PyMOL Handout
PSB-CIBB

Babu A. Manjasetty PhD
Cyril Dian PhD
MariaRosa Quintero Bernabeu PhD

1 Introduction to the software

PyMOL is a molecular viewer, render tool, and 3D molecular editor intended for visualization of 3D chemical structures including atomic resolution X-ray crystal structures of: proteins, nucleic acids (DNA, RNA, and tRNA), and carbohydrates, as well as small molecule structures of drug leads, inhibitors, metabolites, sugars, nucleoside phosphates, and other ligands including inorganic salts and solvent molecules. PyMOL is a USER-SPONSORED molecular visualization system on an OPEN-SOURCE foundation.

1.1 Visualization

To visualize a 3D structure the file has to be in the right format. The supported formats are:

.pml	PyMOL command script to be run on startup
.py, .pym, .pyc	Python program to be run on startup
.pdb	Protein Data Bank format file to be loaded on startup
.mmod	Macromodel format to be loaded on startup
.mol	MDL MOL file to be loaded on startup
.sdf	MDL SD file to be parsed and loaded on startup
.xplor	X-PLOR Map file (ASCII) to be loaded on startup
.ccp4	CCP4 map file (BINARY) to be loaded on startup
.cc1, .cc2	ChemDraw 3D cartesian coordinate file
.pkl	Pickled ChemPy Model (class “chempy.model.Indexe”)
.r3d	Raster3D file
.cex	CEX file (Metaphorics)
.top	AMBER topology file
.crd	AMBER coordinate file
.rst	AMBER restart file
.trj	AMBER trajectory
.pse	PyMOL session file
.phi	Delphi/Grasp Electrostatic Potential Map

Once your structure is loaded, you can modify it. The simplest actions are:

- close-up: zoom-in and out
- surface/map: The surface representation of a protein, in PyMol, shows the ”Connolly” surface or the surface that would be traced out by the surfaces of waters in contact with the protein at all possible positions.
- symmetry: generate symmetry objects of your molecule.
- align: align performs a sequence alignment followed by a structural alignment, and then carrys out zero or more cycles of refinement in order to reject structural outliers found during the fit.

- stereo: the stereo option controls whether or not PyMOL displays the scene in stereo mode. Stereo mode is a convenient way to see 3D. There are also various 3D modes

1.2 Input modes

Pymol supports two modes of input: point and click mode, and command line mode. It is also possible to work using scripts.

- Manually: the point and click allows you to quickly rotate the molecule(s) zoom in and out and change the clipping planes.
- command line: the command line mode where commands are entered into the external GUI window supports all of the commands in the point and click mode, but is more flexible and possibly useful for complex selection and command issuing. Commands entered on the command line are executed when you press the return key.
- Script: running a Python script or a Pymol command script from PyMOL, usually the command: run script.py or rum script.pml is enough. Of course, the file script.py(pml) needs to be in the working directory.

1.3 The Pymol interface

When PyMol is opened, two windows appear. The smaller window (called the "External GUI" in PyMol documentation) contains the menu bar (File, Edit, Help, Display, etc), shortcut buttons for common commands, and the command line. The second window is the PyMOL Viewer, which is where all the magic happens. In the Viewer, 3D models are displayed, and the user interacts (eg rotates) and manipulates the model.

- Pymol viewer (3D Screen): the objects that PyMOL renders in 3D are loaded from coordinate files that describe (in great detail) locations of individual atoms in the molecule. PyMOL can display more than one object at a time, and provides an Object Control Panel to adjust viewing modes, colors, labels, hiding, and just about anything else relating to objects.
- Object list: on the right-hand side of the Pymol viewer there is a list of all the objects that have been loaded. After each object name is a set of command buttons which control the object. Here are the buttons and some of their options:
  - A - Actions: Rename, duplicate, remove, apply presets (like "ball-and-stick" or "publication"), perform computations
  - S - Show: Change the way things appear, eg change to stick or cartoon view.
  - H - Hide: Things that are shown using S accumulate, and don't automatically replace the last view. H is the opposite of S and hides unwanted representations.

- L- Label: Label atoms, residues, etc.
- C - Color: Change the color of atoms and groups.

- Command line: the PyMol command line is a great tool that lets the experienced user change all sorts of options that simply don't appear in the point-and-click graphical interface. It can also be a lot faster. Combined with scripting, it is a powerful option for automating tasks and making intricate sets of changes. But, it's complex, and page upon page of PyMol documentation cover these commands, so we're going to ignore them as much as possible.

- Mouse controls

	L Rota	M Move	R MovZ	Wheel Slab
Shift	+Box	-Box	Clip	MovS
Ctrl	+/-	PkAt	Pk1	-
CtSh	Sele	Cent	Menu	-
DblClk	Menu	Cent	PkAt	-

- Main commands Here you will find the main commands for working with PyMOL. Please refer to the Pymol reference card and other literature for more details.

1. open: in the external GUI, File → open to open your file, that will be an object on Pymol.
2. load (command): reads several file formats. If an object is specified, then the file is loaded into that object. Otherwise, an object is created with the same name as the file prefix. usage: load filename [,object [,state [,format [,finish [,discrete [,multiplex]]]]]]
3. run: executes an external Python script in a local name space, the main Python namespace, the global PyMOL namespace, or in its own namespace (as a module). Usage: run python-script [, (local — global — module — main — private)]
4. set: is one of the most utilized commands. PyMOL representations, states, options, etc. are changed with set. Briefly, set changes one of the PyMOL state variables. Usage set name, [,value [,object-or-selection [,state]]]
5. select: mouse → selection mode or as a command. Creates a named selection from an atom selection. Selections are one of the most powerful aspects of PyMOL and learning to use selections well is paramount to quickly achieving your goals in PyMOL.
6. get (command): returns the value of a setting.
7. Help (command): typing help name of command on the command line, you get information about how to use that command. Also there is a general help menu on the external GUI.

8. Save: writes selected atoms to a file. File → save (different options). Also as a command, it is necessary to specify options. Usage save file [,(selection) [,state [,format]]]
9. Quit: close the program.
10. Reinitialize: reinitialize Pymol.

- Wizards and plugins: there are special Python scripts which work with PyMOL in order to obtain direct user interaction and easily perform complicated tasks. You can access them through the wizards menu on the External GUI.

  Plugins are external modules which extend Pymol's capabilities. Available plugins (if any) are shown in the Plugin menu on the External GUI. If no plugins are listed, then either none have been installed, or those that are installed are not yet functional.

- Rendering: this is a small plugin to render images with a given DPI. The "Ray" button raytraces, and the "Draw" button just draws the image without raytracing (a fast way to see that the height/width look good).(see 6)

2 Simple figure

- Cartoon: right-hand side of the Pymol viewer, changes the default cartoon for a set of atoms.

- Color: right-hand side of the Pymol viewer, changes the color of an object or an atom selection.

- Background/light/shadow/fog/clip: Lighting is important for high-quality shots. PyMOL supports of up to 10 virtual lights. You can turn the lights on/off and also position them where you want. Also it is possible to change the background color and the shadows in your figure. You can also chose the clipping planes with clip.

- Ray high resolution: creates a ray-traced image of the current frame. See 6

- Example script

```
load gank_0207.pdb,a
bg_color white
set cartoon_fancy_helices, 0
```

- Center: translates the window, the clipping slab, and the origin to a point centered within the atom selection.

- Zoom: scales and translates the window and the origin to cover the atom selection.

- Selection (around/expand ect.): creates a named selection from an atom selection. Selections are one of the most powerful aspects of PyMOL and learning to use selections well is paramount to quickly achieving your goals in PyMOL. Selections can be done mainly by typing in sele object-name/segi-id/chain-id/resi-id/name-id, or by selecting directly on the sequence.
- sequence: botton right-hand side of the Pymol viewer. Shows the sequence of your molecule.
- View: makes it possible to save and restore viewpoints on a given scene within a single session.
- Residues display: right-hand side of the Pymol viewer
- Non bonded: right-hand side of the Pymol viewer. Shows non-bonded atoms
- Coordination/hydrogen bonds: bond creates a new bond between two selections, each of which should contain one atom. You can easily create a new bond by selecting two atoms, each with the CTRL-MIDDLE-MOUSE-BUTTON and typing "bond" on the command line.
- Modification of any parameters (alter): alter changes one or more atomic properties over a selection using the python evaluator with a separate name space for each atom.
- Example script

```
load pdb, prot1
# zoom consistenly 20 Ang from each object at the center
center prot1
zoom center, 20
```

3 Symmetry/packing/align

- Symmetry: set_symmetry can be used to define or redefine the crystal and spacegroup parameters for a molecule or map object. "symexp" creates all symmetry-related objects for the specified object that occur within a cutoff about an atom selection.
- Align (pymol/coot/ccp4mg ect): performs a sequence alignment followed by a structural alignment, and then carrys out zero or more cycles of refinement in order to reject structural outliers found during the fit. For comparing proteins with lower sequence identity, an alignment program like, Cealign might be a better choice. Example script:

```
align  prot1////CA, prot2, object=alignment
```

- Selection: see 2
- Example script

```
load gank_0207.pdb,a
#symexp prefix, object, selection[, cutoff]
symexp sym=foo,(foo),5.0
delete sym*
```

4 NMR structures

NMR models should be loaded into the same object, but should have different states. Load a model into an object and then slplit it into different states:

```
set all_states, on
split_states name
```

It is possible to fit two structures, i.e. supperimpose them. Only matching atoms in both selections will be used for the fit.

```
fit (selection), (target-selection)
```

5 Maps and Surfaces

5.1 Loading and rendering electron density maps using ccp4 format

Use FFT to create a map in CCP4. (This task can be run from the Run FFT-Create Map option under Map and Mask Utilities). You can create a simple map or a 'Fo-Fc map as you wish. CCP4i will create a 2Fo-Fc map by default. To make a Fo-Fc map, set F1=DELFWT and PHIC=PHDELWT in the task window. Select the option to cover 'all atoms in PDB file'. For pymol to read the map later, you will need to add the file extension .map.ccp4 to the generated map.

1. Open pymol and read in your PDB file. Create a selection about whatever you want to see the map around. Typically one would display the selection as sticks.
2. Open your map in pymol, e.g. mymap.map.ccp4 (you need the ccp4 extension). An object named mymap.map will be created by pymol.

3. Identify a selection about which to display your map, e.g. Select site, resi X-Y and resn Z

4. To display the map around your selection (e.g., named site) issue a command similar to this:

```
isomesh map, mymap.map, 2.0, site, carve=1.6
```

   This command will create a mesh map object named map from the object mymap.map, contoured at 2.0 sigma, around the selection site, within 1.6Åof selected atoms. You can change the contour level and carve parameter to suit.

5. For a publication quality figure the following are suggestions:

```
color grey50, map # sets map to 50% gray
set mesh_width, 0.5 # makes meshes thinner for ray-tracing
bg_color white #sets background to white
set ray_trace_fog, 0 # turns off raytrace fog--optional
set depth_cue, 0 # turns off depth cueing--optional
set ray_shadows, off # turns off ray-tracing shadows
```

5.2 Surface

To calculate a surface of your protein (see also 1.1), it is always good to prepare a pdb without non bonded atoms (ligands, water, ions etc.)

Create full surface

```
load  "pdb", object name
hide all
show surface, "object name"
set surface_quality, 1
rebuild
```

Create partial surface

```
load "pdb", object name
hide all
set surface_quality, 1
rebuild
sel A, id 1-100
 create B, A
 show surface, B
```

Transparency

```
set transparency, 0.5  # between 0 to 1
```

5.3 Electrostatic surface using APBS: APBS-generated electrostatic surface displayed in PyMOL

APBS, the Adaptive Poisson-Boltzmann Solver, is a freely available macromolecular electrostatics calculation program. It is a cost-effective but uncompromised alternative to GRASP, and it can be used within pymol. Pymol can display the results of the calculations as an electrostatic potential molecular surface.

PyMol currently supports the APBS plugin. This plugin makes it possible to run APBS from within PyMOL, and then display the results as a color-coded electrostatic surface (units KbT / ec) in the molecular display window (as with the image to the right). See the APBS wiki for more details, including instructions on how to download, install and use the plugin.

In order to calculate an electrostatic potential molecular surface using APBS pymol plugin with your protein pdb, a modified version of your pdb is required with the .PQR extension.

Pdb2pqr software or server were designed to convert PDB-format structural information into PQR-format parameterized files. A PQR file is a popular and compact way to include atomic parameters in a PDB-like format by replacing the occupancy column of a PDB file (P) with the atomic charge (Q) and the temperature factor column with the radius (R).

- Preparation PQR

  There is a program that allows to convert pdb files to pqr format:
  http://www.poissonboltzmann.org/pdb2pqr/

  Before to run pdb2pqr with your pdb you need to make sur that:

  1. Your pdb should not contain any residues in several conformers (mainly structure at high resolution)
  2. Your pdb should contain complete residues information (missing lateral chain have to be incorporated)
  3. Bfactor of your pdb need to be <100
  4. Remove non bonded atoms (except specific metals)

  Nucleic acids may prove problematic for the apbs plugin. If so, use the pdb2pqr command-line tool to create a pqr file manually, instead of using the plugin to generate it. Then direct the APBS GUI on the main menu to read the pqr file you externally generated.

- How to use APBS plugin

  http://www.poissonboltzmann.org/apbs/examples/visualization/apbs-electrostatics-in-pymol

  Go to Plugin → APBS Tools to open the APBS calculation plugin.

  1. Under the "Main" tab of the PyMOL APBS Tools window, select Use another PQR and either browse to (via the Choose Externally Generated PQR: button) or input the path to your PQR file. This step is necessary to ensure you use the radii and charges assigned by PDB2PQR.
  2. Under the "APBS Location" tab of the PyMOL APBS Tools window, either browse to (via the APBS binary location: : button) or input the path to your local APBS binary. It is not necessary to provide a path to the APBS psize.py binary for most biomolecules.
  3. Under the "Temporary File Locations" tab of the PyMOL APBS Tools window, customize the locations of the various temporary files created during the run. This can be useful if you want to save the generated files for later use.
  4. Under the "Configuration" tab of the PyMOL APBS Tools window, hit the Set grid to set the grid spacings. The default values are usually sufficient for all but the most highly charged biomolecules.
  5. Under the "Configuration" tab of the PyMOL APBS Tools window, customize the remaining parameters; the defaults are usually OK.
  6. Under the "Configuration" tab of the PyMOL APBS Tools window, hit the Run APBS button to start the APBS calculation. Depending on the speed of your computer, this could take a few minutes. The Run APBS button will become unselected when the calculation is finished.

- Visualization of the surface potentials and Electrostatic isocontours

  – Surface potentials

    If you haven't already, hide the isocontours by hitting Positive Isosurface and Negative Isosurface and Hide buttons. The surface potential is also straightforward to visualize. Set the "Low" and "High" values to the desired values (usually ±1, ±5, or ±10 kT/e) at which the surface colors are clamped at red (-) or blue (+). Check the "Solvent accessible surface" and "Color by potential on sol. acc. surf." buttons to plot the potential on the solvent-accessible (probe-inflated or Lee-Richards) surface. Hit the "Molecular Surface" Show button to load the surface potential.

    In my opinion, the solvent-accessible surface tends to reveal more global features of the surface potential. Tighter surfaces (e.g., van der Waals and molecular or

Connolly surfaces) provide more information about the shape of the biomolecule but otherwise tend to simply map atomic surface charges onto the biomolecular surface. Thankfully, PyMOL provides an excellent solution to the conflicting need to obtain geometric information from the molecular surface together with useful electrostatic potential information from the solvent-accessible surface. To visualize the molecule in this way, simply uncheck the "Solvent accessible surface" box and check the "Color by potential on sol. acc. surf." box on the "Visualization tab".

– Electrostatic isocontours

PyMOL makes this step very easy: adjust the positive and negative "Contour" fields to the desired values (usually ±1, ±5, or ±10 kT/e) and hit the Positive Isosurface and Negative Isosurface and Show buttons.

6 Ray traced images for publication

6.1 Important Settings

These can be changed using the "set" command. Unless otherwise specified, the settings apply only to the ray-tracing engine and not the OpenGL renderer. Some reconciliation between the two renderers is much needed, so be warned that these settings may change in the future.

Normally, the only settings you will need to change are orthoscopic, antialias, and gamma. If you are down in an enzyme active site which is heavily shadowed, you may want to increase direct to 0.5-0.7 in order to improve brightness and contrast.

- orthoscopic = (0 or 1): controls whether the OpenGL renderer uses the same orthoscopic transformation as the renderer. You'll want to set this to 1 when preparing figures so that OpenGL and raytracing match pixel-for-pixel. controls the relative ambient intensity between OpenGL and the ray-tracer.
- antialias =(0 or 1): generates a "smooth" image (best quality, but takes 4X as long).
- spec_reflect, (0.0-1.0): intensity of the specular reflection from the light.
- ray_shadows =(0 or 1): turn /onoff shadows
- ray_trace_fog = (0 or 1); turn on/off fog
- depth_cue= (0.0-1.0) Fog depth
- direct (0.0-1.0): the planer light intesity originating from the camera
- reflect (0.0-1.0): the planer light intesity originating from the light source

- spec_power (1-100): how I crank-up the glossyness of rendered atoms
- ambient (0.0-1.0): controls the ambient light intensity for both OpenGL and the ray-tracer.
- gamma (0.1-2.0) gamma transformation applied after rendering is complete.

6.2 Resolution/quality

Ray tracing mode

```
# normal color
set ray_trace_mode, 0

# normal color + black outline
set ray_trace_mode,  1

# black outline only
set ray_trace_mode,  2

# quantized color + black outline
set ray_trace_mode,  3

set ray_trace_mode, 1 # (or 2 or 3; best with "bg_color white;set antialias,2")
# These two new modes -- 2 and 3 -- are cool cartoon looking modes.
```

Publication quality figures

To render a figure with the default resolution (640x480), use the ray command or the Ray button on the GUI window. You will be able to preview the low resolution figures on screen

To render a figure with a higher resolution you should use a number which is a multiple of 1024:

```
ray 1204, 1204
ray 1280, 1024
ray 2048, 2048
ray 4096, 4096
```

6.3 Saving images

All images (ray-traced or not) can be saved in PNG format using the "png" command. This format is directly readable by PowerPoint, and can be easily converted into other formats using a package like ImageMagick. You can also save images using the "Save Image" option in the "File" menu. Images are always saved at the same resolution as the viewer window.

```
ray
png my_image.png
```

Code	Name	Aspect ratio	Width	Height
VGA	Video Graphics Array	4:3	640	480
SVGA	Super Video Graphics Array	4:3	800	600
XGA	eXtended Graphics Array	4:3	1024	768
XGA+	eXtended Graphics Array Plus	4:3	1152	864
WXGA	Widescreen eXtended Graphics Array	5:3	1280	768
WXGA	Widescreen eXtended Graphics Array	8:5 (16:10)	1280	800
SXGA	Super eXtended Graphics Array	4:3	1280	960
SXGA	Super eXtended Graphics Array	5:4	1280	1024
HD	High Definition (Basic)	16:9	1366	768
WSXGA	Widescreen Super eXtended Graphics Array	8:5 (16:10)	1440	900
HD+	High Definition (Plus)	16:9	1600	900
UXGA	Ultra eXtended Graphics Array	4:3	1600	1200
WSXGA+	Widescreen Super eXtended Graphics Array Plus	8:5 (16:10)	1680	1050
HD-1080	Full High Definition	16:9	1920	1080
WUXGA	Widescreen Ultra eXtended Graphics Array	8:5 (16:10)	1920	1200

Figure 1: Resolution for the different image formats.

7 Movie/morphing

Simple movie can be created with pymol. What you need is to write a script generating several frames of your protein in different view and generate the corresponding images. The resulting images can be then combined to creat the movie using several free-software.

- First step: you need to define what to show (rotation, translation, zoom ect) and how long you want to show it (how many frames you need to show it)
- Second step: a movie can be made using either the gui (new version of pymol, 1.2 and 1.3) of pymol or a script.
- Third step: making the movie

The newest version of pymol using the gui see: http://www.pymolwiki.org/index.php/MovieSchool_1

1. Single rotation

   The basic idea is to pre-define two views linked by a rotation step, then generate all images between the two views.

```
#Load  movie.pdb

Set the speed of the movie frame per second (FPS)

Select the initial view you want using the cmd get_view


set_view (\
    -0.080400191,   -0.252943367,    0.964139223,\
    -0.889644265,    0.454434782,    0.045032199,\
    -0.449527502,   -0.854115129,   -0.261566907,\
    -0.000249837,    0.000085980, -190.179809570,\
    14.270303726,   49.188472748,    0.545298576,\
   149.940216064,  230.421966553,    0.000000000 )

Initializing the movie using mset

Definition:
mset 1         // simplest case, one state -> one frame
mset 1 x10     // ten frames, all corresponding to state 1
mset 1 x30 1 -15 15 x30 15 -1
// more realistic example:
// the first thirty frames are state 1
// the next 15 frames pass through states 1-15
// the next 30 frames are of state 15
// the next 15 frames iterate back to state 1

mset 1 x180 ( only one state using 480 frames)

# First movie frame
frame 1

# now store this view at frame 1

mview store
(mview store,object=object name, if you have several objects)


# Final movie frame:  frame 180 (turn y, -180) (6 sec)
frame 180
```

```
Using a cmd line tape turn y, 180 followed by get_view

set_view (\
     0.080400273,   -0.236078084,   -0.968406916,\
     0.889644265,    0.455151498,   -0.037094191,\
     0.449527472,   -0.858550072,    0.246620566,\
    -0.000249837,    0.000085980, -190.179809570,\
    14.270303726,   49.188472748,    0.545298576,\
   149.940216064,  230.421966553,    0.000000000 )

mview store
(mview store,object=object name)
mview reinterpolate
(mview reinterpolate, object=object name)

# mview reinterpolate
Last thing is to tell PyMOL to interpolate the 100 frame zoom
 so we don't have to do those 100 snapshots:

#go back to frame 1
Frame1
mview store

#play the movie

mplay
```

2. Single zoom

   The script is identical to the previous one, except that you manually zoom to a ligand for example resid 1001.

   You can save movie images to numbered PNG format files with a common prefix. If you want each frame to be ray-traced, you should turn on raytracing of frames, turn off caching, and clear the cache (see the Movie Menu or use the following commands).

   ```
   set ray$\_$trace_frames=1
   set cache$\_$frames=0
   mclear
   ```

   You can save the movie using the"mpng" command, or you can save it from the File

menu. Either way, you must provide a prefix which will be used to create numbered PNG files.

```
mpng mov  # will create mov0001.png, mov0002.png, etc.
```

You need define all the render parameters before to create all png related to each frames of the movie mpng prefix [, first [, last]]

Options "first" and "last" can be used to specify an inclusive interval over which to render frames. Thus, you can write a smart Python program that will automatically distribute rendering over a cluster of workstations. If these options are left at zero, then the entire movie will be rendered.

3. More complicated More complicated movies can be done using different scenes or several moving objects tutorials can be done on this page.

8 Morphing

8.1 Morphing using LSQMAN/Pymol

You can generate intermediate pdb state between an initial pdb and a final pdb using LSQMAN. To do that you can use a script similar to the one below:

```
Script:
# generation of  30 pdb between 2 structures

lx_lsqman <<end-lsq
#Read in the two files
re m1 initial.pdb
re m2 final.pdb
#Tell LSQMAN to use all atoms
at no
#Fix silly nomenclature problems
nomen m1
nomen m2
fix
m1
a1-190
#(chain firstresidue-lastresidue)
m2
a1-190
strict
```

```
seq
torsion
#Morph between the two structures
morph
m1
A1-190
m2
A1-190
30 # number of pdb generated
morpha
c
m
A1-190
999
quit
end-lsq
```

8.2 Renumber the Files

This is actually the last line of the above script, so that renumbering is automatic. It renumber the files 01 → 30, so the "ls" command will put them in the right order.

```
ls morph?_?.pdb | awk -F "_" '{print ("mv "\$0" " \$1"_0"\$2)}' | /bin/sh
```

8.3 Assign Secondary Structure

The Pymol manual says not to trust its secondary structure assignment procedure (which is slow anyway). I used DSSP and a helper script dssp2pdb to add the secondary structure to each intermediate files. I used the following script to add the secondary structure elements to all the morphed PDB files.

```
#!/bin/bash
#Run dssp on the starting structure
dssp 2pdz-w.pdb 2pdz-w.dssp
#Get the stuff to add to each file
dssp2pdb -35 2pdz-w.dssp > tEmP
#If a morphed pdb doesn't already have it, add the secondary structure info.
for i in `ls morph*pdb`; do
 if ! [ `grep -l HELIX \$i` ]; then
  cat tEmP \$i > tEmP2
 mv tEmP2 \$i
fi
```

```
done
rm -f tEmP
```

8.4 Generate the Morphing movie

In Pymol, each morphed structure is considered one state of a molecule. Each state is loaded with the command

```
load file, object, state#
```

To automatically generate the "loading" part of the pymol script, I used the following command:

```
ls morph*pdb |awk '{print ("load",\$0",mov,"NR)}' > load.pml

which loads each PDB into a different state of the molecule mol with these commands.

load morpha_01.pdb,mov,1
load morpha_02.pdb,mov,2
...
load morphb_29.pdb,mov,59
load morphb_30.pdb,mov,60

Then generate the command that will do the movie. In this case:

"mset" sets up a relationship between molecular states and movie frames. This makes it
possible to control which states are shown in which frame.

# the first thirty frames are state 1
# the next 15 frames pass through states 1-15
# the next 30 frames are of state 15
# the next 15 frames iterate back to state 1

#mset ( (state 1) (xframes) )  ,  ((state 1 state15) (blank meaning one frame)) ;
( (state 15) (xframes) ) ; ((state 15 state1) (blank meaning one frame))

mset 1 x30 1 -15 15 x30 15 -1
mplay
```

9 Pymol Reference:

manual
http://www.pymol.org/

http://www.pymolwiki.org/index.php/Main_Page
tutorial
http://137.189.50.96/kbwong/teaching/pymol/pymol_tutorial.html
http://www.ebi.ac.uk/ gareth/pymol/
http://freedom.bph.jhu.edu/fleming/compbio/files/PyMOL_Tutorial.pdf
http://www.weizmann.ac.il/Structural_Biology/Pages/Levy/group_meet/PyMol_tutorial.pdf
Plugins: Emovie, APBS tool, Carver ect.
http://www.pymolwiki.org/index.php/Category:Plugins
http://www.weizmann.ac.il/ISPC/eMovie.html

10 Annex: Pymol reference card

Pymol Reference Card

Modes

Pymol supports two modes of input: point and click mode, and command line mode. The point and click allows you to quickly rotate the molecule(s) zoom in and out and change the clipping planes. The command line mode where commands are entered into the external GUI window supports all of the commands in the point and click mode, but is more flexible and possibly useful for complex selection and command issuing. Commands entered on the command line are executed when you press the return key.

command help	`help` *keyword*

Loading Files

file loading	`load data/test/pept.pdb`
loading from terminal	`pymol data/test/pept.pdb`
toggle between text and graphics	*Esc*
toggle Y axis rocking	`rock`
stereo view	`stereo on/off`
stereo type	`stereo crosseye / walleye / quadbuffer`
undo action	`undo`
reset view	`reset`
reinitialize Pymol	`reinitialize`
quit (force, even if unsaved)	`quit`

Mouse Control

	L Rota	M Move	R MovZ	Wheel Slab
Shift	+Box	-Box	Clip	MovS
Ctrl	+/-	PkAt	Pk1	—
CtSh	Sele	Cent	Menu	—
DblClk	Menu	Cent	PkAt	—

set the center of rotation `origin` *selection*

Atom Selection

object-name/segi-id/chain-id/resi-id/name-id

molecular system selection	`/pept`
molecule selection	`/pept/lig`
chain selection	`/pept/lig/a`
residue selection	`/pept/lig/a/10`
atom	`/pept/lig/a/10/ca`
ranges	`lig/a/10-12/ca`
ranges	`a/6+8/c+o`
missing selections	`/pept//a`
naming a selection	`select bb, name c+o+n+ca`
count atoms in a selection	`count_atoms bb`
remove atoms from a selection	`remove resi 5`
general	`all, none, hydro, hetatm, visible, present`
atoms not in a selection	`select sidechains, ! bb`
atoms with a vdW gap < 3 Å	`resi 6 around 3`
atom centers with a gap < 1.0 Å	`resi 6 near 1`
atoms centers within < 1.0 Å	`within 4 resi 6`

Basic Commands

Some commands used with atoms selections. If you are unsure about the selection, click on the molecule part that you want in the viewing window and then look at the output line to see the selection.

fill viewer with selection	`zoom, /pept//a`
center a selection	`center /pept//a`
colour a selection	`colour pink, /pept//a`
force Pymol to reapply colours	`recolor`
set background colour	`bg_color, white`
vdW representation of selection	`show spheres, 156/ca`
stick representation of selection	`show sticks, a//`
line representation of selection	`show lines, /pept`
ribbon representation of selection	`show ribbon, /pept`
dot representation of selection	`show dots, /pept`
mesh representation of selection	`show mesh, /pept`
surface representation of selection	`show surface, /pept`
nonbonded representation of selection	`show nonbonded, /pept`
nonbonded sphere representation of selection	`show nb_spheres, /pept`
cartoon representation of selection	`show cartoon automatic, a//`
clear all	`hide all`
rotate a selection	`rotate` *axis, angle, selection*
translate a selection	`translate [x,y,z],` *selection*

Cartoon Settings

Setting the value at the end to 0 forces the secondary structure to go though the CA position.

cylindrical helices	`set cartoon_cylindrical_helices,1`
fancy helices [tubular edge]	`set cartoon_fancy_helices,1`
flat sheets	`set cartoon_flat_sheets,1`
smooth loops	`set cartoon_smooth_loops,1`
find rings for cartoon	`set cartoon_ring_finder,[1,2,3,4]`
ring mode	`set cartoon_ring_mode,[1,2,3]`
nucleic acid mode	`set nucleic_acid_mode,[0,1,2,3,4]`
cartoon sidechains	`set cartoon_side_chain_helper; rebuild`
primary colour	`set cartoon_color,blue`
secondary colour	`set cartoon_highlight_color,grey`
limit colour to ss	`set cartoon_discrete_colors,on`
cartoon transparency	`set cartoon_transparency,0.5`
cartoon loop	`show cartoon loop, a//`
cartoon rectangular	`show cartoon rect, a//`
cartoon oval	`show cartoon oval, a//`
cartoon tubular	`show cartoon tube, a//`
cartoon arrow	`show cartoon arrow, a//`
cartoon dumbell	`show cartoon dumbell, a//`

Image Output

low resolution	`ray`
high resolution	`ray 2000,2000`
ultra-high resolution	`ray 5000,5000`
change the default size [pts]	`viewport 640,480`
image shadow control	`set ray_shadow,0`
image fog control	`set ray_trace_fog,0`
image depth cue control	`set depth_cue,0`
image antialiasing control	`set antialias,1`
export image as .png	`png` *image*`.png`

Hydrogen Bonding

Draw bonds between atoms and label the residues that are involved.

draw a line between atoms	`distance 542/oe1,538/ne`
set the line dash gap	`set dash_gap,0.09`
set the line dash width	`set dash_width,3.0`
set the line dash radius	`set dash_radius,0.0`
set the line dash length	`set dash_length,0.15`
set round dash ends	`set dash_round_ends,on`
hide a label	`hide labels, dist01`
label a reside	`label (542/oe1), "%s" %("E542")`
set label font	`set label_font_id,4`
set label colour	`set label_color,white`

Electrostatics

There are a number of ways to apply electrostatics in Pymol. The user can use GRASP to generate a map and then import it. Alternatively the user can use the APBS Pymol plugin. Pymol also has a built in function that is quick and dirty.

generate electrostatic surface `action > generate>vacuum electrostatics > protein contact potential`

Pymol Movies (mac)

move the camera	`move x,10`
turn the camera	`turn x,90`
play the movie	`mplay`
stop the movie	`mstop`
writeout png files	`mpng` *prefix* `[, first [, last]]`
show a particular frame	`frame` *number*
move forward on frame	`forward`
move back one frame	`backwards`
go to the start of the movie	`rewind`
go to the middle of the movie	`middle`
go to the movie end	`ending`
determine the current frame	`get_frame`
clear the movie cache	`mclear`
execute a command in a frame	`mdo 1, turn x,5; turn y,5;`
dump current movie commands	`mdump`
reset the number of frames per second	`meter_reset`

Miscellaneous

add hydrogens in to a molecule selection	`h_add`
alias a set of commands separated by ";"	`alias go,load 1hpv.pdb; zoom 200/; show sticks, 200/ around 8`
structurally align	`align prot1////CA, prot2, object=alignment`
fit one molelcule to another	`fit` *selection, target*
copy at selection	`copy` *target, source*
create a new selection	`create` *target, selection*
delete a selection	`delete` *selection*
save file	`save` *filename, selection*
protect or deprotect a selection	`[de]protect` *selection*
mask or demask to allow/stop selection	`[un]mask` *selection*
align coordinates with axis	`orient` *selection*
get the current rotation matrix	`get_view`
input a rotation matrix	`set_view`
safely refresh the scene	`refresh`
store a scene	`view` *name*, `store`, *description*
restore a view	`view` *name*, `[recall]`
set a new colour	`set_color` *name, rgb*

Secondary Structures

Pymol has a secondary structure determination algorithm called dss, however it is better to use the DSSP algorithm and then define the limits manually.

to run dss	`dss` *selection*
to define helical structure	`alter 11-40/, ss='H'`
to define loop regions	`alter 40-50/, ss='L'`
to define strand structure	`alter 50-60/, ss='S'`
rebuild the cartoon after alteration	`rebuild`
get dihedral angle	`get_dihedral 4/n,4/c,4/ca,4/cb`

Files

change the working directory	`cd <path>`
list contents of current directory	`ls`
print current working directory	`pwd`

Crystal Structures

To recreate crystal packing of molelcules within 5 Å of pept in the pept.pdb (which must contain CRYST date), use the symexp command. `symexp sym,pept,(pept),5.0`

NMR Structures

NMR models should be loaded into the same object, but should have different states. load a model into an object `load` *file.pdb, object*

show all models in an object	`set all_states,1`
show only one object model	`set all_states,0`
show a particular model	`frame` *model_number*
determine which model	`get_model`
fit two structures to one another	`fit` *selection*
fit and calculate the rms	`rms` *selection*
rms without fitting	`rms_cur` *selection*
fit ensemble structures	`intra_fit` *selection*`,1`
calculate rms	`intra_rms` *selection*`,`*state*
ensemble rms without fitting	`intra_rms_cur` *selection*`,`*state*

Changing Structures

add a bond	`bond atom1, atom2`
remove bonds	`unbond atom1,atom2`
join to molecules together	`fuse [atom1, atom2]`

Old School Images

Load a .pdb and make a cartoon view. Then change the background colour to white and change the ray mode to 2.

	`set ray_trace_mode,2`
make the lines thinner	`set antialias,2`
raytrace the image	`ray`