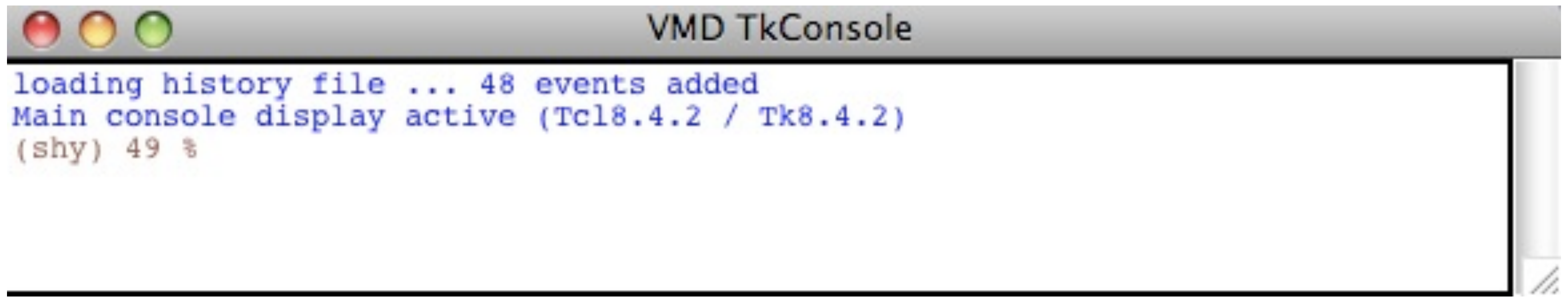
Macromolecule analysis with vmd/Tcl

Shy Arkin
arkin@huji.ac.il

vmd and TCl

- One of the most powerful features in vmd is its Tcl scripting.
- vmd can be run in command line mode only without any graphics.
- Use the TkConsole in the extensions menu.



```
loading history file ... 48 events added
Main console display active (Tcl8.4.2 / Tk8.4.2)
(shy) 49 %
```

Running a script

- You can “source” a script from the command line:
```
source my_script.tcl
```
- You can do this while running vmd in text mode without graphics and pass arguments:
```
vmd -dispdev text -e my_script.tcl -args 0.2
```
- You can type Tcl commands in the TkConsole.

A short Tcl primer

- Defining a variable (no need to define type):
```
set x 10
set text "some text"
```
- Printing a variable
```
puts "the value of x is: $x"
%> the value of x is: 10
puts "the value of text is: $text."
%> the value of text is: some text.
```
- Put a “;” at the end of the line to suppress verbosity. or to put comments (with #).

Variable

- Variables have $ in front of them.
- However, when defining or changing do not use the $ sign:

```
set x 5
incr x
puts $x
```

- Variables are local, but can be imported into subroutines.

Math

- Simple math requires the `expr` command:
  ```
  expr 3 - 8
  set x 10
  expr - 3 * $x
  ```
- Don’t forget that:
  ```
  expr 1 / 4
  ```
  will yield 0, but
  ```
  expr 1 / 4.0
  ```
  will yield 0.25
- So I always use numbers with .0 after them.

Expressions

- Any subroutine/function/procedure can be embedded with brackets, []:

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

- Or directly:

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

Variable and list

- Every variable can be directly treated as a list with operators that stat with “l”:

```
set letters "a b c d"
puts [lindex $letters 2]
%> c
```

- The index is zero based.

conditionals

```
if {$a > 7} {
  puts “$a is bigger than 7”
} elseif {$a == 7} {
  puts “$a is equal to 7”
} else {
  puts “$a is smaller than 7”
}
```

Writing / reading to / from a file

-
  ```
  set out [open "result.txt" w]
  puts $out “my data”
  ```

-
  ```
  set in [open “results.txt” r]
  set text [gets $in]
  ```
- Don’t forget to close them if you want to see the files while running:

  ```
  close $out
  ```

booleans

- | | means or
- && means and
- ! means not
- You can use parenthesis to define things specifically.

Loops

```
for {set i 0} {$i <= 3} {incr i} {
  puts $i
}

%>0
%>1
%>2
%>3
```

Procedures

- A procedure or proc (subroutine) is a way to reuse code:

```
proc add_number {input} {
  return [expr $input + 1]
}
puts [add_number 3]
%> 4
```

- The proc should be defined before it is read.
- Variables are local.

animate	Play/Pause/Rewind a molecular trajectory.
atomselect	Create atom selection objects for analysis.
axes	Position a set of XYZ axes on the screen.
color	Change the color assigned to molecules, or edit the colormap.
colorinfo	(Tcl) Obtain color properties for various objects
display	Change various aspects of the graphical display window.
exit, quit	Quit VMD.
gettimestep	Retrieve a timestep as a binary Tcl array (use for plugins)
help	Display an on-line help file with an HTML viewer.
imd	Control the connection to a remote simulation.
label	Turn on/off labels for atoms, bonds, angles, dihedral angles, or springs.
light	Control the light sources used to illuminate graphical objects.
logfile	Turn on/off logging a VMD session to a file or the console.
material	Create new material definitions and modify their settings.
measure	Measure properties of moleculear structures.

menu	Control or query the on-screen GUI menu forms.
molecule or mol	Load, modify, or delete a molecule.
molinfo	Get information about a molecule or loaded file.
mouse	Change the current state (mode) of the mouse.
play	Start executing text commands from a specified file.
render	Output the currently displayed image (scene) to a file.
rock	Rotate the current scene continually at a specified rate.
rotate	Rotate the current scene around a given axis by a certain angle.
scale	Scale the current scene up or down.
stage	Position a checkerboard stage on the screen.
tool	Initialize and control external spatial tracking devices.
translate	Translate the objects in the current scene.
user	Add new keyboard commands.
vmdinfo	(Tcl) Get information about this version of VMD
volmap	Create volumetric data based on molecular information
wait	Wait a number of seconds before reading another command. Animation continues.
sleep	Sleep a number of seconds before reading another command. Animation is frozen.

Selections

- Atom selection is a popular example:
```
set a [atomselect top “name CA”]
set b [atomselect 7 “name C” frame 6]
```
- You can then use these selections for many, many things.
- For example to figure out how many atoms are in your selection:
```
$sel num
```

Selections cont.

- `$sel list`
  list the indices of all atoms in the selection
- `$sel get {x y z}`
  yields the x, y and z coordinates.
- `$sel set charge [$sel get {beta}]`
  sets the charge of every atom in the selection according to the its beta value.

all	*bool*	everything
none	*bool*	nothing
name	*str*	atom name
type	*str*	atom type
index	*num*	the atom number, starting at 0
serial	*num*	the atom number, starting at 1
atomicnumber	*num*	atomic number (0 if undefined)
element	*str*	atomic element symbol string ('X' if undefined)
altloc	*str*	alternate location/conformation identifier
chain	*str*	the one-character chain identifier
residue	*num*	a set of connected atoms with the same residue number
protein	*bool*	a residue with atoms named `C, N, CA,` and `O`
nucleic	*bool*	a residue with atoms named `P, O1P, O2P` and either `O3', C3', C4', C5', O5'` or `O3*, C3*, C4*, C5*, O5*`. This definition assumes that the base is phosphorylated, an assumption which will be corrected in the future.
backbone	*bool*	the `C`, `N`, `CA`, and `O` atoms of a protein and the equivalent atoms in a nucleic acid.
sidechain	*bool*	non-backbone atoms and bonds
water, waters	*bool*	all atoms with the resname `H2O, HHO, OHH, HOH, OH2, SOL, WAT, TIP, TIP2, TIP3` or `TIP4`
fragment	*num*	a set of connected residues
pfrag	*num*	a set of connected protein residues
nfrag	*num*	a set of connected nucleic residues
sequence	*str*	a sequence given by one letter names
numbonds	*num*	number of bonds
resname	*str*	residue name

resid	*num*	residue id
segname	*str*	segment name
x, y, z	*float*	x, y, or z coordinates
radius	*float*	atomic radius
mass	*float*	atomic mass
charge	*float*	atomic charge
beta	*float*	temperature factor
occupancy	*float*	occupancy
user	*float*	time-varying user-specified value
at	*bool*	residues named `ADA A THY T`
acidic	*bool*	residues named `ASP GLU`
acyclic	*bool*	"protein and not cyclic"
aliphatic	*bool*	residues named `ALA GLY ILE LEU VAL`
alpha	*bool*	atom's residue is an alpha helix
amino	*bool*	a residue with atoms named `C, N, CA`, and `O`
aromatic	*bool*	residues named `HIS PHE TRP TYR`
basic	*bool*	residues named `ARG HIS LYS`
bonded	*bool*	atoms for which numbonds > 0
buried	*bool*	residues named `ALA LEU VAL ILE PHE CYS MET TRP`
cg	*bool*	residues named `CYT C GUA G`
charged	*bool*	"basic or acidic"
cyclic	*bool*	residues named `HIS PHE PRO TRP TYR`

Keyword	Arg	Description
hetero	*bool*	"not (protein or nucleic)"
hydrogen	*bool*	name "[0-9]?H.*"
large	*bool*	"protein and not (small or medium)"
medium	*bool*	residues named `VAL THR ASP ASN PRO CYS ASX PCA HYP`
neutral	*bool*	residues named `VAL PHE GLN TYR HIS CYS MET TRP ASX GLX PCA HYP`
polar	*bool*	"protein and not hydrophobic"
purine	*bool*	residues named `ADE A GUA G`
pyrimidine	*bool*	residues named `CYT C THY T URI U`
small	*bool*	residues named `ALA GLY SER`
surface	*bool*	"protein and not buried"
rasmol	*str*	translates Rasmol selection string to VMD
alpha_helix	*bool*	atom's residue is in an alpha helix
pi_helix	*bool*	atom's residue is in a pi helix
helix_3_10	*bool*	atom's residue is in a 3-10 helix
helix	*bool*	atom's residue is in an alpha or pi or 3-10 helix
extended_beta	*bool*	atom's residue is a beta sheet
bridge_beta	*bool*	atom's residue is a beta sheet
sheet	*bool*	atom's residue is a beta sheet
turn	*bool*	atom's residue is in a turn conformation
coil	*bool*	atom's residue is in a coil conformation
structure	*str*	single letter name for the secondary structure
phi, psi	*float*	backbone conformational angles
within	*str*	selects atoms within a specified distance of a selection (i.e `within 5 of name FE`).
exwithin	*str*	exclusive within, equivalent to `(within 3 of X) and not X`.
same	*str*	selects atoms which have the same keyword as the atoms in a given selection (i.e. `same segname as resid 35`)
ufx, ufy, ufz	*num*	force to apply in the x, y, or z coordinates

Selection measurements

- `measure center $sel`
  yields the coordinates (as a list) of the geometric center of the selection.
- `measure minmax $sel`
  yields a list of 2 lists, that hold the coordinates of the bottom left and top right of the selection.

Vectors and Matrices

- vmd has a ton of predefined routines to deal with vectors and matrices.
- For example:

```
set vec_a {0 0 1}
puts [vecinvert $vec_a]
%> 0 0 -1

set vec_b {0 0 2}
puts [vecadd $vec_a $vec_b]
%> 0 0 3
```

more vector examples

- `vecadd`
- `vecsub`
- `vecmean`
- `vecstddev`
- `vecscale`
- `vecdot`
- `veccross`
- `veclength`
- `vecnorm`
- `vecdist`
- `vecinvert`

Moving

- There are three main ways to move:
    - `moveby` - moves each of the atoms in the selection over by the given vector offset.
    - `moveto` - moves all the atoms in a selection to a given coordinate.
    - `move` - applies a given transformation matrix to each of the atom coordinates.

Moving examples 1

- The move your selection such that its center is at 0 0 0:
```
set all [atomselect top "all"]
$all moveby [vecinvert [measure \\
center $all weight mass]]
```
- To rotate a molecule around the x axis by 30 degrees:
```
set all [atomselect top "all"]
set matrix [transaxis x 30 deg]
$all move $matrix
```

Moving examples 2

- To align two proteins (that must have the same number of atoms):

```
set prot_a [atomselect 0 "all"]
set prot_b [atomselect 1 "all"]
set matrix [measure fit $prot_a $prot_b]
$prot_a move $matrix
```

- You can then measure the rmsd:

```
puts [measure rmsd $prot_a $prot_b]
```

- And then why not move it back:

```
$prot_a move [measure inverse $matrix]
```

Making a proc of it all

```
proc rmsd {a b} {
  if {[expr [$a num] - [$b num]] == 0} {
    set matrix [measure fit $a $b];
    $a move $matrix;
    set result [measure rmsd $a $b];
    $a move [measure inverse $matrix];
    return $result;
  } else {
    puts stderr "different number of atoms\n";
    return 0;
  }
}
```

Rotating a helix 1

```
# Define some selections
set a [atomselect top "resid 12  to 31  and chain A"]
set b [atomselect top "resid 59  to 85  and chain A"]
set c [atomselect top "resid 152 to 174 and chain A"]
set d [atomselect top "resid 181 to 200 and chain A"]
set e [atomselect top "resid 205 to 219 and chain A"]
set f [atomselect top "resid 223 to 236 and chain A"]
```

Rotating a helix 2

```
# the procedure
proc rotate_helix {angle sel average buffer} {
  # Rotate a selection about its axis by $angle degrees (in deg)
  # The helix axis is determined by averaging a number of Ca atoms
  # given by $average without $buffer residues from either ends
  set CAs [atomselect top "name CA and [$sel text]"];
  # find the range of residues
  set first_resid [lindex [$CAs get {resid}] 0];
  set last_resid  [lindex [$CAs get {resid}] end];
  # calculate the helix axis
  set a [expr $first_resid + $buffer]
  set b [expr $first_resid + $buffer + $average]
  set start [atomselect top "name CA and resid $a to $b"];
  set a [expr $last_resid - $buffer - $average]
  set b [expr $last_resid - $buffer]
  set end  [atomselect top "name CA and resid $a to $b"];
  set start_coords [measure center $start];
  set end_coords   [measure center $end  ];
  set matrix [trans bond $start_coords $end_coords $angle deg]
  $sel move $matrix
}
```

Rotating a helix 3

```
# then enjoy watching it rotate...
for {set i 0} {$i <= 360} {incr i} {
  rotate_helix 1 $a 4 2
  rotate_helix 1 $b 4 2
  rotate_helix 1 $c 4 2
  rotate_helix 1 $d 4 2
  rotate_helix 1 $e 4 2
  rotate_helix 1 $f 4 2
  display update
}
```

Analysis examples

- Backbone dependent rotamer library.
- H-bonding network

BB dependent Rotamers

```
source utilities.tcl
# setting the rotamer angles for Valine
set val {"N" "CA" "CB" "CG2"};
# getting the list of all valines
set valines [atomselect top "resname VAL and name CA"]
foreach i [$valines get {resid}] {
  set a1 [atomselect top "resid $i and name [lindex $val 0]"]
  set a2 [atomselect top "resid $i and name [lindex $val 1]"]
  set a3 [atomselect top "resid $i and name [lindex $val 2]"]
  set a4 [atomselect top "resid $i and name [lindex $val 3]"]
  set dihedral [dihedral_selection $a1 $a2 $a3 $a4]
  puts "[$a1 get {structure}]\t$dihedral"
}
```

Now run on the entire PDB

```
# run as: vmd -dispdev text -e rotamer_ss_pdb.tcl
source utilities.tcl
set val {"N" "CA" "CB" "CG2"};
set out [open "result.dat" w]
set in [open "pdb_list_short" r];
while {[gets $in line] > -1} {
  set pdb [lindex $line 0];
  set chain [lindex $line 1];
  puts stderr $pdb;
  mol delete all;
  mol pdbload $pdb;
  set valines [atomselect top "resname VAL and name CA and chain $chain"]
  foreach i [$valines get {resid}] {
    set a1 [atomselect top "resid $i and chain $chain and name [lindex $val 0]"]
    set a2 [atomselect top "resid $i and chain $chain and name [lindex $val 1]"]
    set a3 [atomselect top "resid $i and chain $chain and name [lindex $val 2]"]
    set a4 [atomselect top "resid $i and chain $chain and name [lindex $val 3]"]
    set dihedral [dihedral_selection $a1 $a2 $a3 $a4]
    puts $out "$pdb $i [$a1 get {structure}] $dihedral $chain"
  }
}
exit;
```

H-bonds 1

```
set number_of_cycles 8
set d 3.5
# find the center
set all [atomselect top all]
set center [measure center $all]
set x [lindex $center 0]
set y [lindex $center 1]
set z [lindex $center 2]
set start [atomselect top "type \"O.*\" \"N.*\" and \\
           (abs(x-$x)<3 and abs(y-$y)<3 and abs(z-$z)<3)"]
puts [$start_atoms list]
# set the initial lists
set now [$start list];
set tot $current
set i 0
while {$i < $number_of_cycles} {
  set front [atomselect top "(type \"O.*\" \"N.*\" and within \\
             $d of (index $now)) and not (index $tot $now)"]
  set tot [concat $tot $now];
  set now [$front list];
  incr i;
  if {$now == 0} {
    break;
  }
}
```

H-bonds

```
puts $tot;
# get the residues
set all [atomselect top "same residue as (index $tot)"]
set all [$all list]
# this updates the selection
mol representation DynamicBonds $cutoff 0.2 20.0
mol color ColorID 8
mol selection index $tot
mol material Opaque
mol addrep top
mol selupdate 1 top 0
mol representation Licorice 0.3 20.0 20.0
mol color Type
mol selection index $all
mol material Opaque
mol addrep top
mol selupdate 1 top 0
```

Area per lipid

- Generate random x and y values that are within the size of your system.
- Check to see if they are closer to the protein of lipid, and tabulate the ratio that of those that are closer to the protein and those that are closer to the lipid
- Multiplying the above ratio by the area and dividing by the number of lipids in that leaflet yields the area per lipid.

Movies

```
proc make_movie {step} {
  # set the number of frames in the movie
  for {set i 0} {$i < 360} {incr i $step} {
      # here you do what you want to the molecule
      # such as rotation, translation, etc.
      display update
      # take the picture
      set filename rotation.[format "%04d" $i].pov
      render POV3 $filename povray +H700 +W700 -I%s \\
                  -O%s.tga +D +X +A +FT Display=off
   }
}
```

Trajectories

- It is very easy to read in a trajectory:

```
animate read xtc data.xtc
animate read xtc data.xtc beg 3 end 700 skip 10 waitfor all
```

- You can set the beginning, end, skip and if you want to read it without updating.
- You can run any analysis on every frame by simply update the selections (next page).

Selection update

- A simple way is to update the selection:
```
$sel update
```
- This is very slow since the selection is redefined from scratch.
- If the selection does not depend on distances, then it is much faster to:
```
$sel frame 106
```

Large trajectories

- If the trajectory is very large and does not fit on the RAM of your PC then you can load one frame at a time.
- This is achieved with the specials smodule.
- This will make the code live for a long time as well.