DESKTOP MOLECULAR GRAPHICS *PyMol*

CONTENTS

==========

From the preface of the User's guide: "*PyMOL was created in an efficient but highly pragmatic manner, with heavy emphasis on delivering powerful features to end users. Expediency has almost always taken precedence over elegance, and adherence to established software development practices is inconsistent. PyMOL is about getting the job done now, as fast as possible, by whatever means were available.*"
PyMol is a multiplatform molecular graphics software with many advanced features such as rendered cartoon ribbons and surfaces, internal ray tracing and movie tools and is fast becoming the new "standard" in molecular graphics. The PyMol web site is located at `http://pymol.sourceforge.net/`

==========

Note: within the exercises, **Bold** text shows what actions are taken by the user : typing text or clicking the mouse.

==========

***PyMol - Exercise A*:** Download a PDB from the repository

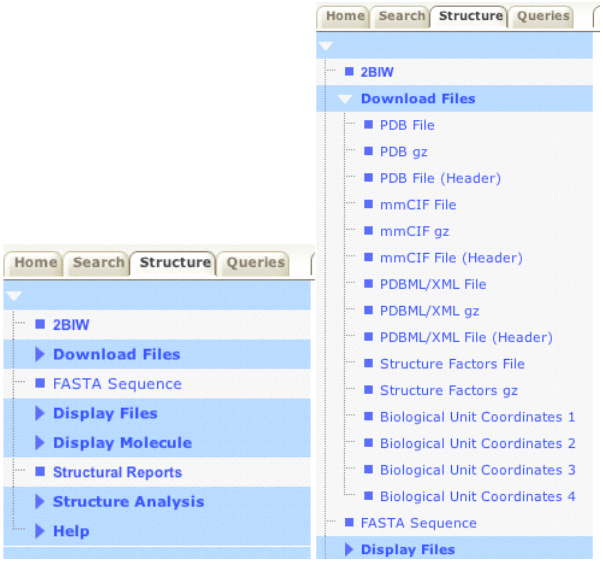
Reminder: Structures have a PDB ID code made of 4 letters and numbers. PDB files contain coordinates pertinent to the crystallographic arrangement of the molecules within the crystal. The biological functional entity can be either a multimer of the deposited structure, or just one of multiple copies within the file. In the following example we will download one functional biological subunit, in this case a monomer.

1) Open a web browser such as Safari or Firefox.
2) Point your web browser to **`www.rcsb.org`**
3) In the Search box enter the following ID: **2biw** and click SEARCH button

4) On the left column click **Download Files** to show submenus
5) **Click** the first option **Biological Unit Coordinates 2**

Note: DO NOT USE Biological Unit Coordinates 1

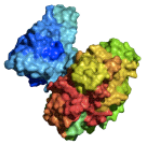
6) The file is saved on the desktop as 2BIW.pdb2



You can now close your browser, or hide it (Command-H).

==========

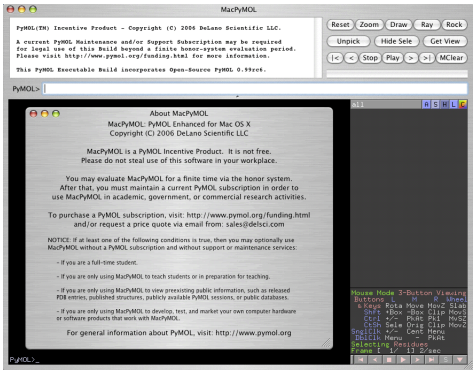
PyMol - Exercise B: Open PyMol and load a PDB file



PyMol or MacPyMol should be located within the Applications > Classes at the BNMC computer. Your instructor may give you a different location if necessary.

1) **double-click** on the PyMol icon to launch the software.

Note: Some versions of PyMol have the top and bottom panels in separate windows, but offer the same interface.



2) Type the next commands after PyMOL> **within the *top* line command:**

PyMOL> **cd desktop**
PyMOL> **pwd**

Note the echo on the text area above this will echo `/Users/BNMC/Desktop` or a similar path. On a Windows system the path would begin with C:\

PyMOL> **load 2BIW.pdb2**

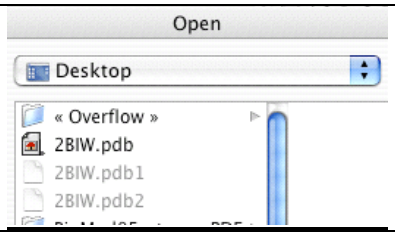
```
PyMOL>load 2BIW.pdb2
HEADER    OXIDOREDUCTASE                          19-JAN-
05   XXXX
```

This will load the structure and echo information in the text panel as reproduced to the right.

Note: file name is case sensitive

```
TITLE     CRYSTAL STRUCTURE OF APOCAROTENOID CLEAVAGE
OXYGENASE FROM
TITLE    2 SYNECHOCYSTIS, NATIVE ENZYME
COMPND    APOCAROTENOID-CLEAVING OXYGENASE
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 4 symmetry operators.
 CmdLoad: "2BIW.pdb2" loaded as"2BIW.pdb2".
```

Note: why not use the **File> Open** menu sequence? Good question… You can actually try that option and it might work on your current system. However you will likely find that when you get to the *Open* form the files you want to open are listed in gray and therefore cannot be opened. The easy fix is to simply rename the file from *.pdb1 or *.pdb2 to simply *.pdb before you try to open it.



==========

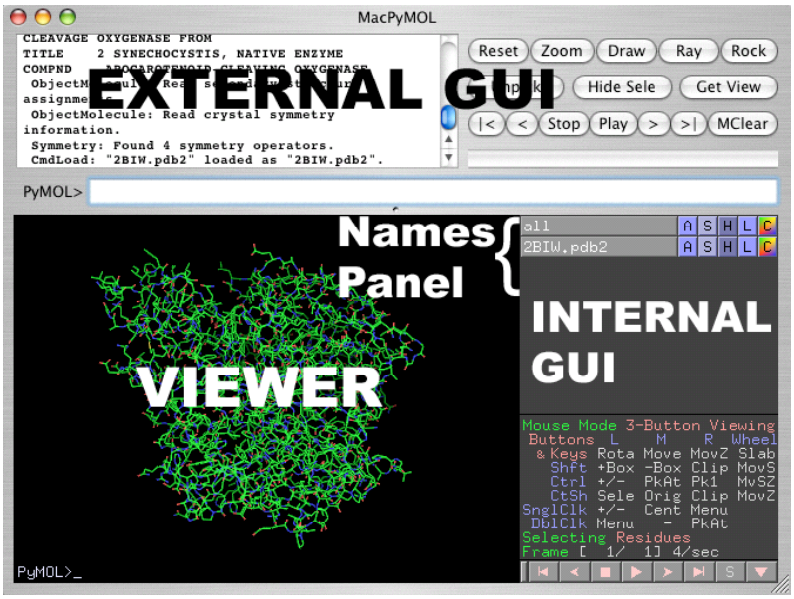
PyMol - Exercise C: PyMol interface

This exercise continues on the previous exercise where 2BIW.pdb2 was loaded within the PyMol software.

Your screen should be similar to this image without the extra markings.

The Viewer and Internal GUI are the parts we will use most.

The default size of the image in the viewer is 640 x 480 pixels.



The menu bar might be attached to the external GUI in systems other than Mac.

By default PyMol will display the molecule(s) contained within the PDB file as a wireframe.

This cyanobacterium molecule is a carotenoid oxygenase and contains a carotenoid ligand.

1) Mouse control of the 3D representation

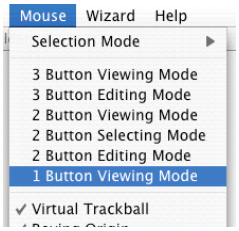
The 3D molecule is represented within a virtual 3D world. The flat surface of the screen represents the X and Y axes while the Z, depth axis is perpendicular to the screen.

PyMol is optimized for a 3-button mouse but most basic functions can still be achieved by a one-button mouse, in particular the rotations around X, Y, and Z.

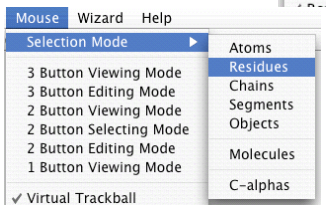
Rotation around the X or Y axis:	(left) click and drag.
Rotation around the Z axis:	(left) click on the top left or right corner.
Translate (move sideways) X or Y:	click middle button and drag.
Zoom (move along Z axis):	click right button and drag up or down.

2) Changing the default mouse settings

By default PyMol assumes that you have a 3-button mouse. If you have a 2- or one-button mouse you can change the setting accordingly with the "Mouse" menu in the top menu bar.
Note the updating of the "Mouse Mode" button mapping at the bottom right within the "Internal GUI" if you change the mouse setting.



Note: the Selection Mode> submenus define what is selected when an atom is clicked on. The selection default is Residues.



3) Changing the representation of the molecule

PyMol can open more than one molecule at a time, or separate complex PDB files into individual components. Each opened or loaded molecule is given a name within the "Names Panel" (see picture above). The first name is always "all." Clicking on the name itself will undisplay the corresponding molecule(s) (temporarily invisible).

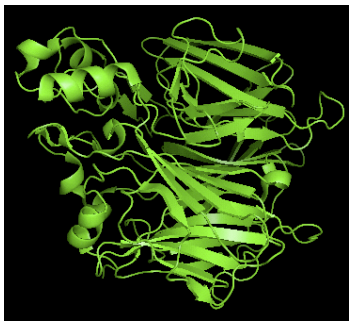
The **ASHLC** menu (

) is abbreviated for **A**ction, **S**how, **H**ide **L**ablel and **C**olor. Some menu items have submenu components. Selections made under the "all" line will affect all the opened molecules.

Rule: Once a selection is shown (S) it must be selectively hidden (H) as it is not removed when another selection (S) is made. Selections are therefore additive, which allows for the creation of images with mixed graphical representations.

Let's first make a cartoon representation of this protein: within the 2BIW line **click S** and select **Cartoon**. The molecule is now shown as both cartoon and wireframe. Remove the wireframe by **clicking H** and **lines.**

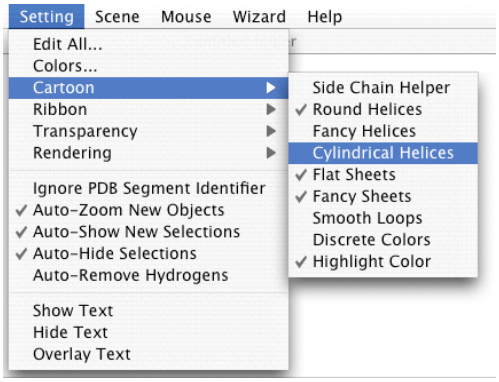
(Note that within the S menu list, the "as" menu contains a lot of redundancies.)



Options: most options can be set within the “Setting” menu within the top menu bar.

For example, it is possible to change the way all alpha helices are rendered.

Go to the following menu cascade: **Setting > Cartoon > Cylindrical Helices**
Select this option **again** to remove its effect and do the following: **Setting > Cartoon > Fancy Helices**



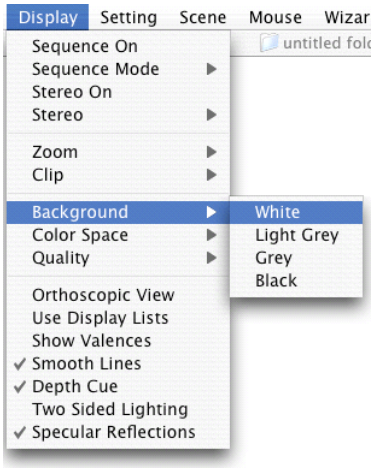
Testing other cartoon settings: Engaging the option **Smooth Loops** will simplify the drawing. Removing the option **Highlight Color** will make the edges of strands and inside helices surfaces a gray color (default).

Black backgrounds look very nice on the screen but do not print well on paper and do not photocopy well. Changing the backgroud to white is usually very useful:

The “Display” menu within the top menu bar contains options for most options pertinent to displaying the image witin the PyMol viewer.

To change the background color to white follow this menu cascade:
Display > Background > White

Note that there is “fog” within the back of the molecule, which can be toggled on and off with the **Depth Cue** menu item within this same menu list.



Changing the color of the ribbon is easy with the following cascade menu within the PyMol “Names Panel” of the “Internal GUI” under the C menu as shown in the following menu cascade:

2BIW.pdb2 > C > oranges > orange

You may also choose another option which is to color by secondary structure by following this menu cascade instead:

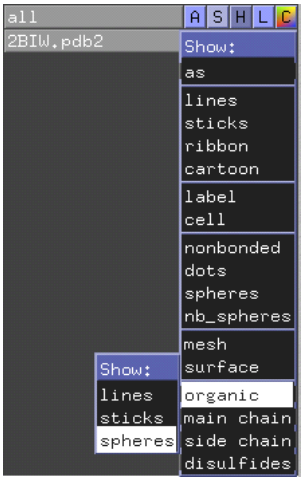
2BIW.pdb2 > C > by ss > Helix Sheet Loop (choose one of the the 2 color options displayed e.g. pink and cyan.

4) changing the representation of the molecule: adding ligand

When we opted to show the molecule as a cartoon above, one thing happened: the protein was shown as the familiar cartoon represenation, but if any ligand is present (which is the true for this file) it simply disappears as it is not part of the cartoon representation of the protein. Here we will “rescue” the ligand!

This menu cascade is also within the “ASHLC” menus of the PyMol “Internal GUI” “Names Panel.” Since ligands are usually small, organic molecules, the following cascade within the **Show** menu for line 2BIW will show the ligand. Perform the following cascade:

2BIW.pdb2 > S > organic > spheres



This cascade will select the carotenoid present witin the PDB file.
Note that it is the same color as other parts of the protein, as it is part of the name 2BIW.pdb2 line. It will be either orange or the color for loops in the C/ss/Helix-Sheet-Loop color scheme chosen in the step above.

5) Mouse selection

Now that we have made the ligand visible, it becomes easier to select it with the mouse to make further changes.

Click on one of the spheres of the carotenoid ligand. This simple click makes various things happen:

a. pink, square dots appear onto all the spheres of the lignad, indicating that it has been selected.
b. A new name line appears within the “Names Panel” GUI called “(sele)” and is now dedicated to this subset of atoms. Note that the content of (sele) changes as other atoms are clicked
c. The name of the atom that was clicked appears within the top text window of the “external GUI.” For example:

```
You clicked /2BIW.pdb2//B/TYR`322/OH
Selector: selection "sele" defined with 12 atoms.
```

This could be read as “you clicked atom OH which is on the 322nd atom in the file, and belongs to Tyrosine 322 of chain B in the object created when opening file 2BIW.pdb2. The complete selection contains 12 atoms (which make up the complete Tyrosine without the hydrogens, since these are usually not present in PDB files.)

Now that the ligand atoms are segregated within the name of (sele) we can change its color if we want by the following cascade within the Names Panel:

(sele) > by element > CHNOS….
(selecting the first CHNOS… after HNOS… in the list would display the ligand as green with one red oxygen).

6) Final image(s)

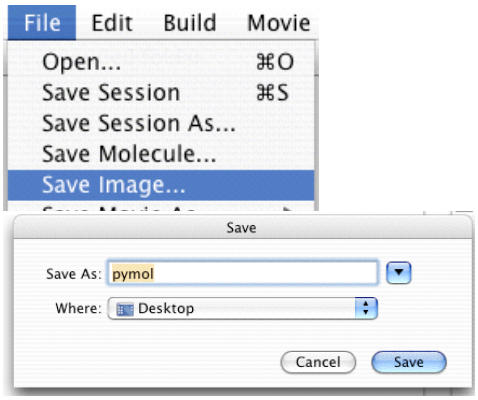
If you still see the selection dots over the ligand from the previous section simply **click anywhere** on the white background to unselect. Alternatively click on the “Hide-Sele” button at the top right hand side of the “external GUI.”

Rotate the molecule to find a perspective that you deem instructive of the conformation of the protein and it’s bound ligand.

Follow this menu cascade to save the image currently within the Viewer:

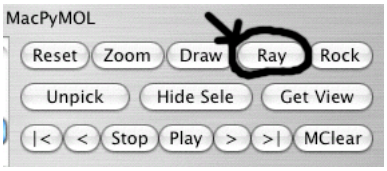
File > Save Image…

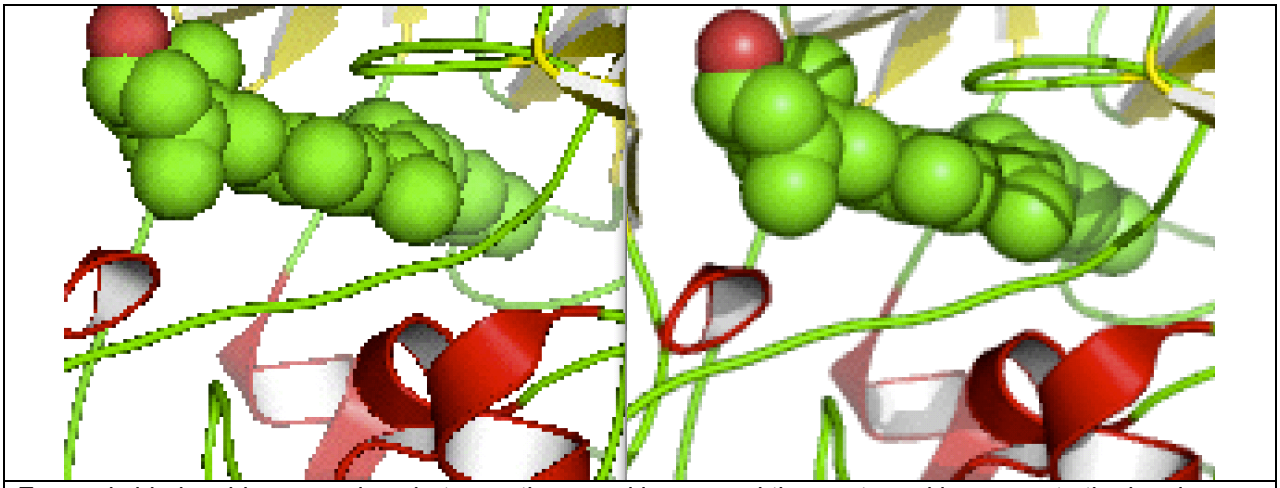
Then replace the default word “pymol” to give a name to the file you want to save, e.g. image1
The image will be saved as a PNG image on the desktop



However this image is rather crude in terms of graphics and resolution. PyMol offers an internal “ray tracer” to create stunning rendered images with a high visual quality much more pleasant to the eye and ideal for publication.

To create a standard ray-traced image of the current Viewer scene, **click the “Ray”** button at the top right of the “external GUI.” This will take a few seconds to a few minutes depending on the complexity of the PDB file and the the chosen display, and will also depend on the speed of the computer CPU. Once rendered, the image appears within the Viewer. To save the file, use the save cascade as above: **File > Save Image…**





Zoomed side-by-side comparison between the pymol image and the ray-traced image: note the jagginess of the original image and the smooth appearance of the ray-traced image, with shadows as a bonus.

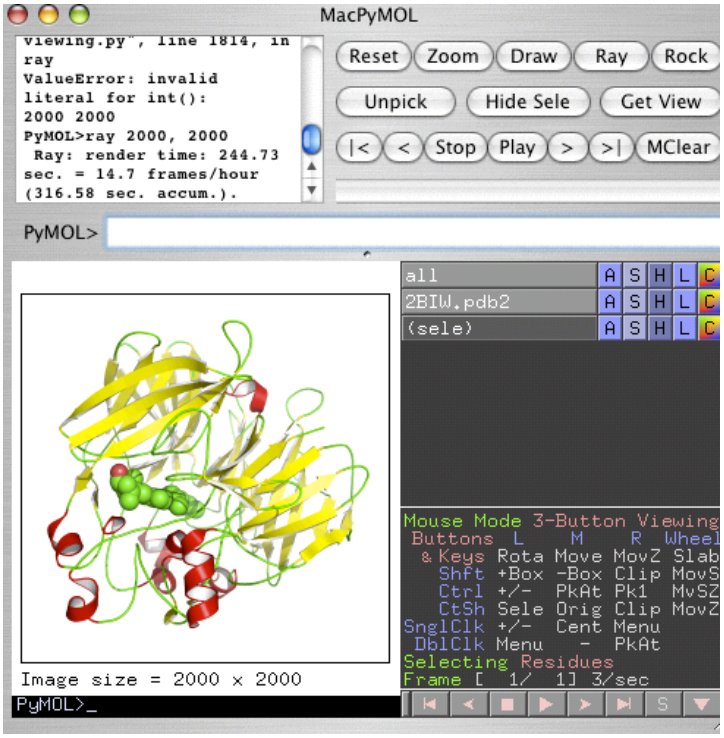
This is useful to create preview images for slide presentations, but is still one step short of the quality needed for high resolution publication, as the image created by the default setting is the same size as that of the current Viewer (default size is 640 x 480 pixels when PyMol is first opened.) The default file format is the new PNG standard. All we need to do at this point is change the default setting of the ray-traced file size.

Within the PyMOL> line command **type** the following command to change the dimensions of the final ray-traced image:

ray 2000, 2000

(2000 is an example number. Note the comma between the two numbers. The higher the numbers the longer it will take to compute the image). PyMol will display the size of the calculated image within the Viewer under the image here 2000 x 2000

Note: `ray 2000` is also a valid command. In this case PyMol will evaluate the missing value and adjust it to minimize the file size.



==========

PyMol - Exercise D: Action preset menus

This is mostly a self-paced exploration of one of the menus that is sure to change over time as PyMol evolves.

If you are not continuing this exercise from the previous exercise load the structure 2BIW.pdb2 as in Exercise B, then click S> Cartoon to be in a similar sate as in the previous exercise.

1) <u>The preset menu: from default to more complex</u>

The preset menu is part of the Action set of the ASHLC menus controlling the aspect of molecules from the Names Panel.

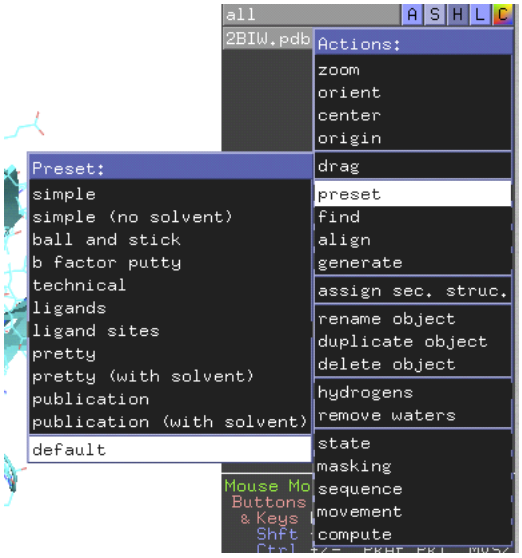
Follow this menu cascade sequence to return to the default view, as when you just opened the molecule. However no rotation occurs:

2BIW.pdb2 > A > preset > default

This command has a similar effect but is not the same as the following cascade: hide everything and show lines:

2BIW.pdb2 > H > everything
and
2BIW.pdb2 > S > lines

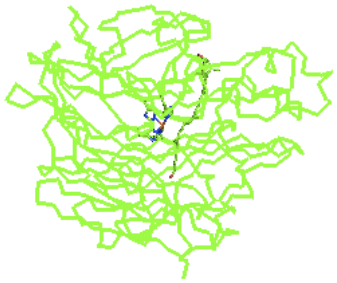
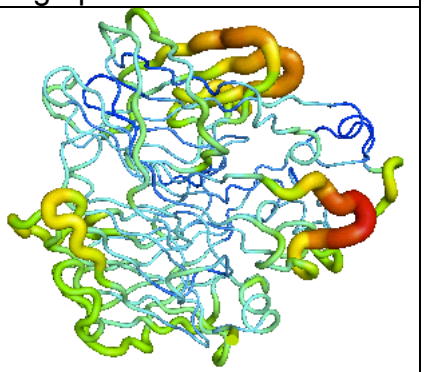
Note: the "preset" options will set some variables that are specific to these views and may change further drawings. To remove the effect of these presets affecting an object representation, use the A>preset>default menu cascade reset parameters.

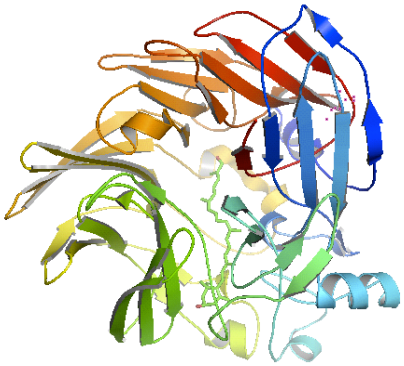
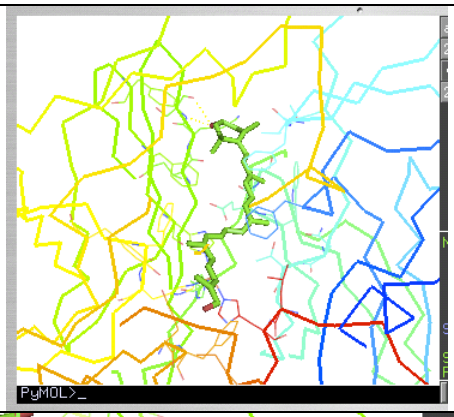
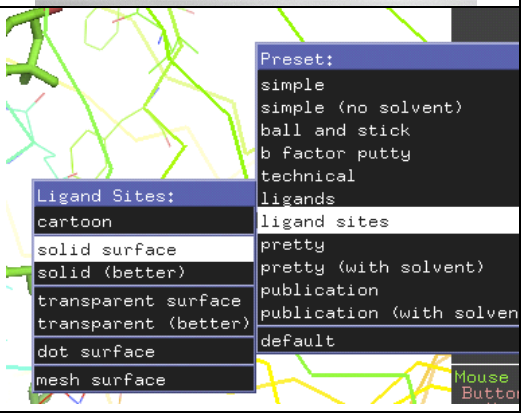


Note: to get back to the original opening view simply type **reset** at the PyMOL> line command.

2) Exploring more

Explore the other menus of this series.
The cascade menu **2BIW.pdb2 > A > preset** is assumed in the following commands:

simple The tracing can then be made thicker by unselecting the smooth option with the following menu cascade: **Setting > Ribbon > Smooth** Note: a set of 3 histidines is also shown.	
ball and stick	This is not very useful for a large protein such as this.
b factor putty The segments with the highest temperature factor are shown as thicker cylinders. Regions of better resolution have thinner diameter and are usually found at the core of the protein. Mostly loops in the outside of the protein wobble: the core portions of the proteins usually appear more stable than the external loops. This is mostly useful for crystallographers but is a cool representation.	
technical	Color domains in separate rainbow colors and shows backbone and side chains. Note that a subset name appears in the Names Panel (2BIW.pdb2_pol_co) that control the dashed-line hydrogen bonds.

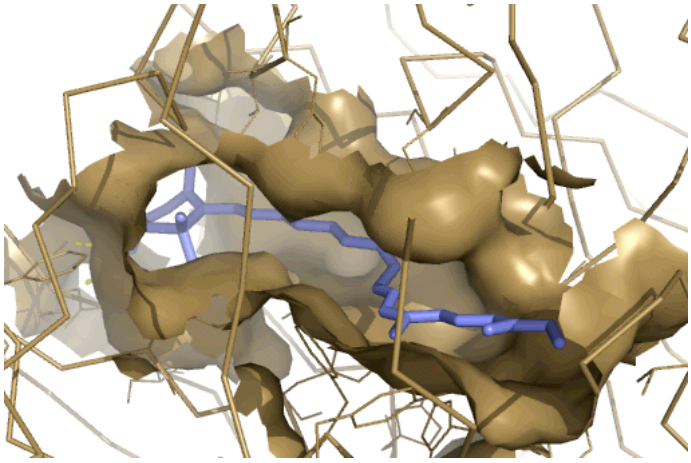
pretty and **publication** create similar images of a rainbow color cartoon with a stick ligand. Pretty creates the default cartoon setting, while publication creates an image very similar to an image created by Molscript*. (* Per J. Kraulis 1991: MOLSCRIPT: A Program to Produce Both Detailed and Schematic Plots of Protein Structures. *Journal of Applied Crystallography*. **24**: 946-950. http://www.avatar.se/molscript/)	
ligands This option will zoom in on the ligand site and show the protein as backbone except in the near vicinity of the ligand where side chains are shown. The ligand is depicted as a thicker cylinder. Note: to zoom out, simply click on the A in the ASHLC menu again.	
ligand sites There are a few options available in this submenu, all pertinent to looking closely at the ligand in it's binding pocket. You can explore a few of them on your own, once you are done then select the following option: **preset > ligand sites > solid surface**	

You should obtain a centered, zoomed view of the ligand shown as a stick model within a partial molecular surface pocket. The colors are the scheme of previously chosen colors, for example if you tried the "technical" preset earlier the coloring would likely be like a rainbow. The following quick changes will make a much nicer image: **2BIW.pdb2 > C > yellows > sand**

Now with the mouse click carefully anywhere on the ligand stick to select it. As before you will have a (sele) name within the Names Panel. You can now change the color with the following menu cascade: **sele > C > blues > slate**

Of course you can pick your own colors. Make sure that there is sufficient contrast between the color of the surface and the color of the ligand.

If you have not yet done so, **rotate the molecule** to select a nice viewing angle.



If you want to add a more stunning effect **click on the "ray"** button as we did before to complete your publication quality picture.

Note: If you are preparing a figure for a black and white print publication, it might be advantageous to use the various gray scales, black, and white within the "grays" option.

==========

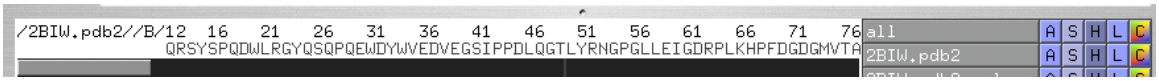
PyMol - Exercise E: Useful commands to analyze structure and create images

We are still working with the 2BIW.pdb2 file loaded above. Review exercise B if you need to reload the structure.
This exercise assumes you have just completed creating the above view with the preset menu cascade: **2BIW.pdb2 > A > preset > ligand sites > solid surface**

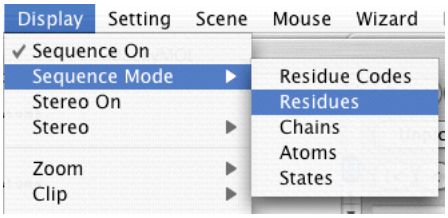
1) Sequence viewer

The molecule sequence can be shown at the top of the graphical area. To do this follow the top menu: **Display > Sequence on**

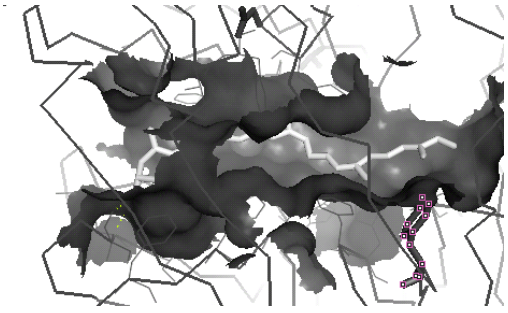
The sequence appears just below the PyMOL> line command at the top of the Viewer. The slide cursor underneath can be used to move the sequence viewed further.



By default the one-letter code is displayed. Since we are looking at a protein, changing the display to the three-letter code will be useful within the next step . Do this with the menu cascade:
Display > Sequence Mode > Residues



Now **rotate the molecule** to view similar to that shown at right.

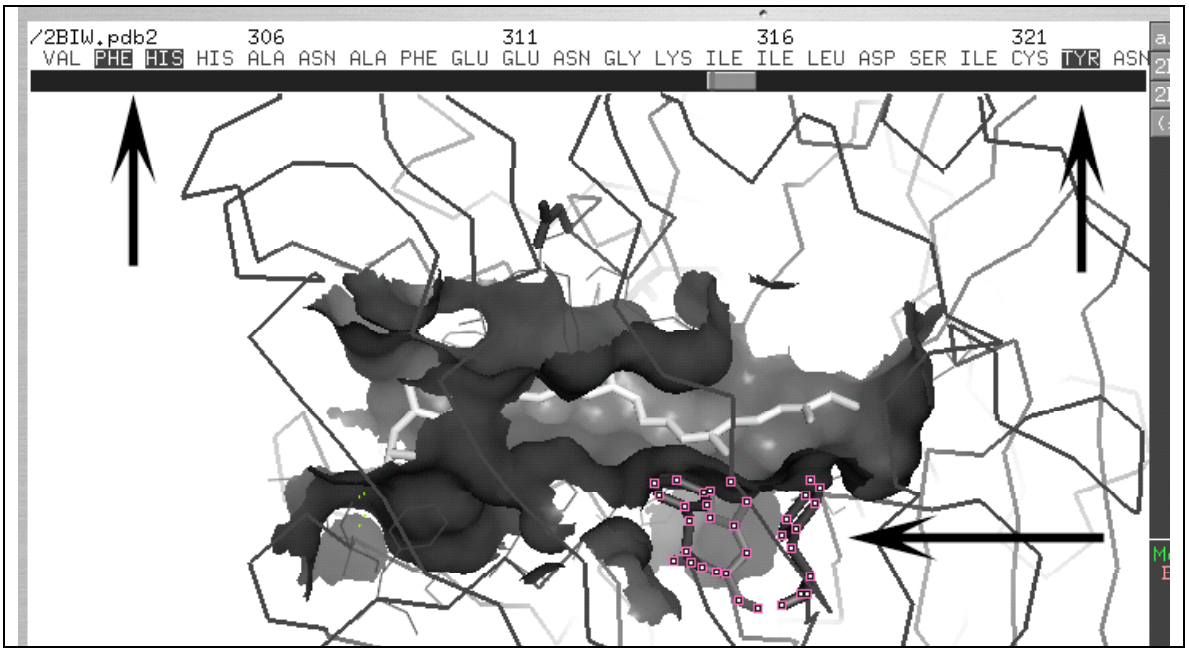


With the mouse, **click on** the tyrosine residue under the right hand side, which contribute to the pocket surface. Change it's aspect to stick with the following menu cascade: **(sele) > S > sticks**

Note: when you clicked, information about the clicked atom appeared within the text box of the "external GUI," e.g.:"`You clicked /2BIW.pdb2//B/TYR`322/CG.`"
Therefore we know that Tyrosine 322 is the one we chose.

Now **slide the sequence cursor to the right** to move the sequence displayed until you find the region of the selected TYR. If you have the one-letter code engaged, it appears under the number 321. However we know it is 322. If you have engaged the three-letter code as instructed above TYR 322 is shown highlighted within the sequence line.

Now **display the sequence line** to see residues PHE 303 and HIS 304. With the mouse **click on residue 303 and residue 304 within the sequence line**. Note that they are selected within the graphical window. The arrows within the following image show where you should look. As before, transform those side chains to a thicker stick with the menu cascade:
(sele) > S > sticks



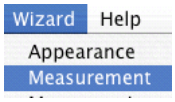
Quiz question: We just selected amino acids TYR 322, PHE 303, and HIS 304. Can you describe with one word one important quality of this binding pocket?
Answer: This binding pocket has the following quality: _____

2) Measuring distances

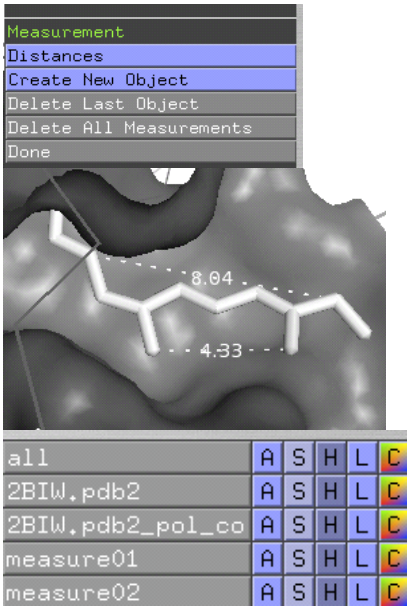
Distances are measured between two atoms and are expressed in the same unit as the XYZ coordinates within the PDB file: Angstroms (1Å = 10^{-10} m).

As an example we shall measure the distance between two atoms within the carotenoid ligand.

Within the top menu select
Wizard > Measurement
(Note: in older PyMol versions Measurement was called Distance.)
This will create a prompt within the Viewer:
" Please click on the first atom…"



Within the “Internal GUI” a “Measurement” table also appears. It can also be used to remove measurement objects after they are no longer needed.



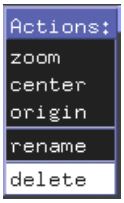
Click on the first atom on the ligand
Click on the second atom on the ligand

(you can select the same atoms as 2 examples within the image at right, or select alternate atoms)

The default color of the measurement object is yellow. This can be easily changed with the familiar ASHLC menu for each measurement, as a new name is entered within the “Names Panel” for each distance, e.g. “measure01.”
Change the color to white with the following menu cascade:
measure01 > C > grays > white

Quiz exercise: you can also measure the distance between the OH at the tip of TYR 322 and the nearest ligand atom (C30). *Answer:* _____

When you are done using the “Measurement” panel on the bottom right **click Done.**
If you no longer need to display the distance object, **click Delete All Measurements**.

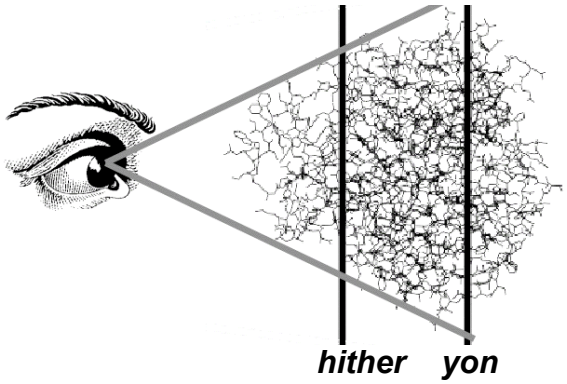


Alternatively you can use the corresponding “A” menu and select the delete option.

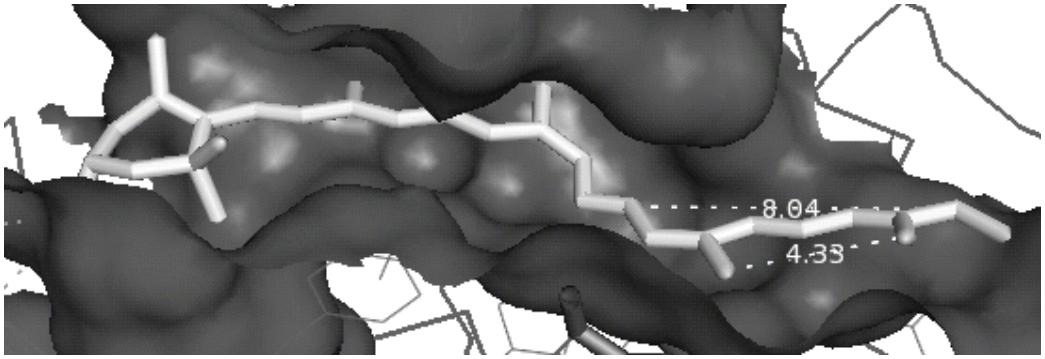
3) Clipping planes

Clipping planes are imaginary planes in the front and back of the molecule. Parts of the molecule that are outside the planes are “clipped” and therefore invisible. This is very useful for complex or large structures.

This image represents the molecule seen side-ways “inside” the computer monitor.
The 2 black lines represent the *yon* (far away) and *hither* (close) clipping planes. These are parallel to the flat screen of the computer monitor display. The gray lines converge toward the user’s eye, who is looking at the molecule on the computer screen.



The BNMC computers are equipped with 3-button mice. To move clipping planes **press shift** and the **right mouse button** simultaneously while dragging up and down. As an exercise, **try to remove some of the molecular surface** covering the ligand to create a picture similar to this:



Note: we have seen previously that the top menu “Mouse” contains all the possible options, with a reminder displayed at the bottom right of the “Internal GUI.” The method to adjust the clipping plane will depend on the number of buttons on your mouse. Use the “Mouse” menu to adapt your display to your mouse.

4) Labels

A label containing the residue type and sequence number (e.g. TYR 322) can be added to a selected residue with the menu cascade **(sele) > L > residues**

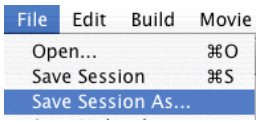
For publications or slide making, labels could be added with e.g. Adobe (PhotoShop, Illustrator) or Microsoft (PowerPoint) graphic editing software.

At this point you can explore the L menu, knowing that “clear” will remove the mess that may occur!

5) Saving a PyMol session

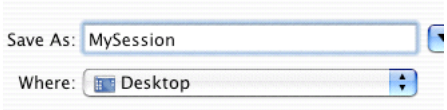
A session file is a binary file containing all the information and graphical displays currently within PyMol. It is a way to save the “current state”of the software with all that it contains. Later, the file can be opened and everything is restored.

a. Saving the session file



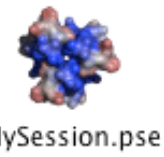
From the top menu follow the menu cascade:

File > Save Session



In the next window enter a name: **MySession** and save it on the **Desktop** to easily find it again.

The susscessful writing of the file will be reported within the ext window of the “External GUI” and a file called Mysession.pse on the desktop (.pse is added automatically).



Now quit PyMol: **File > Quit**

b. Restoring the session file

While it is possible to simply double click on the MySession.pse icon, this could create problems if there are multiple copies of PyMol within your computer and even more so if they are various versions of the software. It is therefore best to first open the PyMol software copy you want to utilize.

For this exercise: **double click** on the **Applications > Classes > MacPyMol** software to open it.

Within the PyMol top menu follow the menu cascade: **File > Open…..**
and then **select Mysession.pse** from the **Desktop**.

All the PyMol names and selected items are restored as if you had just made them, even if days have passed since you saved the file. How convenient!

Note: the file has a binary format. There is no information within the file that is useful for any other purpose. The file for this exercise should likely be slightly less than 1Mb in size.

==========

PyMol - Exercise F: A simple animation within PyMol, and for PowerPoint

There are various ways to animate and create movies within PyMol. This short exercise is meant as a simple option to both create a rotation within the PyMol Viewer, and a simple option for generating a movie suitable for PowerPoint.
At this point you should have the simple pocket surface with the carotenoid shown as stick as in the previous exercise.

The command `mset` defines a movie. Since our current level of visualization only contains one PDB file, the value of `mset` will be 1. The command `mdo` defines an action, here this action will consist of a rotation about the Y axis.

1) Automatic rotation within the Viewer

The purpose of this set of command is to create a perpetual rotation within the Viewer around the Y axis with an increment of 5 degrees.

Within the top PyMOL> command line, **type the following commands**:

```
mset 1
mdo 1: turn y,5;
```

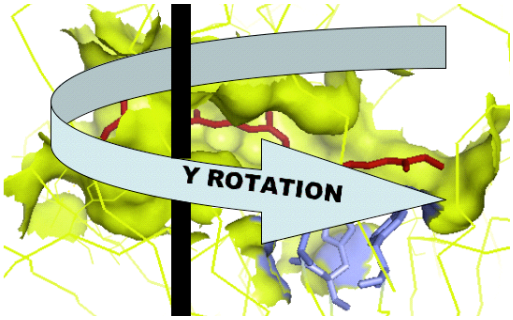
Once the commands have been typed, the “vcr” buttons at the bottom right of the “Internal GUI” become active. We only need to use the Play (triangle) and Stop (square) buttons to activate the animation.

|Stop|Play

click the **Play** button

watch the rotation happen

click the Stop button when done



Note: the mdo command is followed by a colon (:). However in previous versions of PyMol it was followed by a comma (,). The ending semi-colon (;) is necessary only if other commands are required. For example, a rotation both within the Y and X axes of 5 degrees each would be written as: `mdo 1: turn y,5; turn x,5;`

2) Creation of a movie for PowerPoint

The purpose of this short exercise is the creation of a single movie file suitable to be played independently or imported within PowerPoint.

The creation of movies within PyMol is different depending on the version and on the operating system. The MacPyMol version of PyMol contains a built-in function to tap within the QuickTimePRO engine (even if you do not have the PRO license) and can be used to create movie files. On other systems, the movie is saved as a series of images (in PNG format) which then need to be assembled manually with e.g. QuickTimePRO, Adobe Premiere or similar movie making software, meant to compile a series of images (frames) into a single file movie. This aspect of movie making as well as the indiosyncracies of PowerPoint and movie compatibility were reviewed in a previous lab or module.

Since we were using some movie/animation features above, the first command to use is `mclear` to remove functions and frames that might have been saved.

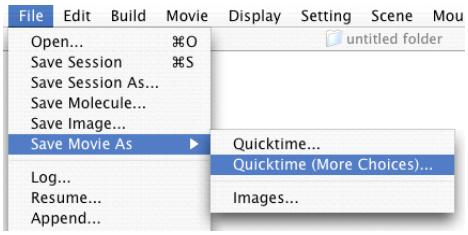
Modification of the command `mset`: since we are going to save the file into a QuickTime movie, we need to define how many frames the movie will contain. For example, to have 36 frames the command will be written as `mset 1 x36`

Modification of the `mdo` command: for each frame we need to specify that we want a rotation around the (e.g.) Y axis of (e.g) 5 degrees. However this would require to declare each of the (e.g.) 36 frames and for each frame declare the action we want to take (for example a rotation). Within PyMol the undocumented command `util.mroll(start, finish, loop-flag)` does all of this if we have declared the number of frames with `mset`. A complete 360 degrees rotation is assumed.

Within the top PyMOL> command line, **type the following commands**:

```
mclear
mset 1 x36
util.mroll(1,36,1)
```

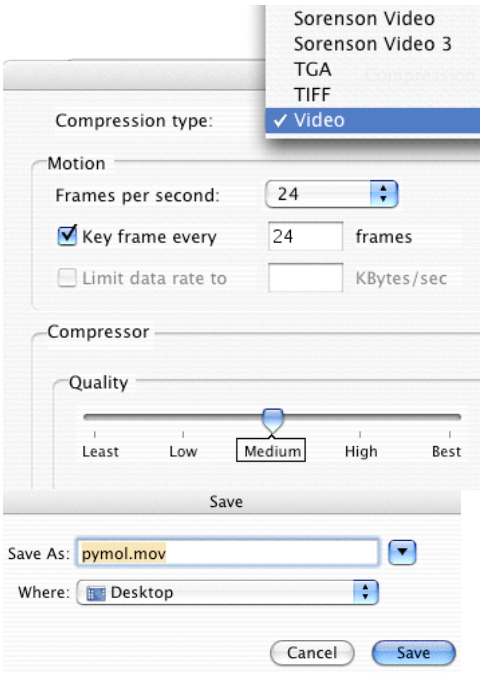
Now follow this menu cascade:



File > Save Movie As > Quicktime (More Choices)…

(Note: this menu available for native Mac version only, usually called MacPyMol)

When saving this way, the default encoding (codec) is "Video" and can work well for this exercise. Other codecs are also available depending on what is installed on your system.



Change the default 12 frames per second to **24** from the pull-down menu.

Keep the quality to medium

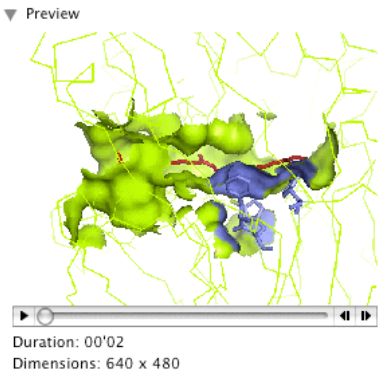
Press the Save button

The file should be about 2 Mb in size.

Save the file on the Desktop to find it easily. The default name is **pymol.mov** and can be changed here if desired.

Press the Save button

Double click the pymol.mov file and watch it.



Within the Mac finder the movie can also be previewed. Additionally it tells us the dimensions of the movie: 640x480. (Your movie will have the size of the current Viewer window.)

Indeed PyMol simply saved the 36 frames with these dimensions and used QuickTime to assemble them. On non-Mac computers, simply assemble the individual frames.

Note: to change the size of the resulting movie file or image files use the following command to set the size of the Viewer: `viewport 320,240` to create images and movies of that size (it is 4 times smaller in surface than a 640x 480 image).

3) Ray traced movies

The ray tracing option is also available when making movies. The movie making is almost identical as the previous section, but each image is ray-traced before it is saved. The movie can then be saved as a series of images or as a movie file from the File menu.

To create a ray-traced movie **type the following commands** within the top PyMOL> command line:

```
viewport 320,240
set ray_trace_frames=1
set cache_frames=0
mclear
mset 1 x36
util.mroll(1,36,1)
```

a. if you want to save the individual images use the following command:

`mpng MyMovie` (this command will create MyMovie.0001.png, MyMovie0002.png etc.)

The images can then be assembled into a movie by an independent software.

b. If you want to save the movie with QuickTime follow the same procedure as above.

Note however that the progression bar () will remain active, indicating that the ray-traced images are still available for other commands (for example the mpng save command above).

Note: ray-tracing is computationally intensive. The bigger the images (viewport size) the longer it takes. You can see which frame is being rendered by looking at the bottom right of the "Internal GUI."

==========

PyMol - Exercise G: Harnessing the power of PyMol: introducing scripts

GUI interfaces are nice, but they are slow and cumbersome. Like Rasmol, PyMol also has a line-command interface that we have already used to type specific commands. These commands are simply lines of text, and therefore can be placed sequentially within a plain text file called a script. When the script is invoked from the PyMol line command, the commands on each line of the script are executed, and the final image is shown within the Viewer.

Note: The Py in PyMol refers to Python, a graphical scripting language, and PyMol can be used as a PyMol algorithm interpreter. However, this is beyond the scope of these exercises. It is useful to know this, as you might encounter Python scripts for PyMol. The following exercises are restricted to PyMol-only commands.

1) setting up for scripts: review of files, directories and path.

On the hard drive, information is arranged in files and folders, and each operating system has its own way of organizing and calling on files. As a reminder, files contain information such as lines of text or binary numbers making up an image or a software algorithm, and are organized throughout the hard drive within directories, also called folders. When you want to invoke a

script, PyMol needs to know where it is! There are two fundamentals ways to make sure PyMol knows where the script you want to use is located:

a. Give the full path to the script such as `/Users/BNMC/Desktop/PyMol/myfirstscript.pml` on a Mac OSX or Unix/Linux system, or perhaps something like `D:\Data\My Datafiles\PMscripts\myfirstscript.pml` for Windows. These long lines give the absolute path to the location of the script, and must be present at each invocation.
b. Set the default working directory to where scripts (and PDB files) are located. We did something very similar when we gave the command `cd desktop` in exercise B, so that PyMol would automatically look onto the desktop for the necessary file(s). In some respect this is easier because once the path has been set with the `cd` command, the path does not have to be repeated again each time we want to run this or another script at the same location.

You should be (or become) familiar to the ways directories and paths are dealt with on your computer, which are essentially the Unix/Linux/OS X family or the Windows operating systems.

2) Review for Text-only files

This is rather simple but better repeated here for sake of clarity.

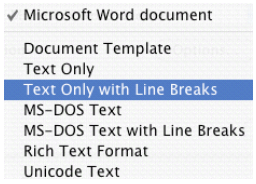
A file containing text that is **bold**, underlined, or *italicized* it NOT a plain text file (note: there are exceptions to this rule in Macintosh world not covered here).

A plain text file:

a. contains ONLY printable characters that can be typed from the simplest of keyboards (ASCII characters): letters, numbers and symbols such as !, @, and $.
b. is not specific to any particular word processor format and is displayed with the default font within any word processor (some will choose `courier` by default for this)
c. contains a "line break" or "end-of-line" code (specific to the operating system) that signifies that the line has terminated "here" and that a new line will begin "next."

Options to create text-only files:

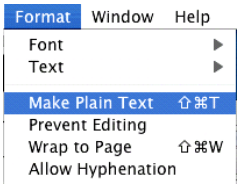
a. Microsoft Word: any file can be "Saved As…"



However MS Word is not a good choice to edit text-only files as the Save As… may need to be repeated after each modification of contents.
Note: the "Line Breaks" refer to the way lines currently wraps around in the displayed text. For scripts **Text Only** should be preferred.

b. Windows: WordPad or NotePad should be free, standard text editors within the Start > Programs > Accessories folder.

c. Mac OS X:
   i. The included Applications > TextEdit can be used only after the proper menu has been called: **Format > Make Plain Text**



ii. Applications > Classes > BBEdit is an older freeware program that offers many options and will be used in this class. (Note: newer versions are shareware).

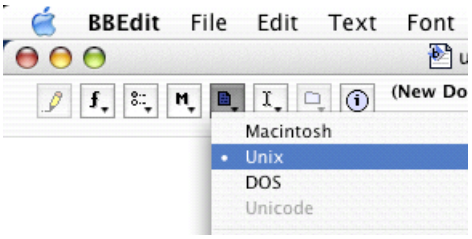
d. General Unix/Linux/MacOSX

Any file created with `vi, pico, xedit`, or a redirect command such as `cat >` is created as a plain text files.

How to deal with line breaks or end-of-lines problems:

A "hard-return" is coded within the text file. A "soft-return" is the apparent wrapping of a long text file to the next line, but in fact is a very long, single line. When saving from MSWord, one of the options is to save the file with Line Breaks that will in fact transform the soft- into hard-returns. For scripts this is not a good idea.

Hard-returns are coded as an operating system-specific code. There are three options: Macintosh, Unix and DOS. Macintosh refers to Macintosh OS Classic (through Mac OS 9.x), Unix is for Unix, Linux, *and Mac OS X*. DOS is for the old Disk Operating System that has evolved into Windows. With BBEdit, one can switch between the 3 hard-returns with a single mouse option as seen on this image:



If you import a file from another system or the web, it may be a good idea to verify it's end-of-line / hard-return status if it does not work as expected or not at all within PyMol.

3) Exercise scripts directory:

   For this exercise the script will be saved on the **Desktop**.

4) If PyMol is running, **Quit PyMol** and **restart** it again, to have a fresh start. If you are starting here to learn about scripts, review how to **find** and **start PyMol** in exercise B.

5) Use TextEdit (as detailed above) or BBEdit to create a new script that we shall call **abc.pml**. **Save abc.pml** on the **desktop**.

Within abc.pml file type the following lines (comment lines which start with #. You can omit typing anything after the comment sign # for these exercises. However, placing comments within your script is a good practice for your future reference or for those with whom you share your scripts with, and can save hours of frustrations!)

```
# this is my first script:
load 2BIW.pdb2
bg_color white
hide lines
show cartoon
```

```
color red, ss h
color cyan, ss s
color yellow, not ss h and not ss s
```

Note the use of the comma (,) after the name of the color.
`ss` signifies secondary structure
`h` signifies helix. (Note that the word “helix” would not be recognized.)
`s` signifies sheet. (Note that the word “sheet” would not be recognized.)
`not h and not s` are be the loops or turns.

6) In your new PyMol session, set the default directory by typing the `cd Desktop` command within the top line command. Then verify that the path is correct with the command `pwd` (present working directory):

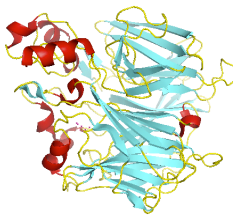
cd Desktop
pwd

7) Invoking a script: the @ command

Rule: a script is invoked with the @ command after its name within the line command: example: @abc.pml

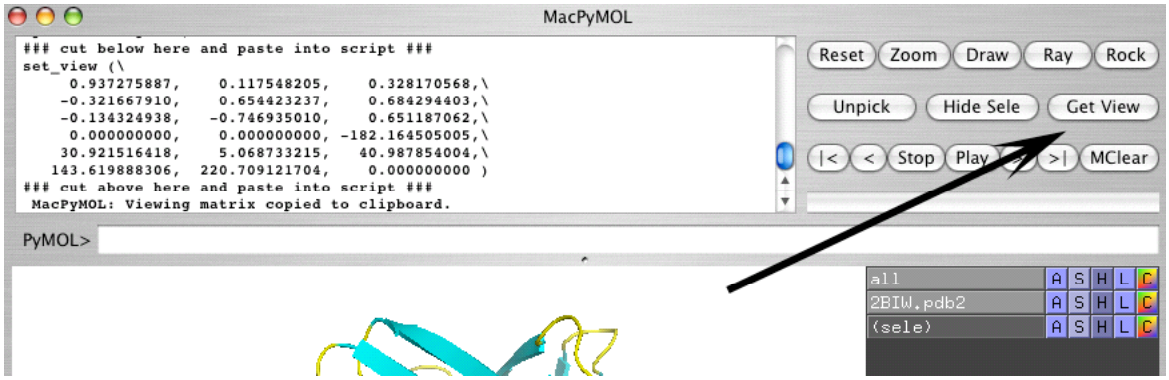
Type **@abc.pml** within the line command.

Note: either the top (External GUI) or bottom (Internal GUI) line command can be used.



Activating the script will create the image as you could have done it by either typing each of the script commands within the line command, or by using some of the mouse options as we have done before. The advantage of scripts is that they are an easy way to create images in a reproducible fashion, but most importantly, it can be an easy way to remember how the images were created.

This image was created with the default orientation that the molecule takes when it is opened. But very often, one important aspect of an image is the orientation of the molecule. PyMol offers a very nice way to return to an existing orientation and build it within a script: the “Get View” button, located at the top right of the “External GUI.”



Rotate the molecule in an orientation you like
Click on Get View button
The rotation matrix is shown on the PyMol text window, but is also placed within the clipboard.
Paste the matrix at the end of the **abc.pml** script
Add # in front of the load command to read: **`# load 2BIW.pdb2`**
Rotate the molecule again in another orientation or you can just type **Reset** instead to go back to the default opening view.
Rerun the script by typing **@abc.pml** on the line command.

The PDB file will not be reloaded since we commented out the line with the load command (#). The formatting and coloring of the protein will occur unnoticed since they are already in the cartoon state colored by our chosen secondary structure colors. But the orientation of the molecule will be restored.

The PyMol way: this is a nice way to work with PyMol. Simply create and modify a script as you go, commenting out the loading of the PDB file(s) after the first run. It is a preferred way of working with PyMol that leaves the legacy of a finished script with complex commands. Therefore you can create your own repository of complex scripts, as a way to safeguard these commands for future use and reference. In addition, you can run the script again and again as you build it, line by line.

==========

PyMol - Exercise H: Select command, parameters, scripting, and subsets.

Understanding the content of this exercise can be beneficial to your PyMol skills!

The PyMol "select" command has the special property to create atom selections with names appearing within the Names Panel. This is a nice feature, but some confusion can arise depending on the words that are used. For example, in the abc.pml script we have just used the command "`color red, ss h`" to colors red only the secondary structures (ss) that are in helix (h) form. However the command "`color red, ss helix`" would not at this time have any effect, and PyMol might give us an error message. The reason is that "helix" has no meaning at this point. Interestingly this command CAN work if "helix" is defined first. "helix" can be defined when an atom selection is created with the "select" command. Any subsequent command can then contain the chosen word, and the commands will only apply to that atom

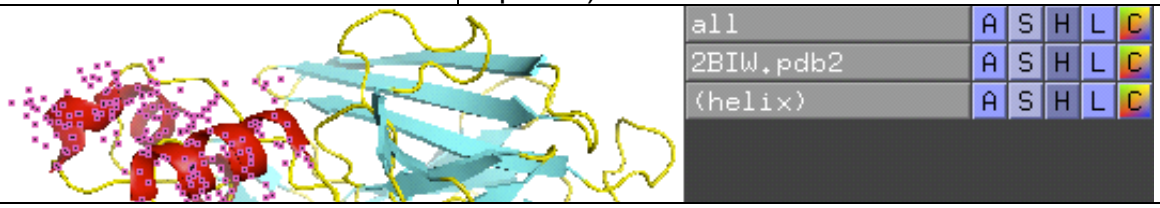
selection, which can be a subset of a larger selection. When switching from Rasmol this can be confusing, as the command “select helix” is a valid command on the Rasmol command line.

1) Defining by selection: an illustrated example

First let’s assume that you just ran the abc.pml script above on a fresh start of PyMol. If not, simply quit and restart PyMol, then in the line-command area type `cd desktop`, press return and then type `@abc.pml` to be in the same state as if you just had done the previous exercise. In the script the helices are made red. Here we will make them green.

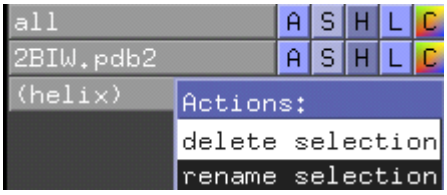
Within the line command type: **color green, helix** Note the error echoed within the PyMol text window	**PyMOL>color green, helix** **Selector-Error: Invalid Selection Name.** **helix<--**

Within the line command type: **select helix, ss h**	Two things will happen: a new name “(helix)” will appear within the Names Panel, and all the atoms participating in a helix structure will be selected within the Viewer (pink squares):



Within the line command type: **color green, helix**	Note: this time it worked and the helices were colored green.

Note: The (helix) selection can be deleted with the command: **delete helix**, however the name within the Names Panel only disappears when the mouse is clicked within it. Alternatively the seletion can be deleted with the “A” Actions menu “delete selection.” The next menu allows to rename the selection.



Important lesson: the word “helix” here could have been any other word such as “MyHelixSelection.” Using a “My” prefix before some of the selections might help avoid confusing words that are commands and words that are the names of created selections. This can be important when writing or deciphering other people’s scripts!

…any word can do…

2) PyMol Selectors: keywords to make atom selections

Reminder: PDB (and other) 3D coordinate files are structured with certain fields arranged in columns. PyMol can detect and interpret some of these fields. For example, in the ATOM records, the “atom name column” or “atom name field” is the column where the atoms are given a name, such as C, CA, CB, CG, or CD for example, corresponding to the asymmetric carbon, the alpha-, beta-, gamma- and delta- carbons of an amino acid respectively. Similarly, there will be columns with the name of amino acids, and clearly more than one line of ATOM records make up for all the atoms of one amino acid. Finally, a chain and sequence number are also in column.

```
ATOM   3752  N   GLN B 489      44.222   1.408  52.889  1.00 68.12           N
ATOM   3753  CA  GLN B 489      45.155   0.463  53.497  1.00 70.72           C
ATOM   3754  C   GLN B 489      45.787   1.050  54.759  1.00 71.44           C
ATOM   3755  O   GLN B 489      46.202   2.212  54.757  1.00 71.87           O
ATOM   3756  CB  GLN B 489      46.237   0.102  52.476  1.00 70.38           C
ATOM   3757  CG  GLN B 489      46.909  -1.223  52.738  1.00 72.85           C
ATOM   3758  CD  GLN B 489      47.621  -1.821  51.503  1.00 73.43           C
ATOM   3759  OE1 GLN B 489      47.815  -1.162  50.482  1.00 75.73           O
ATOM   3760  NE2 GLN B 489      48.014  -3.086  51.614  1.00 76.50           N
```

The complete list of selectors can be found on the PyMol manual (`http://pymol.sourceforge.net/html/index.html`) or it's echo as the wikipedia version (`http://www.pymolwiki.org/index.php/`)

In summary, here are the most relevant selectors for working with macromolecules (proteins and nucleic acids)

(Each definition below is followed by an example. Remember: as above, the word after the word "select" is of your own creation. Adding "My" to the word may be a good reminder of that.)

Matching Property Selector	***Identifier and Example***
symbol	*chemical-symbol-list* list of 1- or 2-letter chemical symols from the periodic table **PyMol>** `select Mypolars , symbol o+n`
name	*atom-name-list* list of up to 4-letter codes for atoms in proteins or nucleic acids **PyMol>** `select Mycarbons , name ca+cb+cg+cd`
resn	*residue-name-list* list of 3-letter codes for amino acids **PyMol>** `select Myaas , resn asp+glu+asn+gln` list of up to 2-letter codes for nucleic acids **PyMol>** `select Mybases , resn a+g`

resi	*residue-identifier-list* list of up to 4-digit residue numbers **PyMol>** `select Myresidues , resi 1+2+20+8590` *Residue-identifier-range* **PyMol>** `select MyNterm , resi 1-25`
chain	*chain-identifier-list* list of single letter (rarely numbers) of the chain **PyMol>** `select MyChain , chain a`
ss	*Secondary-structure-type* list of single letters **PyMol>** `select Myalphas , ss h+s+l+""`

3) Putting it together: a fancier script

Lets make a last script to wrap it up. **Create** a script file called **abc2.pml** with BBEdit or other means, and save it on the desktop.

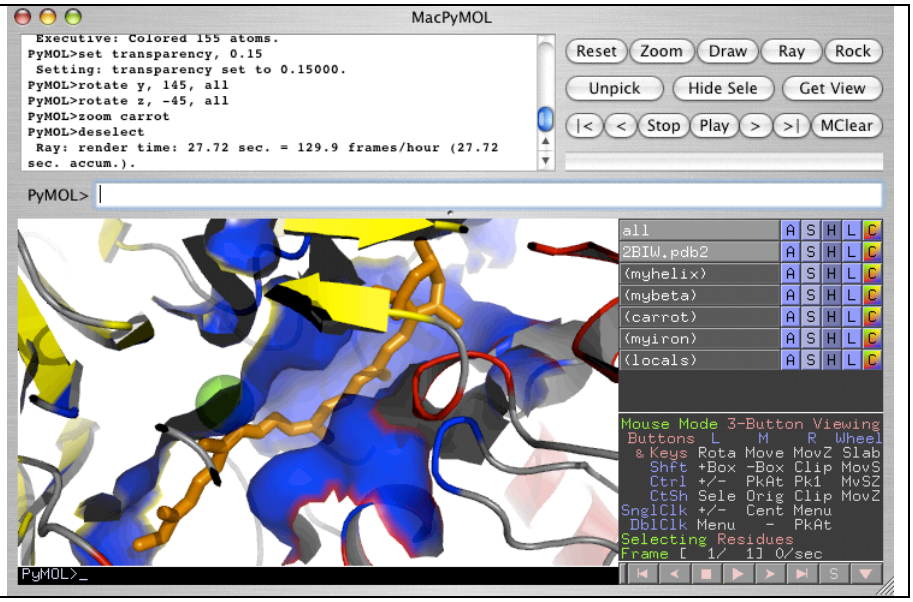
Within abc2.pml type the following script. At this point you can omit typing the commented (#) lines, but remember that placing comments within your scripts (annotation) can save (lots of) time and remove headaches in the future…

```
# my fancy script (assumes cd desktop for these exercises)
# first lets reset everything without need to quit PyMol
delete all
# set some parameters, even if some were done in previous
# white background:
bg_color white
# antialias: 1 for smooth images, 0 for jagged
set antialias = 1
# now reload the PDB file
load 2BIW.pdb2
# hide everything (all lines and so on)
hide everything
# show cartoon ribbon for the protein
show cartoon
# keep standard helix, strand and loops representations.
# other options would be: cartoon loop, cartoon rect,
# cartoon oval, and cartoon tube.
cartoon automatic
# make the helices with edges as in Molcript.
# 1 is on, 0 is off
set cartoon_fancy_helices=1
# color the inside of the helices in gray
set cartoon_highlight_color, gray
# color everything gray, and change other things later
color gray
# make selection of alpha helices and color it
select myhelix, ss h
color red, myhelix
# make selection of beta sheets and color it
select mybeta, ss s
color yellow, mybeta
# The PDB file contains a carotenoid ligand named 3ON
# it carrot when selecting it, show it as stick,
# and color it orange
select carrot, resn 3ON
show stick, carrot
color orange, carrot
# change the stick thickness from the default 0.25 to 0.40
set stick_radius = 0.40
# Inspection of the PDB file reveals there is also an iron
# select it and name it myiron, color it green and show it
# as a sphere. Then increase the sphere look quality.
select myiron, symbol FE
show sphere, myiron
color green, myiron
set sphere_quality = 8
# Select all full amino acids within 4 angstroms of the ligand
# and call this selection locals. Show them as a blue surface.
select locals, byres (chain B and carrot) around 4
show surface, locals
color blue, locals
# make the surface 15% transparent (85% opaque)
set transparency, 0.15
# Rotate about the Y axis to see the inside of the pocket
rotate y, 145, all
```

```
rotate z, -45, all
# zoom in on the ligand
zoom carrot
# make sure nothing is selected to avoid the little pink
# squares.
deselect
# end of this script – you could uncomment the next line to
# make a fancy image:
# ray
```

Within the PyMol line command **type @abc2.pml** to activate the script.

Your screen should be similar to this image, which is the ray-traced version.



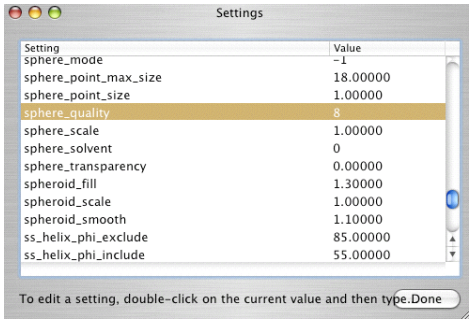
4) Where to go from here ?

Writing a script may seem tedious at first, but once it's written it can be saved, rerun, upgraded, placed on the web, and exchanged with other people.

Phylosophy: *Writing scripts is the PyMol way!*

Where do I find the relevant variables such as sphere_quality?

The top menu cascade **Setting > Edit All…** opens a window that lists all the variables and their current value. The values can be changed interactively within this window, or changed in the script as we have just done. This image shows the current value for "sphere_quality" after running the script: it has now the value 8, given by the script.



Learning by example: this is what we have done so far, at a slow pace. However there are many web sites that offer PyMol help and scripts, often annotated with the # comments to explain what is inside. Studying other's scripts is a way to learn some fancy options. Here are a few web sites to study at your own pace:

• Creating a eye-catching figure with Pymol: (a must see!)
http://www.doe-mbi.ucla.edu/~sawaya/tutorials/Graphics/pymol.html

• Image gallery with corresponding PyMol scripts
http://www.chem.ucsb.edu/~molvisual/dna_biochem.html

• PyMol tips/scripts and tutorial:
http://www.rubor.de/bioinf/pymol_tips.html
http://www.rubor.de/bioinf/pymol_tutorial.html

• Introduction to Using PyMOL
http://pymol.sourceforge.net/carly/

• Visualizing Protein Structures - A Practical Introduction to PyMOL
http://www.ii.uib.no/~pal/teaching/mol305/files/pymol_intro.pdf

• Brief PyMOL tutorial:
http://www.mrc-lmb.cam.ac.uk/rlw/text/pymol_tutorial.htm

• Closed Configuration Of Dna Polymerase I
http://www.doe-mbi.ucla.edu/CHEM125/tutorial_1lv5.html

• PyMol & Movies
http://davapc1.bioch.dundee.ac.uk/teach/pymol/index.htm

• My PyMOL Script repository (*for advanced use*, including Python langage)
http://adelie.biochem.queensu.ca/~rlc/work/pymol/

End of PyMol tutorial. Close or Quit all programs before leaving the classroom.

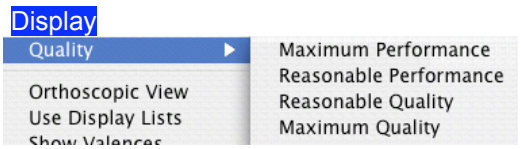
==========

PyMol Q&A

(does not have to be completed in class)

1) How can I work with PyMol on a very slow (laptop) computer?

The Display > Quality menu cascade allows you to adapt to the speed of you machine CPU depending on the quality of the image displayed in the Viewer. This option MAY affect the result of ray-traced images.



2) Labels are created the same color as the atom. How can I change only the label color without changing the display of the atom or molecule?

You can change the label color with the following line command: **set label_color, yellow, sele**
• Yellow can be changed for other colors (can be chosen from the names within the C menu)
• Sele can be either the current selection, or can be switched the the name of the object you are working on.
• note the use of the comma (,) within the line command

3) How can I make a new object?

The current selection (sele) as well as other subsets (e.g. (MyHelix)) are shown within parentheses and represent subsets of the current PDB file we are working on (e.g. 2BIW2.pdb2). Any selection can be copied into an independent object with the "**A**" **Actions: create object** menu cascade. This will create a new entry within the "Names Panel" without the parentheses that becomes independent of the original PDB file.

For example, the ligand can be made into a separate entry:

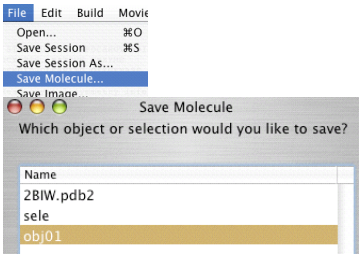
click on the **ligand** (becomes the current (sele))
click on the **A** on the (sele) line and choose: **create object** (a new entry called obj01 appears within Names Panel, and an automatic zooming occurs)
click on the **S** on the (obj01) line to change the object appearance, e.g. as stick
If you make obj01 invisible by clickin on the obj01 icon, the ligand appears still here as it is also part of the 2BIW.pdb2 file.

4) Why can't I make a surface for the ligand itself?

Surfaces are reserved for ATOM records only. Typically ligands are always HETATM records (but authors of the PDB file may choose otherwise, hence always check within the file with a word processor.)

One way to circumvent this is to make the ligand into an independent object (see question above about creating object), save it from PyMol into a text file, replace the ATOM word with HETATM and reopen the file with PyMol:

Assuming you have created obj01 as described above, use the menu cascade **File > Save Molecule….**

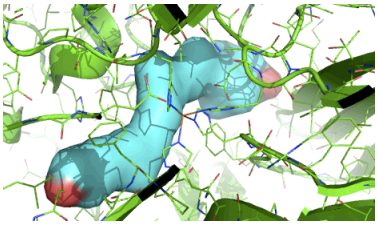


Choose **obj01** within the Save Molecule panel.

Choose a file name and save the file e.g. **obj01.pdb**. The file will be a PDB file with only the object 3D data and contains HETATM and CONECT records.

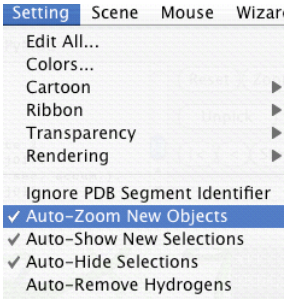
Using a word processor open the PDB file and change replace "HETATM" with "ATOM " (ATOM with 2 blank space to respect the column positions!!!!!)
Delete the current obj01 with the menu cascade: Names Panel: **obj01 > A > delete object**

Open the PDB file **obj01.pdb** with PyMol (menu cascade File> Open or use the "load" line command)
A new obj01 now appears within the Names Panel, and the **S > surface** menu is now working for this object!


ray-traced example

5) How can I turn off the automatic zooming and clipping when I make a new object?

When we created object obj01 from (sele) above (see question about creating objects), the default action of PyMol was to automatically zoom to the new object (in that case the ligand) and reduce the clipping planes. That can be annoying in some cases, but could always be reversed with the line command "reset" or with the menu Display > Clip > None or with the Display > Zoom menu options.



To turn off this option is turned off by un-checking the menu item:
Setting > Auto-Zoom New Objects

6) Can I superimpose in 3D two similar structures?

PyMol will have no problem aligning 2 similar structures. PyMol firsts creates a sequence alignment the then tries to align the structures accordingly. If 2 proteins are named struct1 and struct2 within the Names Panel, the simple line command will align them: **`align struct1, struct2`** (note the use of the comma after struct1)

For more complex alignment questions, see the following web site:
`http://adelie.biochem.queensu.ca/~rlc/work/teaching/BCHM823/pymol/alignment/`

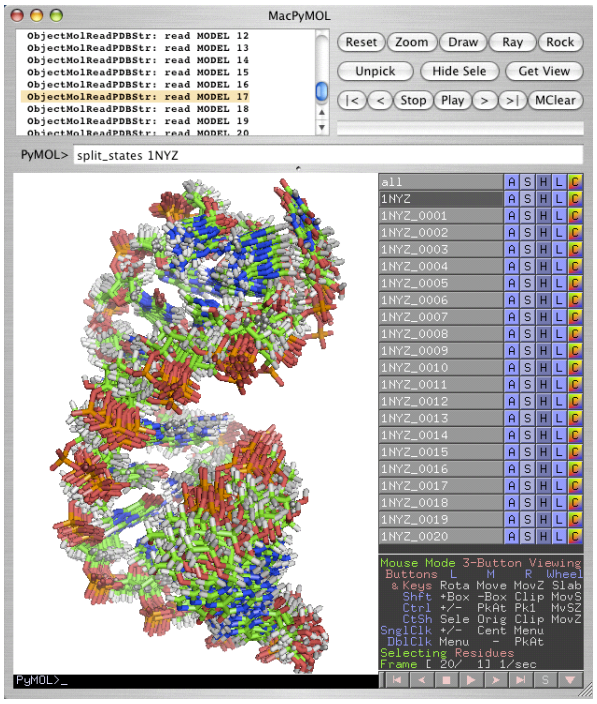
7) Where is the HELP command?

Some on-line web content can be called from the Help menu in PyMol.

PyMol also offers an internal summary of all the command derived from the manual. These are called from the line command **`help xxx`** where xxx is the name of the command, e.g. `help align` or `help color`.

8) I have an NMR file called 1NYZ with 20 structures… How can I see or work on a specific state?

If you open the PDB file with a word processor, you can see that each structural state starts with the `MODEL` keyword and is separated from the next with the `ENDMDL` keyword. PyMol will read all molecule states, in this example 20. This is also echoed within the text panel:
e.g. `ObjectMolReadPDBStr: read MODEL 17`



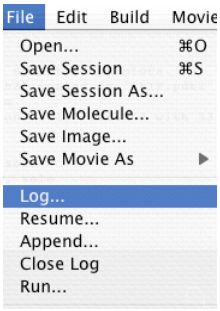
When the file is opened, a single name (e.g. 1NYZ) will be shown within the Names Panel and a single molecule (state) is shown on the Viewer. At this point all actions will apply to all structures at the same time, e.g. displaying as stick model
Names Panel : **1NYZ > S > stick**
If you click on the VCR triangular play button at the very bottom right, you will see an animation showing each state one after the other. You can also save this animation as a movie with the **File > Save Movie As…** menu option.

The following line-command will separate all the structure and make each one of them a separate object assuming the name of the object is 1NYZ: **`split_states 1NYZ`**

When the command is given, 20 new objects are created, labeled from 1NYZ_0001 to 1NYZ_0020. These can be worked on individually with the help of the ASHLC menus within the Names Panel or made invisible by simply clicking on its name in the Names Panel.

9) How can I keep a log of the line commands I type?

All commands are echoed on the Text panel together with any additional text triggered by the command. While it is possible to use Copy/Paste from this panel, it is much more convenient to have a separate file logging all the typed commands only. Furthermore, this log file is *de facto* a script!



The logging can start after the **File > Log…** command is requested and a file name is given, e.g. **test.log**. The file can be saved anywhere on the hard drive, for example the desktop. The menus allow to close the current log, resume logging or append to any previously created plain text file.

You can try to create a log file while typing the commands on the right hand side on the PyMol line command.

```
cd desktop
load 2BIW.pdb2
select hetatm
show stick
hide stick
show stick, sele
color yellow, sele
```

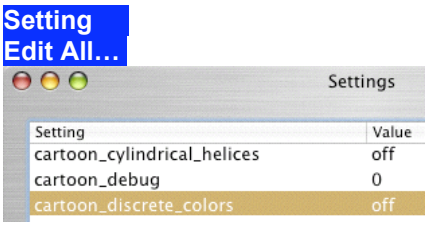
You can then quit and re-open PyMol and run this test.log file with the following command: **cd desktop; @test.log**

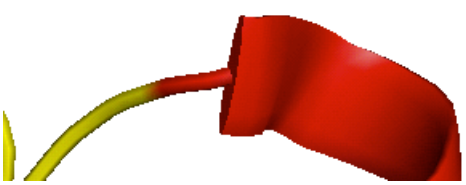
Note: if you want to run it from the **File > Run…** menu, you may need to **rename** the file with a .pml file extention e.g. **test.log.pml** otherwise the file cannot be opened.

Practical note: previously typed command on the line-command can be recalled by using the “up-arrow” key.

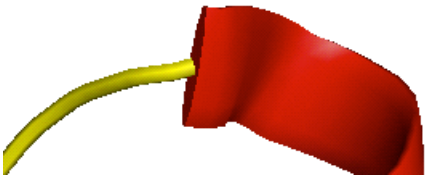
10) When I make a cartoon helix, part of the helix color bleeds on to the turns, how can I fix that?

Turning the variable cartoon_discrete_colors on makes the helix color ends abruptly at the end of the helix. The default value is off. The change can be done manully with the menu cascade Setting > Edit All… or can be given as a typed command: **set cartoon_discrete_colors = on** (space on either side of the = sign is optional)





set cartoon_discrete_colors = off



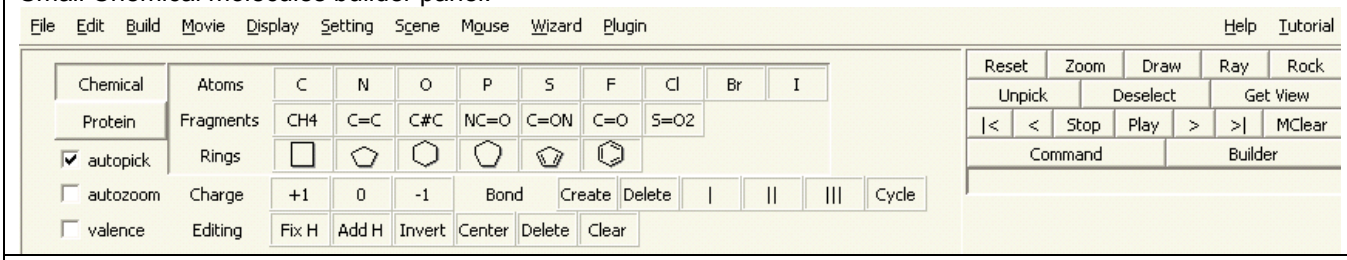
set cartoon_discrete_colors = on

11) I want to build a model helix from a sequence, how can I do that?

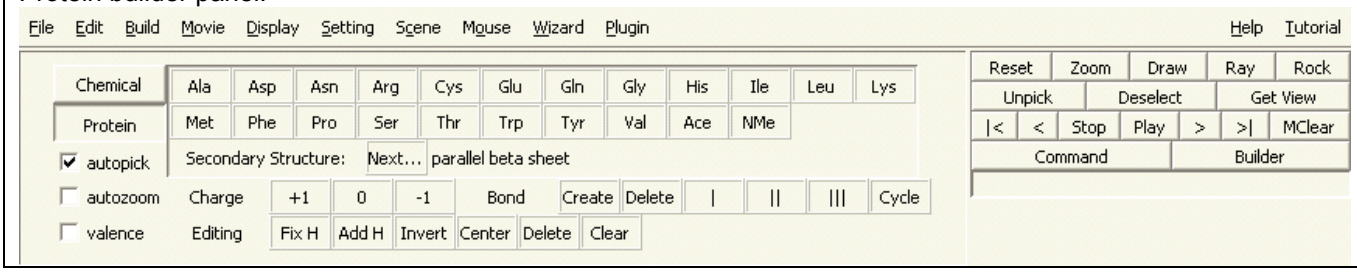
PyMol offers modeling options for small molecules and proteins. For example you can build an alpha helix from scratch with a specific sequence. The “builder” interface to accomplish this is different depending on the operating system. On Windows and X11 versions of PyMol, the “External GUI” has an extra set of buttons to build by clicking on the various components to be assembled such as amino acids. The MacPyMol version has the same capability but instead of clicking icons menu items are chosen.

Windows builder interface:

Small Chemical molecules builder panel:

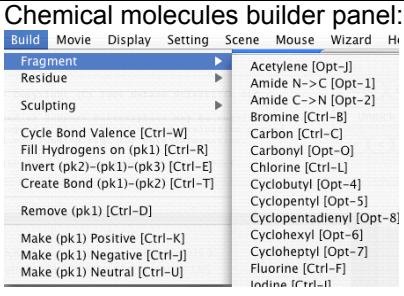


Protein builder panel:

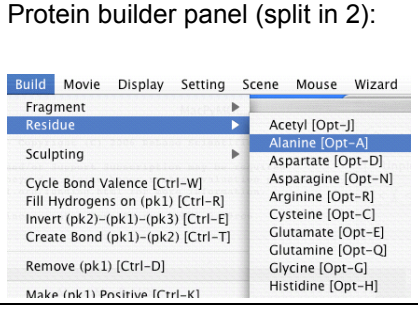


MacPyMol builder interface is via the Build menus:

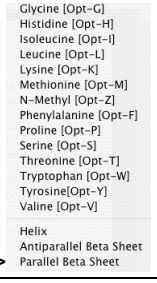
Chemical molecules builder panel:



Protein builder panel (split in 2):

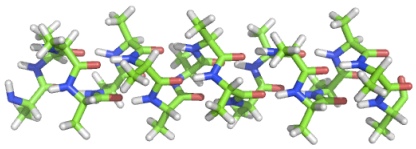
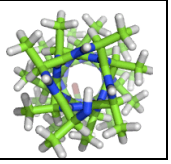


continued…>



Steps to build a model helix:

a. (Windows: click on “Builder” button at right and then on “Protein button at left.)
b. Choose the helix form (bottom of menu Build>Residue. (Windows: click on “Next”)
c. Select the sequence you want to build from the Build (Windows: clicking residue name)
d. The model will be built withint the Viewer.
e. Save the molecule with the File > Save Molecule menu. The name of the molecule will be the name of the first amino acid used e.g. ala.pdb

The alpha helix is created as a “perfect” model and contains all hydrogen atoms.		

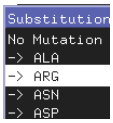
12) How can I mutate one side-chain from an existing structure?

The menu cascade **Wizard > Mutagenesis** opens a new panel below the Names Panel and above the mouse control reminder. Directions will be prompted with text overlaid on the Viewer:
“Pick a residue” and “Select a conformational state, or pick a new residue…”

Steps to mutate one amino acid on a protein structure:

a. Open from the menu **Wizard > Mutagenesis**

b. **Click** on the **residue** you wish to mutate
c. **Select** a conformational state in the new mutageneis panel menu (bottom right)
(options are backbone dependent or independent)
d. **Click** on the **No Mutation** button and select a new amino acid (e.g. ARG)
e. **Click Apply**
f. **Repeat** process for mutating more residues
g. **Click Done** when finished with the Mutagenesis Wizard.



13) I’d like to try various way of orienting, drawing and coloring a structure. How can I compare them?

While PyMol cannot show all the versions at the same time, the **Scene** menu has very nice feature to save the structure in various states of representations and toggle between them. The transition from one scene to the next creates also a beautiful screen animation.

Simple example with file 2BIW.pdb2:

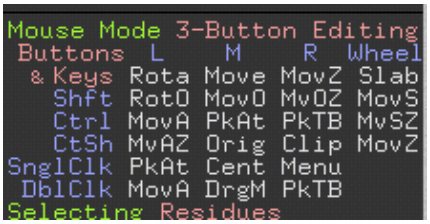
a. **Create** a first version of a representation: show as cartoon (**S > cartoon**) and color by secondary structure (**C > by ss > pick-a-color-scheme**). Now orient the molecule with the mouse in a way you like.
b. **Store** this view in the F1 keyboard key with the menu cascade **Scene > Store > F1**
c. **Create** a second presentation for example change the cartoon color (**C > spectrum**) and rotate the molecule in a different orientation. For a more stunning effect you can zoom in.
d. **Store** this new view into the F2 key: **Scene > Store > F2**
e. Now you can recall the previous scene with the menu cascade **Scene > Recall > F1** and you should witness a beautiful animation and color change while the scene is changing.

14) How can I rotate 2 molecules relative to each other?

There are 2 ways to accomplish this task: with the mouse or with line command.

Mouse method:

a. **Switch to 3-botton mouse editing** with the Mouse menu The table at bottom right summarizes the possible movements. The mouse movement ending with "O" are related to object rotations: Shift+Left button = RotO (rotate object). Shift+Middle button= MovO (move object), Shift+ Right button=MvOZ (translate the object along the Z axis).
b. **Click on the object (molecule)** you want to rotate or translate.



c. **Apply the proper mouse/keystroke** combination for rotation, translation, or moving along Z (toward or away from you)

Line command method: learning by example.

The line command method is shown here by an example from the PyMol author.

1FJ1.pdb contains an antigenic fragment and its bound antibody in a single PDB file.

Once the PDB file is read in with the `load` command, the relevant protein chains are copied into independent objects and given a name (`anti` and `fab`.)

Some selection and coloring is done to help visualize.

`inter` is the name given to the interaction area. `byres` helps select complete rather than partial amino acids.
When a line is long in a script, the charater \ indicates that the command continues on the next line.

The command `orient` orients the molecule along the XYZ axes.
The command `origin` places the center of rotation on the deginated object, which is then rotated around the y axis with the command `rotate`.

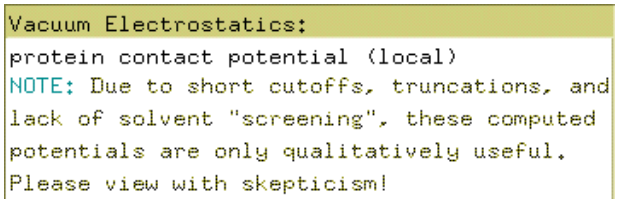
```
load 1FJ1.pdb

# split PDB file

create anti=(chain F)
create fab=(chain A,B)

# delete original object
delete 1FJ1

# color objects
color green,fab
color pink,anti

# color interface
select inter = (byres ((fab within 5 of anti)\
   or (anti within 5 of fab)))

color yellow,inter

# splay apart
orient
origin fab
rotate y,60,fab
origin anti
rotate y,-60, anti

# zoom interface region
zoom inter
show sph,inter
disable inter
```

Provided the PDB file 1FJ1.pdb (http://www.rcsb.org) is available, this script called split.pml can be run from the line command with @split.pml or called from the File>Run… menu cascade IF the filename extension is .pml.

15) How do I make an electrostatic potential surface map?

The most accurate potential maps can be calculated by external software (Grasp, Delphi, APBS, MEAD) and displayed in PyMol. Search the web or the Pymol usergroup archive for details on using external software to create a map `http://sourceforge.net/mailarchive/forum.php?forum_id=60` or see this posting from PyMol author: `http://imsb.au.dk/pipermail/o-info/2004-December/007601.htm`

PyMol offers an approximate map nicknamed "*charge-smoothed*" surface representing approximate charge distribution on the protein surface. Assuming that you have opened PDB file 2BIW.pdb2, the potential surface is calculated and displayed with the following menu cascade on the Names Panel:

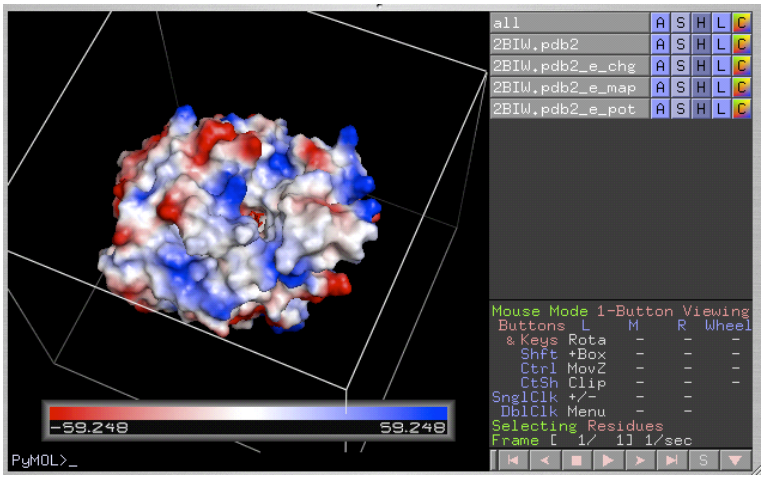


2BIW.pdb2 > A > generate > vacuum electrostatics > protein contact potential (local)

The process creates 3 new entries within the Names Panel: 2BIW.pdb2_e_chg, 2BIW.pdb2_e_map, and 2BIW.pdb2_e_pot (2BIW.pdb2 is subsequently abbreviated X below.)

X_e_chg is an object containing the surface, colored red/white/blue and is an approximate map.
X_e_map is usually not shown (click on name to show) and displays the volume boundaries as e.g. a big cube.

X_e_pot is the object representing the color ramp and value at the bottom of the display.



a. adjusting the value range and color strength:

If two proteins are present, the process has to be applied to each individual protein, and the same _e_ objects are created for each one. The values within the X_e_pot color ramp may be different for each protein. Therefore it is useful to know how to change the value within. Smaller numbers will increase the blue and red strength and contrast within the blend of white surface, while larger numbers will dim the colors. To change the color range within the ramp: (example): **ramp_new 2BIW.pdb2_e_pot , 2BIW.pdb2_e_map, [-100,0,100]**

b. Adjusting the display colors:

The default colors are red/white/blue, hence the above command could also be rewritten to change ramp values and specify colors:

ramp_new 2BIW.pdb2_e_pot , 2BIW.pdb2_e_map, [-100,0,100] , [red, white, blue]
or
ramp_new 2BIW.pdb2_e_pot , 2BIW.pdb2_e_map, [-100,0,100] , [[1,0,0], [1,1,1] , [0,0,1]]

where [[1,0,0], [1,1,1] , [0,0,1]] represents the [R, G, B] (red/green/blue channels) values for each of the 3 displayed colors. To change the displayed color, simply change the definition of the colors. For example, the values **[[1,1,0], [1,1,1] , [0,1,1]]** would create a ramp as yellow/white/cyan.

Note: an alphabetical list of all PyMol-defined color names can be found under the top menu “Setting > Colors…” Within that window the colors can also be edited.
The **C** menu within the Names Panel offers a list of color names grouped by tint and can serve as a preview.

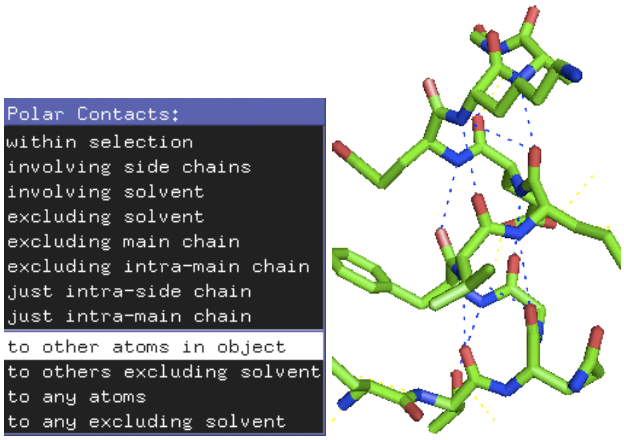
c. Default display:

Note that when the calculation first takes place, the surface is shown but the original PDB file is hidden. In the case of 2BIW it means that the carotenoïd becomes invisible. To restore it, simply click on button 2BIW.pdb2 within the Names Panel: this will restore the display of the protein and ligand as lines or sticks. Since the protein is under the surface it will remain invisible (but some odd side chains might stick out of the surfaces.) To make more complex images it may be useful to create a new object containing the ligand alone (see questions above: *How can I make a new object?* and *Why can’t I make a surface for the ligand itself?*)

16) How can I display hydrogen bonds?

See also “Polar Contacts” within the following web page:
`http://www.pymolwiki.org/index.php/Displaying_Biochemical_Properties`

“Polar Contacts” can be displayed with the menu **A > find > polar contacts** and selecting from one of the many submenus.



“within selection” and “to other atoms in object” are particularly useful.

Example to display the hydrogen bonds within the alpha helix consisting of residues 94 to 105 in 2BIW.pdb2:

First create object helix-1 containing the helix residues with the line command:

select helix-1, resi 94-105

Then within the Names panel follow the menu cascade:

helix-1 > A > find > polar contacts > within selection

A new objects 2BIW.pdb2_polar_conts will be created. The yellow default color for the dashed line is easily changed with the **C** menu for this object. For example a darker color can be chosen for display against a white background.

Note: choosing **to other atoms in object** would show additional hydrogen bonds of the helix side-chains to other parts of the protein.

Note: for more complex issues regarding hydrogen bonds or adding hydrogen atoms refer to the web link above.

17) END of Q&A.

==========

Reply to a student's Question: Is there a way in pymol to make a quick time movie that rotates in the x, y, and z direction at the same time?

For a rotation within PyMol it's easy, and the command is:

```
mdo 1: turn y,5; turn x,5; turn z,5;
```

for 5 degrees rotations on each angle

I agree with you that the PyMol movie making is not as straight forward as it sounds for more elaborate movies.

First, we shall assume that you are using a PDB file with one "state." In other words you are not using an NMR file that contains multiple structures (refer to the addendum I gave in class if that confuses you, relative to NMR file 1NYZ.pdb.)

For movies in PyMol, the first thing to understand is that you have to specify the number of frames at the begining of the movie commands. THis is done with `mset`, but `mset` is defined in different ways depending on what you are doing. Assuming you are using a one state PDB file, you need to state how many frames the movie is going to be:

```
mset 1 x30    # creates a 30 frame movie consisting of state 1 played 30 times.
```

you can review more `mset` options on the following link, but I think that you should not need any of the other options for your rotations.

```
http://pymol.sourceforge.net/newman/user/S0300movies.html
```

After some research on the net here are some solutions I have found to your question about rotating about the 3 axes at the same time. The answers were not as easy to find and I am not sure if they completely answser what you want.

1) the easiest:
==========

In some way I assume that you want to make a "nice" rotation that is not as boring as the simple Y rotation. In this case the command `movie.nutate` would be helpful and most likely what you are looking for. With a small angle it resembles the default movement "lemniscate" in VMD movie maker. Here is an example: (# is the comment symbol and everything is ignore by PyMol after it.)

```
mclear      # clears out previous movies you might have tried
mset 1 x72  # uses state 1 for a 72 frames movie
movie.nutate 1,72,360    # starts with frame 1 until frame 72
                         # for a 360 rotation around all axes.
```

Of course the movie looks better with more frames and more angles:

```
mclear
mset 1 x360
movie.nutate 1,360,360
```

`movie.nutate` belongs to the built-in PyMol Python script movie.py and has more options as I have found at the following site:
http://www.koders.com/python/fid54F75215FAB3B81AE8E0824786C2FD828C60845C.aspx

```
def nutate(first,last,angle=30,phase=0,loop=1,shift=math.pi/2.0,factor=0.01)
```

so you can see that the first number is the first frame that will be affected. You could decide to start moving only after frame 10 for example. The second number is the last frame that is going to be moving with the rotations. The third number is the rotation desired. If you use e.g. 20 instead of 360 you will have the "Leminscate" effect seen in VMD. The next numbers can be played with to test their effect(s) but are not mandatory to the command.

NOTE: the use of `movie.nutate` was suggested to a similar question by a user to the PyMol author Warren Delano. See:
http://sourceforge.net/mailarchive/message.php?msg_id=2078718

2) Using cmd.mdo
==========

This one is inspired from the following answer on the sourceforge archive:
http://sourceforge.net/mailarchive/message.php?msg_id=9881931

```
mclear
mset 1 x360
for a in range(1,360): cmd.mdo(a,"rotate x,5;rotate y,5;rotate z,5")
```
You will note that the rotation is not as smooth as the movie.nutate option, even with a rotation of 1 rather than 5.

3) Related question: rotating around Y and then X....
==========

Answer from link:
http://sourceforge.net/mailarchive/message.php?msg_id=2371062

```
mset 1 x360
movie.roll 1,180,1,axis=y
movie.roll 181,360,1,axis=x
```

Frames 1 to 180 are for Y rotation and frames 181 to 360 are for X rotation. You could decompose the frames in such a way to add a Z rotation as well:

```
mset 1 x360
movie.roll 1,120,1,axis=y
movie.roll 121,240,1,axis=x
movie.roll 241,360,1,axis=z
```

However there is another command that can do all that in one line: movie.tdroll:

```
AUTHOR
Byron DeLaBarre
USAGE
movie.tdroll(rangx,rangey,rangez,skip=1,mset=0)
rangex/y/z = rotation range on respective axis
enter 0 for no rotation.
skip is angle increment in each frame
Use skip to reduce final movie size or to speed up rotation.       EXAMPLE
movie.tdroll 360,360,360,5
```

as shown on link:
http://sourceforge.net/mailarchive/message.php?msg_id=1534644

```
 movie.tdroll ("Three-Dimensional roll") will allow you to use make a quick   movie
of rotation along multiple axes of whatever is showing on the pymol   screen.  You
can speed it up or slow it down by changing the "skip" value.   Play with setting
different axes to 0 until you  achieve what you want.  I   like movie.tdroll
180,180,0,5 for a cool display of the molecule.
```

4) Saving your movie:
==========

If you are on a Mac, the menu File > Save Movie > As QuickTime should work with all the examples above.

If you are on a PC the files will be saved as single PNG files that need to be assembled, e.g. with QuickTimePro on a PC.

You can also export the individual frames with the line-command `mpng NAME` which would save frames NAME0001.png, NAME0002.png etc until the last frame.

I hope this will get you to what you wanted to do....As you can see, a simple question can lead to a very long answer....

Jean-Yves

P.S. I almost forgot.... There is a separate Python module that can be downloaded and then invoked within PyMol. This module is not part of PyMol and needs to be downloaded, and then activated within PyMol before the new commands are available

You can see an example of this at the following link:
http://sourceforge.net/mailarchive/forum.php?thread_id=1204166&forum_id=60

However the download link is not working, but I found the following link that contains the Python script, but it is INCLUDED within the following archive at the bottom of the archive.

http://sourceforge.net/mailarchive/message.php?msg_id=4272454

The assumed name of the script is movie.py, however it is a different movie.py python script than the one I mentioned above in the movie.nutate paragraph.

This link also contains an example script on the usage of this. THe command "run movie.py" activates the script which is simply kept in a text file.

Some of the meaning of the commands are at:
http://www.rubor.de/bioinf/pymol_movie.html
(but the download link does not work, see above).

NOTE: movie.py is not part of the standard PyMol installation, and any script making use of movie.py commands would only work movie.py is available on the PyMol that is running.

==========