Quick Reference Guide for Intermediate PyMOL Users

Supplemental information intended for users who are already comfortable using PyMOL for basic molecular visualization

Advanced User Interface Controls

Sequence Viewer Actions

Left-click-and-drag to toggle selection membership.
Shift-left-click to extend the existing selection.
Ctrl-shift-left-click to extend the existing selection while also centering upon it.

Middle-click to center on the clicked entry.
Ctrl-middle-click to zoom the clicked entry.
Ctrl-shift-middle-click-and-drag to zoom a boxed region.

Right-click on selected sequence for the selection menu.
Right-click on unselected sequence for the sequence menu.

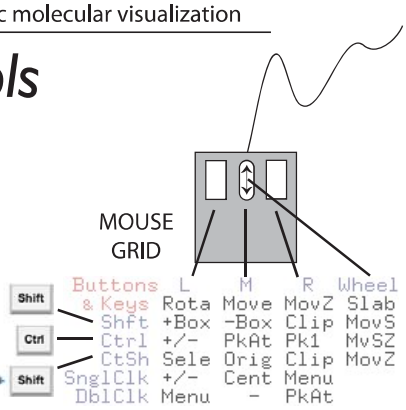
Names List Panel Actions

Left-click to toggle individual entries one at a time.
Left-click-and-drag to toggle multiple entries.
Ctrl-left-click-and-drag to browse (automatically activate & deactivate) entries one by one.

Middle-click to activate and center.

Right-click-and-drag to move entries up /down.

Use the "**group**" command to a create group, then:
Left click on [+] to open group.
Left-click on [-] to close group.

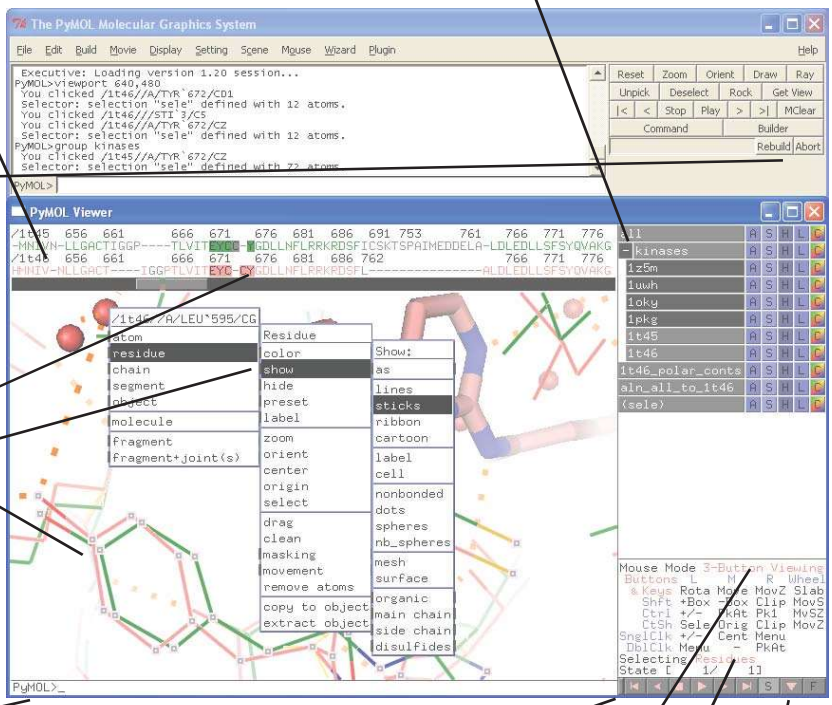


Mouse Wheel Codes

`Slab` : Adjust the depth of the visible slab (between the clipping planes).
`MovS` : Move the visible slab.
`MvSZ` : Move view center relative to the slab.
`MovZ` : Move the camera along the Z axis (zoom).

The "Abort" button can interrupt certain long-running tasks (ray tracing, surface calculation, etc.).

The "Rebuild" button restores consistency afterwards.



Context-specific menus can be used to apply actions to individual atoms, whole residues, chains, segments, objects, molecules, fragments, and the active atom selection ALL with a single menu-click-browse-and-release sequence.

Mouse Button Codes for Viewing

`Cent` : Center the view on a given atom.
`Clip` : Move clipping planes using vertical (front) or horizontal (back) motion.
`Menu` : Activate context-dependent menu.
`Move` : Translate camera in the XY plane (of the screen).
`MovZ` : Move the camera along the Z axis (zoom).
`Rota` : Free camera rotation.
`Pk1` : Pick one atom within the molecular graph.
`PkAt` : Pick atoms within the molecular graph.
`Sele` : Set the active selection.
`Orig` : Set the origin (atom) for camera rotation.
`+/-` : Toggle atom membership within the active selection.
`+Box` : Add atoms to the active selection using a box.
`-Box` : Remove atoms from selection using a box.

Command Line & Keyboard Actions

[TAB] attempts to complete a given command or parameter. If completion is ambiguous, a list of possible completions is output to the console.

[Ctrl-A] moves the cursor to the beginning of the line.

[Ctrl-E] moves the cursor to the end of the line.

[Ctrl-K] deletes all characters after the cursor.

[Up-Arrow] & [Down-Arrow] browse the command history.

[Left-Arrow] & [Right-Arrow] move the cursor or changes the movie frame or state (if no text is present).

[ESC] switches between graphics and text in the 3D viewer window (most useful in full-screen mode).

[Page-Down] & [Page-Up] advance forward or backward through the defined Scenes.

[Ctrl-Page-Down] & [Ctrl-Page-Up] define a new Scene before or after the current scene.

This handle resizes the control panel width.

Clicking here switches between viewing and editing modes.

Clicking here changes the selection mode.

Corner Buttons

S Toggles the sequence viewer.
▼ Toggles camera rocking
F Toggles full-screen mode.

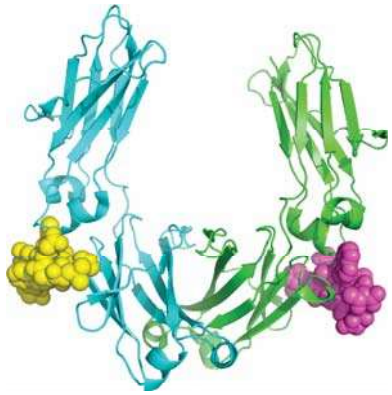
Mouse Button Codes for Editing

`DrgM` : Activates dragging for a discrete molecule.
`Rot_` : Free rotation.
`Mov_` : Move in XY plane.
`MvZ_` : Move along Z axis.
"_" above is A, D, F, or 0:
A for atom
D for dragged selection
F for fragment
0 for object
`PkTB` : Pick and (optionally) drag to change bond torsion.

Commands, Scripts, and Programs

Commands are called by keyword and typically accept comma-separated arguments.

```
PyMOL> color red, chain A
```



Commands can be aggregated into scripts (text files with a ".pml" file extension)

```
# load_and_render.pml
fetch 1dn2, async=0   ;# get PDB online; wait until done
as cartoon
show spheres, chain E+F
util.cbc  ;# color by chain
orient    ;# splay out along principal axes
unset opaque_background  ;# transparent fog, no background
png image.png, width=900, dpi=300, ray=1
```

The core commands you really need to know for everyday scripting with PyMOL are:

```
fetch pdb_code, name
load filename, name
show / as / hide representation, selection
color color, selection
orient / zoom selection
select name, selection
label selection, string-expression
distance / angle / dihedral name, selection1, ...
enable / disable name
ray / draw width, height
png filename, dpi=number
```

Often-used selection keywords and operators

```
polymer / organic / inorganic / solvent
name name-pattern
resi resi-pattern
resn resn-list
chain chain-list
byres / bymol / bychain selection
id / index / rank id-pattern
selection within distance of selection
selection extend bond-count
```

Nesting and set-wise operators: **(,), and, or, not, in, like**

Named selections save time by helping you refer back to specific atoms.

```
PyMOL> select binding_site, byres organic expand 5
PyMOL> show surface, binding_site
PyMOL> color cyan, binding_site and elem C
```

Selection Macros are a concise way of identifying atoms based on matching identifier fields separated by slashes. They come in two forms...

Fields in macros with a leading slash are matched left-to-right:
⏩ /object/segi/chain/resn`resi/name`alt

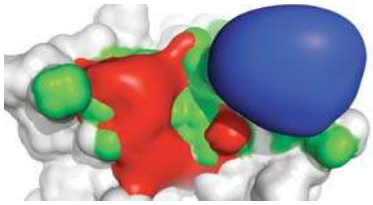
```
/1hpv              ;# /object
/1dn2/B/B/`271     ;# /object/segi/chain/`resi
/1t46//A/LEU/CA    ;# /object//chain/resn/name
```

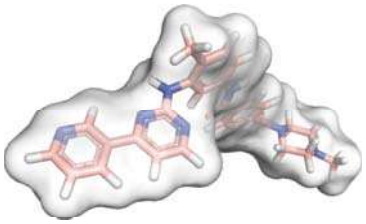
Fields in macros without a leading slash are matched right-to-left:
object/segi/chain/resn`resi/name`alt ⏪

```
132/*`A            ;# resi/name`alt
*/CA               ;# resi/name
145-130/N+CA+C     ;# resi/name
phe`106/           ;# resn`resi/
A/132-135/         ;# chain/resi/
B//                ;# chain//
1t46/A/A/his/      ;# object/segi/chain/resn/
```

Often-used settings, configured via: **set** *setting_name* **=** *setting_value* and **unset** *setting_name*

opaque_background = off (transparent) vs. **= on** (opaque - default)
transparency = 0.5 (for surfaces; also set sphere_transparency, cartoon.., stick..., etc.)
transparency_mode = 1 (multi-layer) vs. **= 2** (uni-layer - default)
ray_transparency_oblique = 1.5 (varies transparency based on surface normal)
surface_mode = 1 (surface all atoms) vs. **= 0** (do not surface HETATMs - default)
surface_quality = 1 (higher quality) vs. **= 0** (default)
surface_color = white (or other predefined color)





Commands scripts (.pml files) can include Python code three ways:

(#1) Implicitly, based on default Python fall-through handler:

```
load $PYMOL_PATH/test/dat/pept.pdb  ;# PyMOL
import os                           ;# Python
print os.environ['PYMOL_PATH']      ;# Python
```

PyMOL can also run Python (.py) programs directly. The PyMOL API is accessed via:

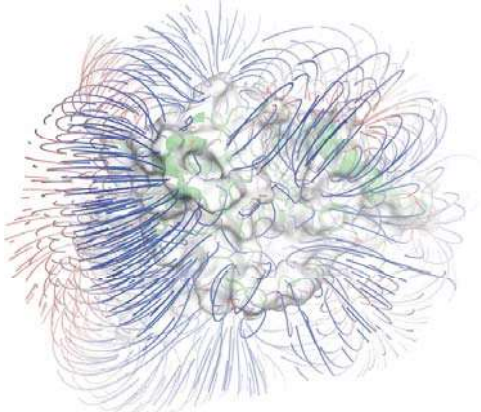
```
from pymol import cmd
cmd.method-name(arg1, arg2, ...)
```

(#2) explicitly on a single line with a leading forward slash ("/") (with optional "\" continuation lines):

```
/a=4; b=5; print a+b \
print a-b
```

or (#3) explicitly in the form of embedded Python blocks with standard block indentation.

```
# a simple XYZ file reader
python                                  ;# start of Python block
for line in open("example.xyz").readlines():
    field = line.strip().split()
    if len(field)==4:
        xyz = map(float, field[1:4])
        cmd.pseudoatom("my_example",elem=field[0], pos=xyz)
python end                              ;# end of Python block
show spheres, my_example
zoom
```



Movies and Animation

Besides static files, PyMOL can read multi-model files (.pdb, .xyz), multi-ligand files (.sdf), and molecular dynamics trajectories (.trj, .trr, .xtc)

```
PyMOL> load multimodel.pdb    PyMOL> load top.pdb, mov; load_traj traj.xtc,mov,start=1,stop=101,interval=10
```

Movie playback speed can be controlled with the movie_fps setting (frames per second) -- the default is 30 frames per second (TV-quality).

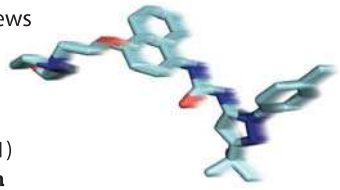
```
PyMOL> set movie_fps, 5
```

Movies are constructed from molecular states (changing atoms & coordinates) combined with frames (changing camera, object matrices, and commands)

```
PyMOL> load $PYMOL_PATH/test/dat/ligs3d.sdf; mset 1 x5 1 -10 10 x5 10 -1

  implies: FRAME: 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30 (movies loop
           STATE: 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,10,10,10,10,10,10, 9, 8, 7, 6, 5, 4, 3, 2, 1   by default)
```

Each frame can have a camera "waypoint" stored with it, and can be interpolated for in-between frames with stored views



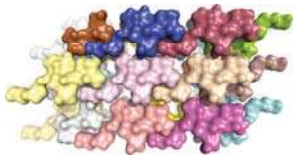
```
PyMOL> mview store,1; turn y,30; mview store,10; turn y,-30; mview store, 20
PyMOL> mview interpolate     ;# do interpolation for missing frames
PyMOL> mview reinterpolate   ;# redo interpolation for entire movie
```

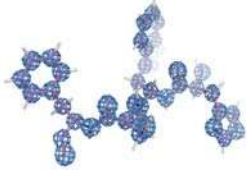
Objects can also be animated with mview using the "**object=**" optional argument (works best when matrix_mode = 1)

```
PyMOL> delete all; mset 1 x60; set matrix_mode=1; fragment lys; fragment tyr; zoom
PyMOL> frame 1; mview store,object=lys; frame 30; translate [5,0,0],object=lys; mview store,object=lys
PyMOL> mview interpolate,object=lys
PyMOL> frame 1; mview store,object=tyr; frame 45; rotate    x, 90,  object=tyr; mview store,object=tyr
PyMOL> mview interpolate,object=tyr
```

Movie commands can perform arbitrary actions on a per-frame basis (for a given frame, "**mdo**" resets existing commands , "**mappend**" doesn't).

```
PyMOL> mdo  1: show_as lines
PyMOL> mdo 30: show_as sticks
```





Also see: "Movie" menu program options, and "File" menu "Save As" MPEG or individual PNG files.

Symmetry Expansion & Electron Density Maps

Can read CCP4 maps, ASCII X-PLOR maps, binary O/BRIX/DSN6 maps, CUBE and GRD files (file extensions: .ccp4, .xplor, .omap, .brix, .dsn6, .cube., & .grd).

```
PyMOL> load ref.pdb
PyMOL> load 2fofc.ccp4
PyMOL> isomesh full_map, 2fofc, 1.0    ;# display density mesh for the entire map at 1.0 isosurface level
PyMOL> isomesh brick_region, 2fofc, 1.0, ref//C/26/, 3.0        ;# parallelepiped with 3.0 Å atom buffer
PyMOL> isomesh carve_region, 2fofc, 1.0, ref//C/26/, carve=2.1 ;# only show density within 2.1 Å of atoms
```

Symmetry expansion is accomplished using "symexp" command, which relies upon CRYST records present in the input PDB file.

```
PyMOL> symexp prefix, object, selection, cutoff   # example: PyMOL> symexp sym, ref, ref, 5
```

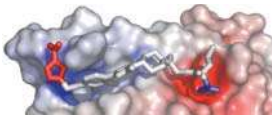
Fitting and Alignment Operations

Most protein structure alignments can be accomplished with the "align" command, which performs a sequence-based alignment followed by an iterative structure alignment. You specify atom selections contained within the mobile and target objects, and PyMOL will match them up.

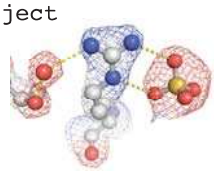
```
align mobile, target  ;# example: PyMOL> align 1t46////CA, 1t45////CA  ;# performs C-alpha-only alignment
```

Other types of alignment tasks can be performed with the following set of commmands:

```
super mobile, target       ;# structure-based alignment which refines just like "align"
pair_fit mobile, target    ;# atom-pair-based alignment, example: PyMOL> pair_fit obj1/2-7/CA, obj2/4-9/CA
fit mobile, target         ;# identifier-based alignment, where all atom identifiers match
intra_fit selection        ;# molecular-state-based alignment within a single molecular object
matrix_copy source_name, target_name    ;# copies matrix from one object to another
```

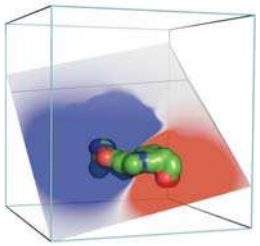


Electrostatic Calculations & Visualizations



Can read binary PHI and DX maps (.phi, .dx). PyMOL 1.2+ builds ship with APBS bundled. The APBS plugin can help to calculate electrostatic potentials around protein atoms. Once you have loaded the potential map, the following commands become useful:

```
isosurface name, map, level, selection, buffer
gradient name, map    # draws gradient field lines
slice_new name, map   # creates a slice plane through the map
```



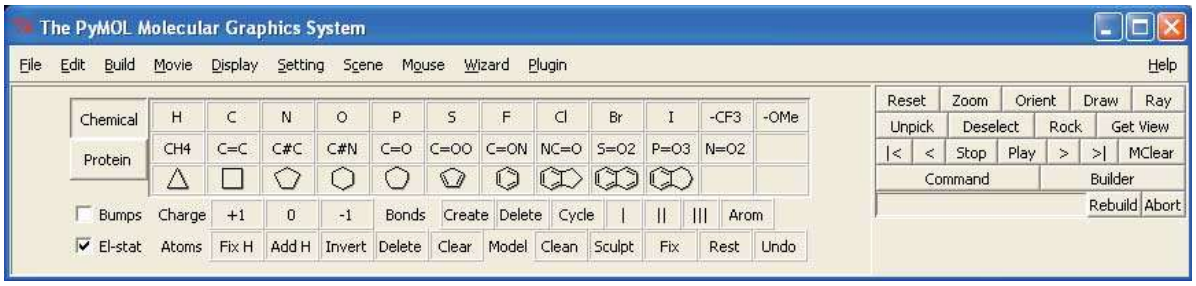
Coloring is achieved using "ramp" objects which can convert vertex coordinates into colors based on map values.

```
ramp_new name, map, range, color  # example: PyMOL> ramp_new ramp1, map1, [-3,0,-3]
                                   PyMOL> set surface_color, ramp1
                                   PyMOL> color ramp1, slice1
```

Modeling: Edit, Build, Cleanup, and Sculpt!

Point mutations are best made using the Mutagenesis wizard (see "Wizard" menu).

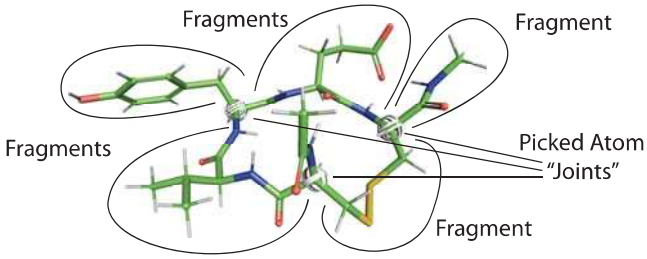
```
remove selection                                              ;# remove atoms from molecular object
create name, selection, source_state, target_state                  ;# create new object / state
fab input, name                          ;# fabricate peptide from sequence, example: PyMOL> fab APEPTIDE
fragment name, object                       ;# load fragment from library, example: PyMOL> fragment arg
pseudoatom object, selection, name, ...                    ;# create pseudoatom at center of selection
alter selection, expression                                  ;# change atom properties and identifiers
alter_state state, selection, expression     ;# change coordinate values using expression (e.g. x=x+1)
bond atom1, atom2, order                                  ;# create bond(s) between atoms in same object
unbond atom1, atom2                                       ;# remove all bonds between atom selections
valence order, selection1, selection2                  ;# adjust valences for bonds between selections
clean selection, present, state             ;# run MMFF minimization on selected atoms in single object
```



This panel is accessed by pressing the "Builder" button in the upper window.

To use the builder on Mac OS X, rename "MacPyMOL" to be "PyMOLX11Hybrid", and then relaunch (will run in a cross-platform compatibility mode).

In addition to providing traditional editing functions, such as bond torsioning (see the PkTB mouse action), PyMOL has a graph-based molecular editing capability which, combined with sculpting, allow for dynamic manipulation of connected molecular fragments delimited by picked atom "joints" (chosen using the PkAt mouse action). Each fragment can be rotated and translated and otherwise transformed, with interactive "molecular sculpting" or a subsequent MMFF forcefield-based conformational cleanup.



Scenes, Sessions, and Shows

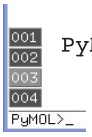
Scenes are like PowerPoint slides in that they each contain a distinct visualization intended to be part of a sequential presentation. To create a Scene, simply choose "Append" from the "Sene menu." Scenes can also include a text annotation displayed at the top of of the screen, as shown here:

```
PyMOL> scene new, store, This is my annotation text for a new scene.
PyMOL> scene auto, update, This is my updated annotation text for an existing scene.
```

Scenes are aggregated together into "Session" files (.pse) along with all of the required molecular content. Session files are completely self-contained and can be emailed or shared just like PowerPoint documents.

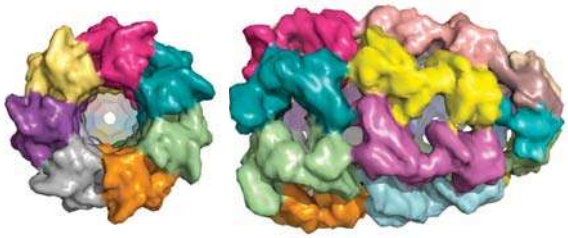
Show files are merely Session files with a (.psw) file extension that signals PyMOL to open the file in full-screen mode. You can convert a Session file to a Show file (and back) simply by changing the file extension. You can advance through a Show using [Space-Bar] or [Page-Up] / [Page-Down].

Show files can also be played inside of PowerPoint using the ActiveX Control (AxPyMOL), which is still under development. A FREE copy of the current (sometimes unstable) AxPyMOL build can be downloaded from `http://axpymol.org`




```
PyMOL> set scene_buttons ;# displays scene buttons in a "sorter".  Right-click-and-drag to reorder.
```

Creating Low Resolution Surfaces of Large Systems



```
PyMOL> fetch 1aon, struct, async=0     ;# load 58,870 atom structure
PyMOL> util.cbc                                      ;# color by chain
PyMOL> alter all, b=40                          ;# set temperature factors
PyMOL> set gaussian_resolution, 6               ;# set fake map resolution
PyMOL> map_new map, gaussian, 4, struct, 6  ;# create map from atoms
PyMOL> isosurface surf, map, 1                  ;# create surface from map
PyMOL> ramp_new ramp, struct, [0,6,6], [-1,-1,0]    ;# color by atom
PyMOL> set surface_color, ramp, surf           ;# set surface color
PyMOL> disable; enable surf                     ;# only show suface
```

Please Also Consult the Online Resources: http://pymolwiki.org http://pymol.org/archive http://delsci.info