PyMol Movie-Making Tutorial PharmSci 272

This document provides a tutorial you can work through to do some movie-making in PyMol, with an example HIV protease structure and a small molecule 3D structure shown in class. It can easily be adapted to suit your own visualization purposes.

Overview

This tutorial focuses entirely on visualization, and emphasizes movie-making. You will work through a tutorial to make sample movies of two different molecules (a protein and a cavitand) using PyMol, including scripting for these movies using Python.

Your tutorial begins with making a very simple movie of a protein in PyMol interactively, using the menus. However, more advanced movie-making techniques take advantage of the easy interface between Python and PyMol. So, after your first intro movie, you will move on to some advanced movie-making, this time of a small molecule (but the same techniques would apply to a protein). We'll build up a Python movie-making script which will make a rather nice movie of our target molecule.

PyMol Issues and Installation

You can get PyMol for free online as an educational user via http://www.pymol.org/educational; however, the full version has somewhat more features and my group has an academic subscription which allows me to use it for courses as well, so I would recommend installing that version, though the terms of the license will require you to stop using this incentive build when you are done with the course.

To install the subscription version, download the appropriate file and install:

- Windows (32 bit): https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/apps/PyMOL-v1.7.0.1-Win32.msi
- Windows (64 bit): https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/apps/PyMOL-v1.7.0.1-Win64.msi
- OS X: https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/apps/MacPyMOL-v1.7.dmg
- Linux (32 bit): https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/apps/pymol-v1.7-Linux-x86.tar.bz2
- Linux (64 bit): https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/apps/pymol-v1.7-Linux-x86_64.tar.bz2

If you have PyMol but it isn't working properly for you, we may need to arrange for you to use another computer. If you need my help with this (I might be able to arrange for access to a computer in my lab) please let me know as soon as possible.

Some PyMol Basics and Background

I provide some links to PyMol documentation below. There is also a user-supported Wiki page with tutorials, scripts, and answers to questions (http://www.pymolwiki.org/) as well as a users' mailing list (http://sourceforge.net/mail/?group_id=4546). In view of what is already available, I will provide a some discussion of key areas and refer you to the documentation and help for more.

PyMol has both command-line (text entry) and menu-based versions of most common commands. The menus are better for beginners, and help you learn PyMol, but the command-line is better for power users of PyMol and for writing scripts. Specifically, PyMol can run commands in the Python programming language, so if you learn Python or pick up even just a bit of it to help you dabble in PyMol, your ability to do sophisticated things in PyMol will expand greatly. We'll come back to this. One important tip, however, is that the escape key switches between the molecular viewer and text views.

Common tasks in PyMol include structural alignment. To align two molecules (such as two proteins), right click one of them, choose action-> align-> to molecule and then choose the structure to align onto. Mutagenesis is another common task, where you may want to change one amino acid in a protein to another and determine possible ways the new amino acid could fit. To do this, go to Wizards->Mutagenesis Wizard, select a residue, and choose a target mutation (such as mutate to ARG), then use the play button right at the bottom to cycle through possible placements of the new residue. Select "apply" when done.

Making of simple movies in PyMol is easy. At the most basic level, you simply load a structure file (such a pdb or mol2 file, see later) containing multiple structures, such as from a simulation, and then use the movie controls at the bottom right to stop, play, or step frame by frame. PyMol can also 'render' these into movies which can be exported and played from within presentations. It's also worth knowing that the 'help' command provides some basic categories you can get help on, such as 'help selections' for help on how to select things, or 'help align' for help on the alignment command.

As noted above, PyMol can also easily be extended through Python. Here is an example script which, while technically in the Python programming language, just does a few simple PyMol things that you can more or less guess by reading it:

```
cmd.set('bg_rgb', '1,1,1') #set background color to white
for prot in ['1HVR', '1MUI', '1HXW']: #Loop over proteins
   cmd.load(prot+'.pdb', prot) #Load each protein as a new object
   cmd.align(port, '1HVR') #Align each protein
cmd.show('cartoon') #Show cartoons for protein
cmd.hide('lines') #Hide lines
cmd.center('1HVR') #Center on one protein
cmd.remove('SOLVENT') #Remove solvent
cmd.show('sticks', 'HETATM') #Show sticks for ligands bound
```

This simply cleans up the representation of three different proteins which are loaded, and aligns them to one another, removing solvent and adjusting how they are displayed. If this is saved to a text file 'load_orient_color_show.py", one could perform these operations by doing 'run load_orient_color_show.py' within PyMol.

Rather sophisticated movies are possible within PyMol as well, as we will see in this tutorial.

Documentation Links:

- Introductory tutorial with supporting files: https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/IntroductionToPyMOL.zip
- Intermediate tutorial with supporting files: https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/IntermediatePyMOL.zip
- Moviemaking tutorial with supporting files: https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/Moviemaking.zip
- Molecular editing tutorial with supporting files: https://dl.dropboxusercontent.com/u/3409095/Research%20Group/PyMol/MolecularEditing.zip

Tutorial Part 1: Make a very simple protein movie

In this section, you will make a simple movie of HIV protease, an important protein in HIV replication.

Begin by visiting www.pdb.org and entering the PDB code 1HVR, select the PDB code 1HVR, and click on the link at right to "download files" and choose "PDB file (text)". Save this to your computer in a directory you can easily find. Whatever directory this is, we will refer to it as your "working directory" in what follows. Make a note of it, as you will be placing other files here.

Once you have the PDB file, open PyMol. Type the linux command "pwd" into the PyMol command line to see what directory you are currently in (for more info, use Google to look up info on basic linux commands), and use "cd" to get to the working directory you just saved your PDB file into, above. (On Windows, this would look something like, 'cd C:\path\to\my\directory' and on OS X, something like, 'cd /Users/myusername/mydirectory'). Then, type the following commands:

```
log_open mysession.py
load 1HVR.pdb
show cartoon
hide lines
```

From the menus, choose Movie -> Program -> Camera Loop -> Nutate -> 30 degrees over 8 seconds. Then, use the play button at bottom right to play your movie (note that you can adjust the speed at which it plays -- the frame rate -- from the menus). Finally, type 'log_close' (no quotes).

Open your mysession.py file with a text editor to see what's there (under Mac/Linux, I suggest Fraise (http://www.macupdate.com/app/mac/33751/fraise), vi or another plain text editor, NOT TextEdit; for Windows, you'll probably want a free text editor that works well with Python. If you installed Python on your computer, that probably includes one, otherwise you may want to try ActiveState's Komodo Edit which is free).

This file should have a record of the command you used, and you could, if you liked, open a new PyMol session and type the command 'run mysession.py' (no quotes) to run this saved file and repeat all the commands you had just typed.

On many platforms, you could also do 'rendering' at this point (that is, drawing and saving images) and turn these into a movie file that you could actually save out of PyMol and use in presentations, etc. You could turn on ray tracing of the frames to make them look very nice, though this slows the rendering considerably, to the point where it could take several hours.

Tutorial Part 2: Move between views of a molecule

In this section, you will create a simple movie that morphs between two views of a molecule. Here, we will do this by picking the initial view using the mouse and then creating our movie from the command line. In the next section, we will see how to adjust the view from the command line itself. We will save the sequence of commands you use and use it as the basis for more complicated movies.

To start off with, get Octa-Acid.pdb from the course web site and save it to your working directory. Then, use the following commands in PyMol (it's not necessary to type the comments, which are there for your information):

```
log_open movie1.py
cmd.set('bg_rgb', '1,1,1') #Makes background white
cmd.load('Octa-Acid.pdb') #Loads structure
cmd.show('sticks') #Shows sticks representation
```

Switch your view of the molecule using your mouse to orient it how you would like it at the start of your movie, which is going to consist of rotation around the vertical axis. Then, make a 300 frame-movie where, as a starting point, all frames consist of copies of the first frame, and store the current view to the first frame of the movie:

```
cmd.mset('1 x300') #Make a movie consisting of 300 frames
cmd.mview('store','1') #Store current view to frame 1
```

Now, we're going to rotate 180 degrees around the vertical axis and make that the mid-point of our movie, then rotate another 180 degrees and make that the endpoint of our movie. Intermediate frames will be interpolated (basically, figured out based on the frames we've done) by PyMol, though if you don't like how this is done you can fill in more intermediate frames yourself.

```
cmd.mset('1 x300') #Make a movie consisting of 300 frames
cmd.turn('y', '180') #Rotate 180 degrees
cmd.mview('store', '150') #Store rotated view to frame 150
cmd.turn('y','180') #Rotate another 180 degrees
cmd.mview('store, '300') #Store rotated view to frame 300
log_close #Stop writing log
```

Now, try playing your movie and see how it looks.

Next, open your log file (movie1.py) with your text editor as above, and verify it contains these commands. Note that you could remove the 'cmd.do' enclosing each command -- that's here because of how we created the script, but it would work fine without it.

Wrap this section up by testing that your saved script can re-create the movie you just made. Type 'reinitialize' (no quotes) in PyMol to clear things out, then type 'run movie1.py' to run the movie1.py script you just saved. Verify it creates your movie.

There is one big problem with the movie1.py script we just made. Any changes you made initially with your mouse to reorient your molecule will have been lost, as these were not written to the log file. If we want to modify the initial view, we need to add something to movie1.py to adjust the initial view.

One important note -- you can get usage information in PyMol using the 'help' command, so that, for example, you can get help on the cmd.mset command using 'help(cmd.mset)'.

Tutorial Part 3: Refine your movie by altering the initial view

Here, we want to fix the problem just noted -- our molecule loads in a less-than-ideal orientation and we'd like to alter the initial view within our movie-making script (without having to use the mouse).

The first step, here, is to load up your Octa-Acid.pdb again and find an initial view that looks good to you by experimenting with the 'turn' command, i.e. 'cmd.turn('z','45')' (no quotes) to turn around the z axis 45 degrees . (Note that you can also undo the last command by typing 'undo'). Experiment with this until you find an initial view that looks good to you.

When you have an initial view you are happy with, make a copy of movie1.py to movie2.py (in Linux/Terminal, use 'cp' to do this; on Windows, consider using the file explorer and on Mac, Finder.).

Once you have movie2.py made, edit it to insert the turn command right after the PDB file is initially loaded, then save your changes.

When you have modified movie2.py, run 'run movie2.py' (no quotes) in PyMol to create your movie using the new initial view. Verify this works.

Tutorial Part 4: Add some frills to your movie

In this section, let's fancy up the movie. Let's make it longer, and incorporate a change of representation (from sticks to spheres) partway through, as well as adding in some extra zooming. Copy movie2.py to movie3.py and edit it. Make the following changes:

- Switch your initial movie length to 600 frames
- At the end of the current script, before the play command, add 'cmd.mdo(300, 'show spheres')' (no outer quotes) to switch to sphere representation at frame 300
- After the show spheres, add 'cmd.zoom('Octa-Acid', -5)' and 'cmd.turn('x','90')' to zoom in on the octa-acid with a "buffer" of -5 angstroms (meaning that we'll zoom so far we clip off 5 angstroms of the molecule at the edges) and rotate 90 degrees in the x direction
- Store this to your movie: 'cmd.mview('store', '600')'
- Save your script and run it in PyMol

If you get what I got, this will be looking pretty good, except that the sudden appearance of the spheres is a little startling. Let's see if we can make them gradually appear by turning them from transparent into non-transparent over a sequence of steps.

Make a new copy of your script to movie4.py, and edit just before the 'show spheres' command to add 'cmd.mdo(299, 'set sphere_transparency = 1.0')' to make these transparent. Then, just

AFTER the show spheres, command, add a 'for' loop to gradually make the spheres appear, with code something like this:

```
for i in range(301,400):
   if (i-300)%5 == 0:
      transparency = 1.0 - (i-300.)/100.
      cmd.mdo( i, 'set sphere_transparency = %.2f' % transparency )
cmd.mdo(401, 'set sphere_transparency = 0.0')
```

This will gradually change the sphere transparency from full transparency (1.0) to no transparency (0) with increments every 5 steps. You can easily modify this to suit your tastes.

Save your script and run it from within PyMol. Check out your final product.

Tutorial Wrap-Up:

I have placed two other example scripts with input files on the website. One is a slightly fancier version of the Octa-Acid movie-making script that you just put together, and another does some fancy zooming and rotation on HIV protease. Both are commented to give you an idea what they do and how, and you're encouraged to download the input files and try them out.

Your task:

I will likely assign something like the following during the second half of this course: Pick something you would like to visualize -- perhaps a small molecule or protein you are already interested in, or if not, pick something from the PDB and visualize it. Make a script to make a movie of this, using what we did for Octa-Acid as a starting point. You essentially can set this up however you want (and getting it to do what you want may involve some trial and error, and I'm happy to provide pointers) but your final movie should involve at least the following elements, at minimum:

- A morph between at least two views of the molecule (such as a rotation or zoom)
- At least one change of representation

You may also want to use other elements, such as changes in transparency, changes in color, or even changes in what is visible (for example, you could reveal a ligand in a binding site, having it initially invisible). Ideally, your movie should focus around some point you want to make.

If you can easily do so, you may want to save the movie to a file for later reuse. In the Mac version of PyMol, this is as simple as File-> Save Movie As (and you may want to turn on raytracing of frames first) and may be similarly easy in Windows or Linux depending on your installation.