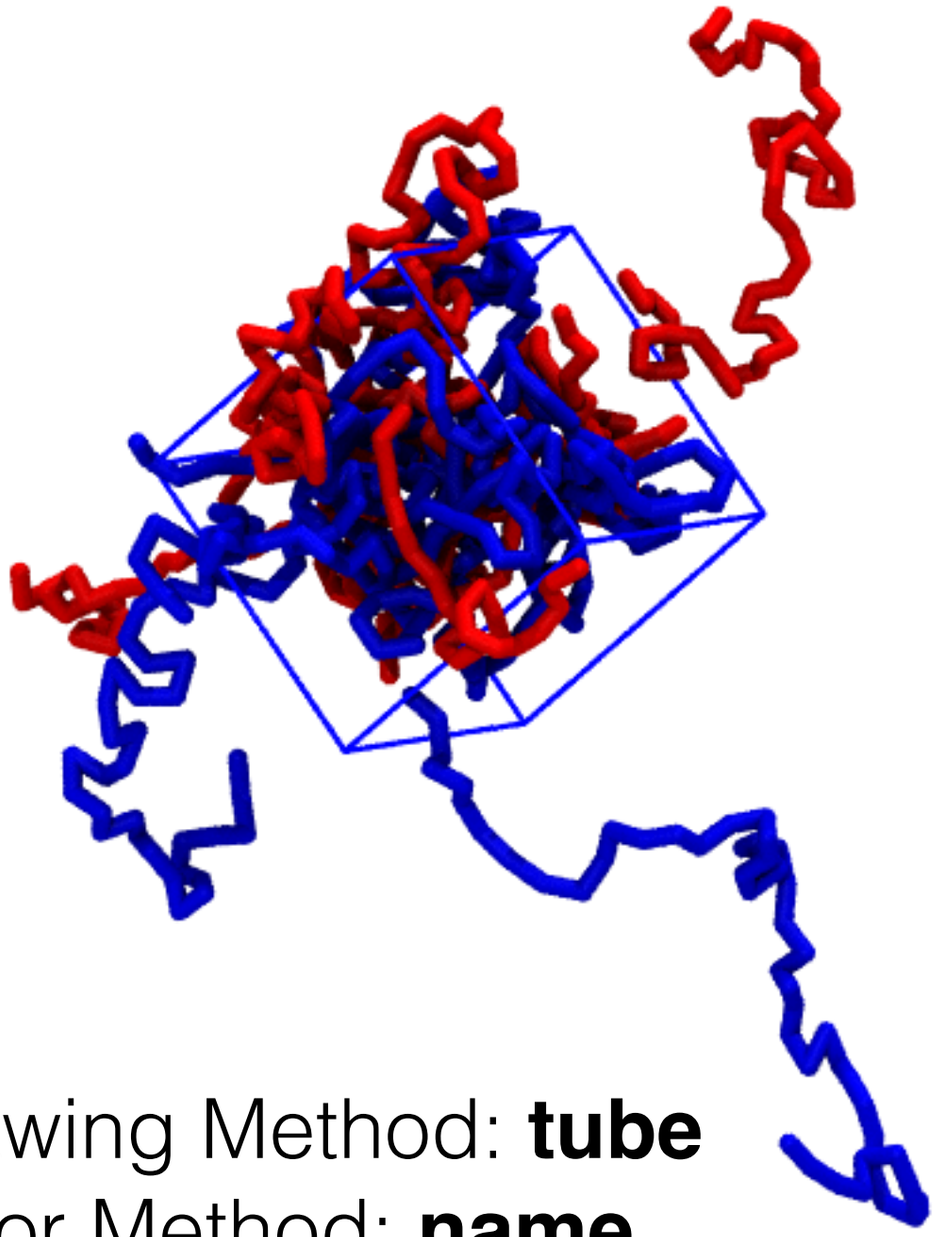
```
# Set the scene: adjust display height
# and rotate axes a bit, set the background
# to white
display height 15
rotate x by -45; rotate y by 45; rotate z by 45
color Display Background white

# Draw the periodic box centered at the origin
pbc box_draw -center origin

# Get the number of frames
set nframes [molinfo $molid get numframes]

# Loop over frames, calling render.
# Format the output filename as snap-03.00001.tga
# Skip the first "junk" frame from lammpsdata
for {set i 1} {$i < $nframes} {incr i} {
      render TachyonInternal\
        [format “snap-03.%05d.tga" $i]
}
## Call external script to encode movie
exec ./03-encode-vmd.sh
```

Set Scene

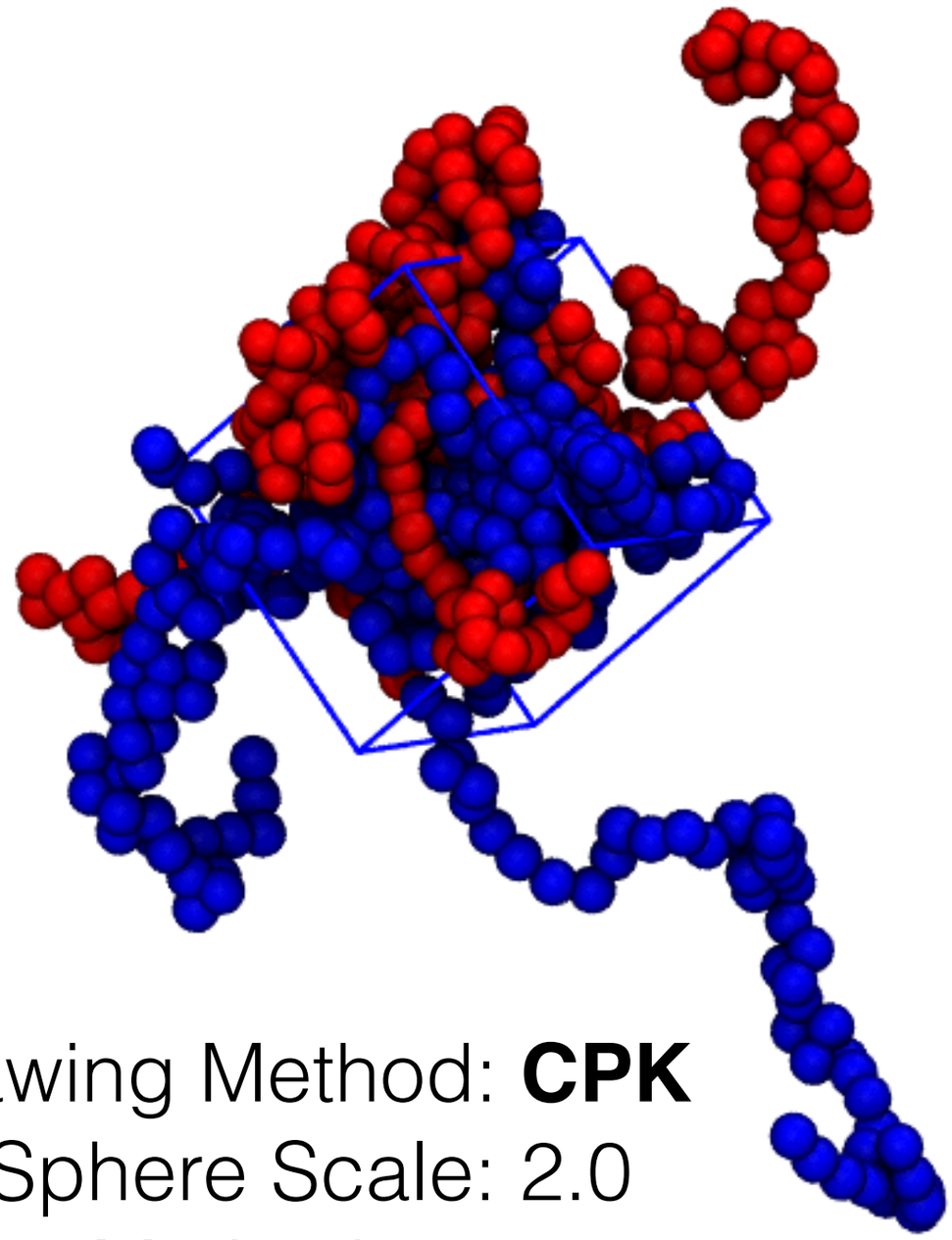
Render Frames

- > ./03-encode-vmd.sh
- -or-
- > ffmpeg -i snap-03.%05d.tga 03.movie.mp4

Combine frames to make a movie

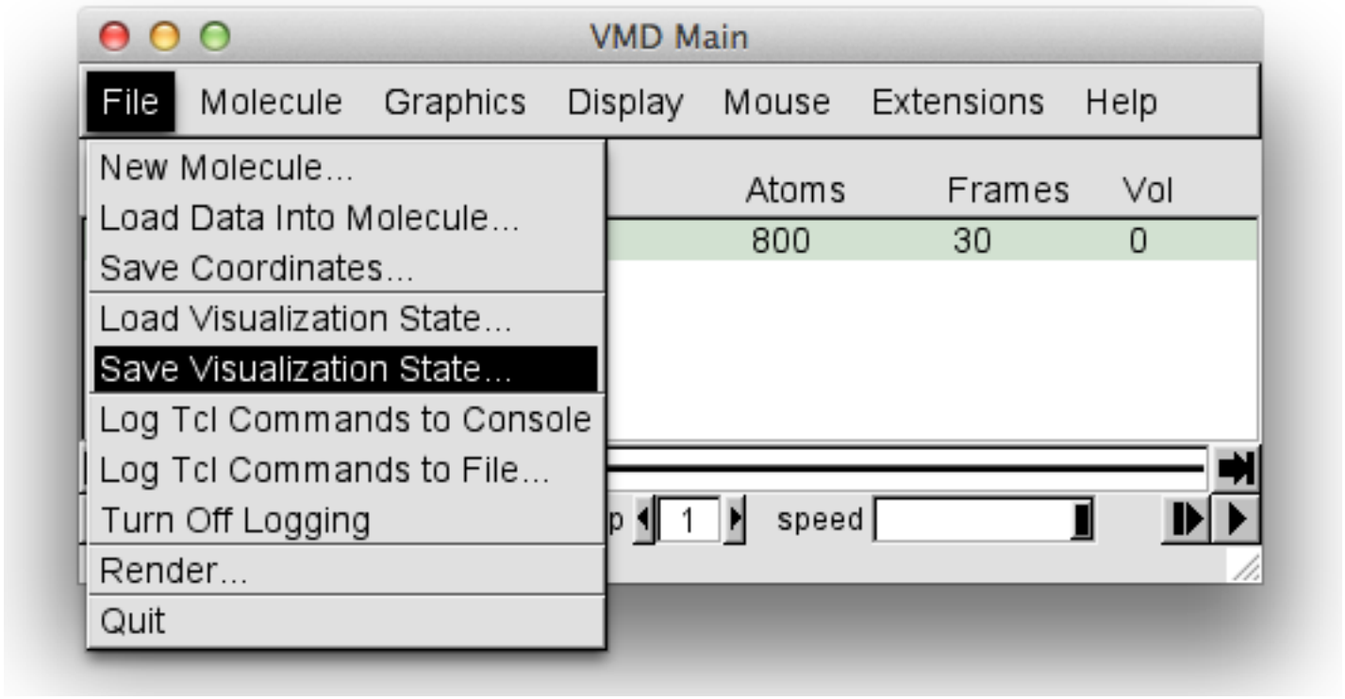


Drawing Method: **tube**
Color Method: **name**
Material: **ao shiny**
Ambient Occlusion: **yes**



Drawing Method: **CPK**
Sphere Scale: 2.0
Color Method: **name**
Material: **ao edgy**
Ambient Occlusion: **yes**

Image/**Move** Creation with VMD



Once you're happy with your scene, make sure you save its state using the 'save visualization state' option!

VMDs true power comes from the TK (TCL) console

Interact with VMD by issuing commands in the TCL scripting language!

TCL (tickle)

- A “high level” interpreted scripting language particularly adept at text processing and interfacing with C, C++ codes. Has a closely related GUI construction language “tk”.

- All data types are manipulated as strings.

- VMD has a built in TCL interpreter for interacting with its internal data structures including molecules, atoms or **groups of atoms**, visualization, rendering, vector/ matrix operations, topologies and **data analysis**.

TCL Syntax

- `command -flag -flag -flag <args> <args>`
- Arguments are typically **$variables** containing strings or one of two data structures: the “**list {}**” or the “**array()**, aka, hash-table”
- Example: Search a list of elements, return the index of the first matching element:
  - `lsearch {a b c d e c} “c” => 2`
  - `lsearch -index 1 { {a b c d e} {f g h i j} } “g” => {f g h i j}`

TCL Lists

- Define and print a list:
  - `set mylist {3 5 1 4 2} or [list 3 5 1 4 2]`
  - `puts $mylist => 3 5 1 4 2`
- Access a list element:
    - `puts [lindex $mylist 2] => 1`
- Sort the list:
  - `puts [lsort -increasing $mylist] => 1 2 3 4 5`

TCL Lists

- Change a list element
  - `lset mylist 2 9 => 3 5 9 4 2`
- Iterate over each element in a list and print it
  - `foreach x $mylist {puts $x} =>`

```
3
5
1
4
2
```

TCL Lists

- Loop over two elements at a time and print them
  - `foreach {x y} $mylist {puts “$x $y”}` =>
    3 5
    1 4
    2
- Append an element to a list
  - `lappend mylist a` => `3 5 1 4 2 a`

TCL Lists

- Assign each element in a list to a variable:
  - (**tcl8.5**)
```
lassign {1 2 3} a b c
puts “$a $b $c” => 1 2 3
```
  - (**tcl 8.4**)
```
foreach {a b c} {1 2 3} {break}
puts “$a $b $c” => 1 2 3
```
- Unset a variable or list:
```
unset mylist
```

TCL Lists

- Variables contained within `{$a $b $c}` are not substituted. How then can we define a list of variables?

```
set a 1; set b 2; set c 3
set mylist [list $a $b $c]
```

TCL Resources

- http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
- http://www.tcl.tk/man/tcl8.5/
- http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html

VMD's 'Atomselect' Command

- `set sel [atomselect molid "selection text"] => atomselect0`

- Primary method for interacting with atoms

- Returns a handle to an internally created proc that references the atoms satisfying the selection text criteria.

  - `e.g. "all", "resname popc", "chain A", "residue 5", "resid 10 to 20", "resid 10 20"`

VMD's 'Atomselect' Command

- `set sel [atomselect molid "selection text"]`
  `=> atomselect0`

- Combine selection criteria using logical operators:

- "and", "or", "not", >, <, =

- "protein and chain A and x > 5"
  "not (water or protein)"
  "name "O.*" and charge < 0.5"

Special “phrases”

- Select all water atoms within 5 angstrom of protein

  “water within 5 or protein” vs.
  “water exwithin 5 of protein”

- Make sure we get complete water molecules:

  same residue as (water exwithin 5 of protein)

VMD's 'Atomselect' Command

- List all keywords: "`atomselect keywords`"

- name type **index serial** atomicnumber element **residue** resname resid chain segname segid x y z radius mass charge beta occupancy all none

- fragment numbonds backbone sidechain protein nucleic water waters helix alpha_helix helix_3_10 pi_helix sheet betasheet beta_sheet turn coil structure phi psi sequence

- user user2 user3 user4

- sqr sqrt abs floor ceil sin cos tan atan asin acos sinh cosh tanh exp log

- acidic cyclic acyclic aliphatic alpha amino aromatic basic bonded buried cg charged hetero hydrophobic small medium large neutral polar purine pyrimidine surface lipid ion sugar solvent carbon hydrogen nitrogen oxygen sulfur noh heme

Selection Querying

- `set sel [atomselect molid “selection text”] => atomselect0`

- Querying a selection:
  ```
  set idx [$sel get index]=>
  {0 1 2 3 4 5}
  ```

- returns the value of the keyword requested for **each atom as a TCL list**

- Atoms are returned in the order in which they were loaded/created in VMD. **NOT** the order you specify:

  - `set sel [atomselect molid “name CB CA”]` and `set sel [atomselect molid “name CA CB”]` return identical lists!

Selection Querying

- Number of atoms selected
  `$sel num`

- The selection text:
  `$sel text`

- The associated molecule (molid)
  `$sel molid`

Selection Querying

- Querying multiple keywords (performance considerations)
  - <u>Two separate selections:</u>

```
set id [$sel get id]
set an [$sel get name]
```

  - **Vs.** Aggregated list (expensive, but has its uses)

```
set id_an [$sel get {index name}] =>
{{0 N} {1 CA} {2 CB} {3 CG}}
```

Modifying Atom Properties

- Change selected atoms' name to CA

```
$sel set name CA
```

- Change selected atoms' chain to "A"

```
$sel set chain A
```

- Change the selected atoms' residue name to"POPC"

```
$sel set resname POPC
```

- Set the coordinates of selected atoms to {1 2 3}

```
$sel set {x y z} {1 2 3}
```

Coordinate Manipulation

- Translate the selected atoms -5 angstroms along z-axis

```
$sel moveby {0.0 0.0 -5.0}
```

- Rotate the selected atoms 180 degrees about z-axis

```
$sel move [transaxis z 180.0 deg]
```

- Rotate selected atoms 104 degrees about bond having coordinates r1, r2

```
$sel move [trans bond r1 r2 104 deg]
```

Putting it together

- How could you get the total number of different atom names and their corresponding values for a given structure?

- How could you renumber all the residues in all chains from a mis-formatted PDB file?

- How could you select all atoms within 5A of the origin?

- How about all atoms within a cube with edge length 3A centered at the origin? What if I wanted entire residues/ fragments not just atoms?

Putting it together

```
## Get unique atom names

set sel [atomselect top “all”]
set names [lsort -unique [$sel get name]]
puts [llength $names]
$sel delete
```

Putting it together

```
#Renumber residues in each chain from 1 to N

set sel [atomselect top “all”]
foreach c [lsort -unique [$sel get chain]] {
   set i 1
   foreach id [lsort -unique [$sel get residue]] {
      set sel2 [atomselect top “chain $c and resid $id”]
      $sel2 set resid $i
      incr i
      $sel2 delete
   }
}
$sel delete
```

Putting it together

```
# Select all atoms within 5 angstroms of the origin
sqrt(x*x + y*y + z*z) < 25
```

```
# Select all atoms within a cube with
# edge length 3 angstroms
x > -3 and x < 3 and y > -3 and y < 3 and z > -3 and z < 3
```

Putting it together

- How could you construct a lookup (hash) table that uses the atom index as a key, and the value being the atom type?

- idx(0) type0
  idx(1) type1
  idx(2) type2
  ….

Putting it together

```
# Build a lookup table associating
# atom index to atom type


set sel [atomselect top “all”]
foreach idx [$sel get index] type [$sel get type] {
    set idx($idx) $type
}
```

Putting it together

- How would you write a procedure to output a PDB formatted file using VMD's atomselection functionalities?

- `proc writePDB {sel} {`
  `...`
  `...`
  `}`

- Note: there is already the command '`$sel writepdb file.pdb`' command, this is just an exercise!

```
#Write out a PDB formatted file for a selection
proc writePDB {sel} {
# Loop over each atom property in parallel
foreach charge [$sel get charge]\
      type [$sel get type]\
      atomname [$sel get name]\
      resname [$sel get resname]\
      resid [$sel get resid]\
      X [$sel get x]\
      Y [$sel get y]\
      Z [$sel get z]\
      chain [$sel get chain]\
      occ [$sel get occupancy]\
      beta [$sel get beta]\
      segid [$sel get segid]\
      ele [$sel get element]\
      serial [$sel get serial] {

    # Display it using the PDB format
    puts [format "%-6s%5d %4s %3s %s%4d    %8.3f%8.3f%8.3f%6.2f%6.2f       %-4s%2s"\
                 "ATOM  " $serial $atomname $resname $chain\
                 $resid $X $Y $Z $occ $beta $segid $ele]
  }
}
```

```
>Main< (writepdb) 246 % writePDB $sel
ATOM   1190  CHA HEM A 154   -5.248  39.769  -0.250  1.00  7.67           C
ATOM   1191  CHB HEM A 154   -3.774  36.790   3.280  1.00  7.05           C
ATOM   1192  CHC HEM A 154   -2.879  33.328   0.013  1.00  7.69           C
ATOM   1193  CHD HEM A 154   -4.342  36.262  -3.536  1.00  8.00           C
ATOM   1194  C1A HEM A 154   -4.960  39.256   0.996  1.00  7.60           C
ATOM   1195  C2A HEM A 154   -5.177  39.925   2.270  1.00  7.96           C
ATOM   1196  C3A HEM A 154   -4.762  39.099   3.235  1.00  7.45           C
ATOM   1197  C4A HEM A 154   -4.279  37.882   2.615  1.00  7.44           C
ATOM   1198  CMA HEM A 154   -4.792  39.352   4.760  1.00  8.07           C
ATOM   1199  CAA HEM A 154   -5.789  41.338   2.464  1.00  7.92           C
ATOM   1200  CBA HEM A 154   -7.283  41.203   2.814  1.00  9.61           C
ATOM   1201  CGA HEM A 154   -8.000  40.335   1.803  1.00 10.04           C
ATOM   1202  O1A HEM A 154   -8.314  39.154   2.116  1.00 11.60           O
ATOM   1203  O2A HEM A 154   -8.252  40.816   0.681  1.00 11.56           O
ATOM   1204  C1B HEM A 154   -3.405  35.579   2.720  1.00  7.15           C
```

NICE!

TopoTools: Constructing/ Editing Molecular Topologies

- **Created by Axel Kohlmeyer**

- Tools for getting/setting/guessing molecular topologies using VMDs atomselection paradigm

- Support for reading/writing LAMMPS data files

- Load TopoTools into tcl interpreter:
  ```
  package require topotools
  ```

- Use the ‘topo’ command
  ```
  topo somecommand someargument [...] [-sel ] [-mol |top]
  ```

TopoTools: Constructing/ Editing Molecular Topologies

- TopoTools primary function is to provide an interface for manipulating molecular structure, including:
  - adding and deleting bonds, angles, dihedrals
  - Associating atom types with bonds, angles, dihedrals
  - Guessing missing bonds, angles and dihedrals
  - Reading/Writing LAMMPS data files.

TopoTools: Constructing/ Editing Molecular Topologies

- Get a list of bonded connectivities based on atom index
  `topo getbondlist => {{0 1} {1 2} {2 3} {3 4}}`

- Guess angles based on bonded connectivity

```
topo -sel $sel guessangles
topo -sel $sel getanglelist =>
{{C-CA-N 0 1 2} {CA-C-O 1 2 3} {N-C-O 3 2 4}
{CA-C-N 1 2 4} {C-N-CA 2 4 5}}
```

- Write out a LAMMPS data file
  `topo writelammpsdata file.data full`

Putting it all together: Building a Water box using just VMD

- You can use VMD to construct a water box which can then be simulated in LAMMPS. Here's how you can do it yourself:

1. Use VMDs vector and matrix routines to construct a single water molecule having TIP3P geometry.

2. Use VMDs selection routines to set the water molecule's properties: name,type,mass,charge,radius

3. Use topotools::replicatemol command to create a box of these water molecules.

4. Use TopoTools to assign bonds/angles to all molecules.

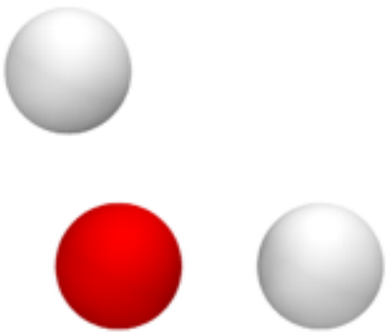
5. Use TopoTools to output a LAMMPS data file

waterbox/waterbox.0.tcl

```
#!/usr/bin/tclsh

proc build_tip3 {} {

    # Create a new molecule that holds three "empty" atoms
    # return the molecule id and store in molid
    set molid [mol new atoms 3]

    # Add a frame to hold atomic data
    animate dup $molid

    # Define O-H bond length in angstroms and H-O-H angle in degrees
    set r 0.9572; set theta 104.52

    # Use VMDs vector/matrix routines to move the atoms into their
    # equilibrium positions and assign their coordinates
    set o {0.0 0.0 0.0};       # coordinate of Oxygen atom
    set h1 [list $r 0.0 0.0]; # coordinate of first hydrogen atom

    # We need to calculate the position of the second hydrogen atom
    # by rotating the v = {h1-o} vector a value $theta in the xy-plane
    set v [vecsub $h1 $o]

    # Create rotation matrix corresponding to a rotation about the
    # z axis a value $theta
    set R [transabout {0.0 0.0 1.0} $theta deg]

    # Multiply the vector v with R, this is the new position of atom h2
    set h2 [vectrans $R $v]
```

Example: VMD Water box

1. Build a TIP3P Water



waterbox/waterbox.0.tcl

Example: VMD Water box

1. Build a TIP3P Water

2. Apply Atomic Properties

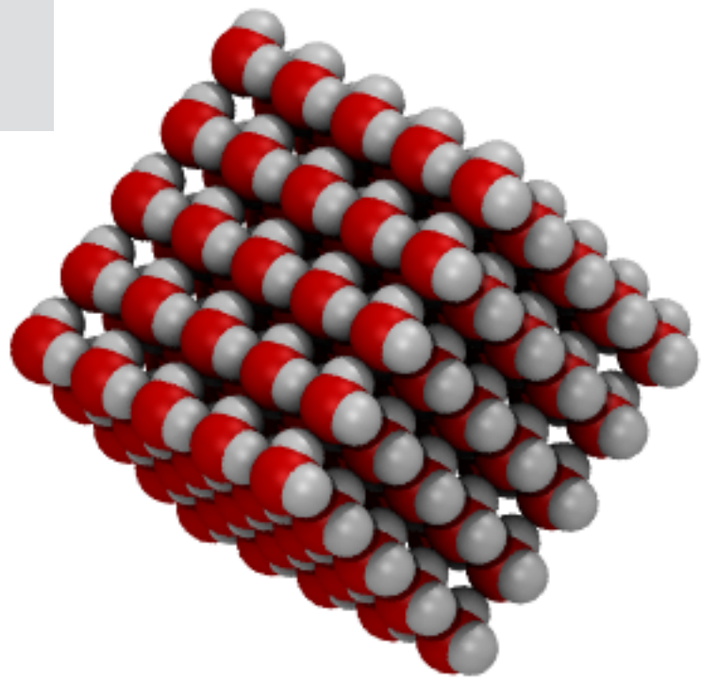
```
# Create a selection of the three "empty" atoms so we can
    # set their coordinates and their molecular properties
    set sel [atomselect $molid "all"]
    set xyz [list $o $h1 $h2]
    $sel set {x y z} $xyz

    # Now we need to set the atomic mass, charge radius, and name of each
    # of the atoms. We create a list-of-sublists, where each sublist
    # contains the properties of each atom. The order of the properties
    # in each sublist must be consistent across the sublists
    # { {o_mass  o_charge  o_radius  o_name  o_type}\
        #   {h1_mass h1_charge h1_radius h1_name h1_type}\
        #   {h2_mass h2_charge h2_radius h2_name h2_type}}
      set props { {15.9994 -0.830 1.299 OH OT}\
                  {1.008    0.415 1.00  H1 HT}\
                  {1.008    0.415 1.00  H2 HT} }
    $sel set {mass charge radius name type} $props

    # reanalyze the mol to let vmd know the
    # atoms have their properties set
    mol reanalyze $molid

    # Delete our selection (always a good idea)
    $sel delete

    # Have the procedure return our new molecules ID
    return $molid
}
```

waterbox/waterbox.1.tcl

```
proc makebox {molid} {

    # We require the package TopoTools to replicate our
    # water molecule, set our bonding and angle topologies,
    # and write out our input lammps data file.

    package require topotools

    # Set the edge length of our periodic box to
    # the 2*sigma_O-O bond (3.1507) length.
    molinfo $molid set {a b c} {3.1507 3.1507 3.1507}

    # Replicate our water molecule in x,y,z 5x5x5 using topotools'
    # replicatemol command.
    set newmol [::TopoTools::replicatemol $molid 5 5 5]
```

Example:
VMD
Water box

3. Replicate waters to make box

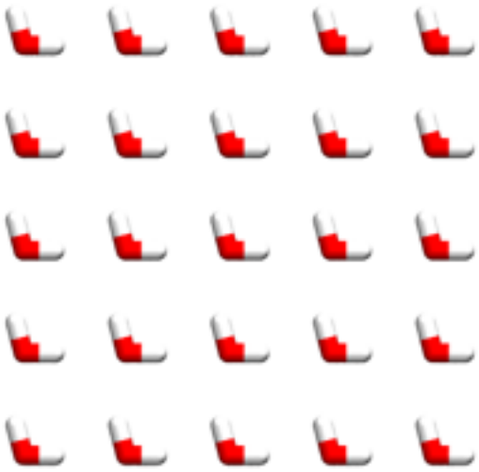


waterbox/waterbox.1.tcl

```
    #Let's set our bonds and angles using
    # topotools, using the 'topo addbond', 'topo addangle' commands.

    # Make a selection of all {O H1} atoms, and retrieve their indices
    # (required by addbond/addangle command).
    set sel [atomselect $newmol "name OH H1 H2"]

    # Get our list of indices, they are returned as
    # a list of n atom indices in the order in which
    # they were loaded into VMD:
    # { idx_O1 idx_H11 idx_H21\
        idx_O2 idx_H12 idx_H22\ ...
        idx_On idx_H1n  idx_H2n}
     set ids [$sel get index]
   # Loop over tuples of atom indices and add bonds and angles
     foreach {id1 id2 id3} $ids {
         topo addbond  $id1 $id2;  # Add bond between O-H1
         topo addbond  $id1 $id3;  # Add bond between O-H2
         topo addangle $id2 $id1 $id3; # Add angle H1-O-H2
     }

     ## Update VMDs and internal data structures
     mol reanalyze $newmol
     topo -molid $newmol retypebonds
     topo -molid $newmol retypeangles


}
```

Example: VMD Water box

4. Apply bonding and angle topologies



waterbox/waterbox.1.tcl

```
# write out a protein structure file (topology), pdb (coordinates)
 # and a lammps data file.
 set fname tip3_125
 animate write pdb $fname\.pdb
 animate write psf $fname\.psf
 topo -molid $newmol writelammpsdata $fname\.data full

 # Delete our selection
 $sel delete

 # Return our new waterbox
 return $newmol
```

Example:
VMD
Water box

5. Write out input files

```
375 atoms
250 bonds
125 angles
0 dihedrals
0 impropers
2 atom types
1 bond types
1 angle types
0 dihedral types
0 improper types
-7.547650 8.205850  xlo xhi
-7.562936 8.190564  ylo yhi
-7.876750 7.876750  zlo zhi

Pair Coeffs
##
# 1  HT
# 2  OT

# Bond Coeffs
#
# 1  HT-OT

# Angle Coeffs
#
# 1  HT-OT-HT

Masses

 1 1.008000 # HT
 2 15.999400 # OT

Atoms

1 1 2 -0.830000  x y z # OT UNK
2 1 1  0.415000  x y z # HT UNK
3 1 1  0.415000  x y z # HT UNK
```

waterbox/tip3_125.param

Example:
VMD
Water box

```
pair_style        lj/cut/coul/long 10.0 8.0
bond_style        harmonic
angle_style       harmonic

#Masses
mass 1 1.008000  # HT
mass 2 15.999400 # OT

pair_coeff        1  1    0.0460 0.4000 #HT-HT
pair_coeff        1  2    0.0836 1.7753 #HT-OT
pair_coeff        2  2    0.1521 3.1507 #OT-OT

bond_coeff        1     450.0 0.9572 # HT-OT

angle_coeff       1     55.0  104.52 # HT-OT-HT
```

Example: VMD Water box

waterbox/tip3_125.in

```
# Waterbox
echo both

neighbor        2.5 bin
neigh_modify    every 2 delay 0 check yes

units           real

atom_style      full
read_data       tip3_opt.data
include         tip3_125.param

kspace_style    pppm 1.0e-6

minimize        1.0e-4 1.0e-6 300 1000

velocity        all create 310.0 12345 mom yes rot yes dist gaussian
fix             mynpt all npt temp 310.0 310.0 400.0 iso 1.0 1.0 1000.0

dump            d1 all dcd 10 tip3_125.eq.dcd
dump_modify     d1 unwrap yes

timestep         0.5
thermo           10
run              1000
```

```
lmp_g++ < tip3_125.in
```

Your turn!

- Visualize the output from the water box simulation Can you make a movie?

- There are two other water box scripts provided, waterbox.3.tcl and waterbox.4.tcl. Try running them and examining their output. What do you think they are doing?

- Axel has two very nice TopoTools Tutorials for creating topologies for various molecules. Companion files are in 03/ and start at 05.tcl, in.05