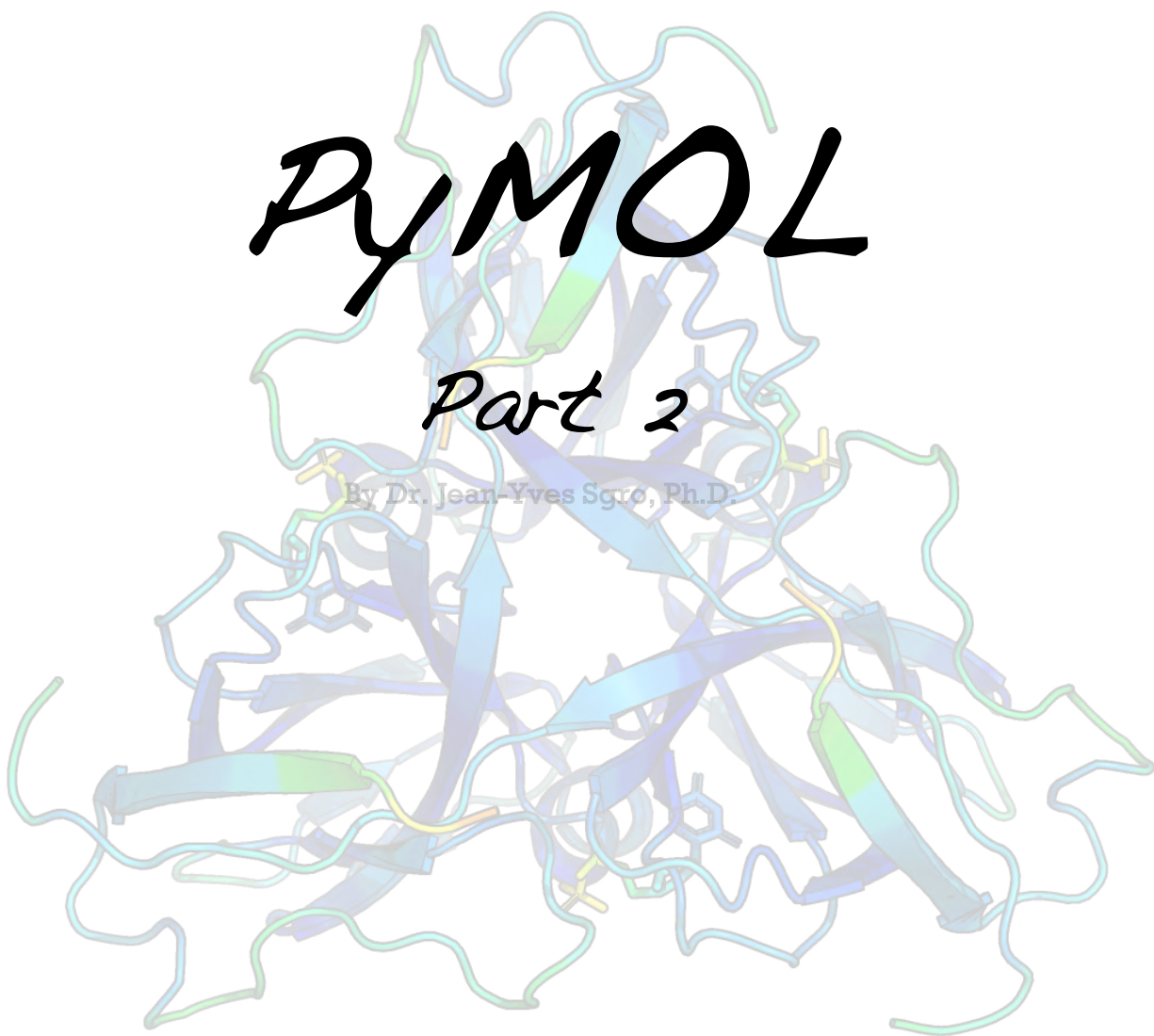


Book 3

PyMOL

Part 2

By Dr. Jean-Yves Sgro, Ph.D.



Original © 1997-2019 Dr. Jean-Yves Sgro, Ph.D.

Permission is granted to make and distribute copies, either in part or in full and in any language, of this document on any support provided the above copyright notice is included in all copies. Permission is granted to translate this document, either in part or in full, in any language provided the above copyright notice is included. The sale of this, or any derivative work on physical media cannot exceed the actual price of support media.

This work is released under the Creative Commons license
Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)



<https://creativecommons.org/licenses/by-nc-sa/4.0/>

You are free to:



Share — copy and redistribute the material in any medium or format **Adapt** — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:



Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.



NonCommercial — You may not use the material for commercial purposes.



ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Cover: PyMOL rendering of PDB 1DUD (*Crystal structure of the Escherichia coli dUTPase in complex with a substrate analogue (dUDP)*). Larsson, G., Svensson, L.A., Nyman, P.O. (1996) Nat.Struct.Mol.Biol. **3**: 532-538)

Preamble

For many years I participated in the teaching of Prof. Ann Palmenberg classes, in particular teaching about molecular graphics on the Desktop at a time when computer use in the classroom was not yet preeminent. This class was a useful complement to the main topic of Sequence Analysis and Evolution also using computers.

I personally used what was called “Unix Workstations” (Silicon Graphics, sgi) which had powerful graphics and rather beautiful “photorealistic” renderings. For teaching molecular graphics on the Desktop, I first used Rasmol, which was an amazing software fitting in about half of a 3.5” floppy disk or about 500 kilobytes. Rasmol included a sophisticated line command language but lacked the beautiful “photorealistic” renderings of the workstations. This was remedied by adding two other software in the course, one for the “publication quality renderings” (VMD) and the other for the modeling abilities of side-chain mutations and automated 3D superimposition of structures (Swiss PDB viewer later called DeepView.)

When PyMOL was still in preliminary development at version 0.99 I spent one intense week porting all the class material to PyMOL. Now, rather than using three different software, all was possible with only PyMOL. Over the years I extended and updated the PyMOL course material.

The UW-Madison Biochemistry students were the primary audience for these classes in courses Biochem 660 and 712, and occasionally in Biochem 511. I offered the PyMOL class in Biochem 660 for over a decade. The PyMOL tutorial and preliminary molecular graphics and file format introduction were part of a very large, made-to-order physical copy of a class book of about 500 pages that also contained tutorials on using other software. The PyMOL section was about 200 pages.

2017 was the last year that Biochem 660 was offered. I have therefore decided to release the complete PyMOL tutorial which you will find split in multiple PDF files. In this final revision, I had updated all web pages, and added links to archived pages when web site were defunct to keep the text as relevant as possible for future use.

I hope that it will be useful to you in accomplishing your molecular graphics goals.

Sincerely,

Jean-Yves Sgro, Ph.D.

Distinguished Scientist | Senior Scientist

[Biotechnology Center](#) | [Biochemistry Department](#)

University of Wisconsin-Madison - USA

Email: jsgro@wisc.edu

Formerly at the Institute for Molecular Virology and [VirusWorld web site](#) creator.

Table of Contents

PyMOL Tutorials (2): Animations	3
1. Movie making for the impatient	3
1.1 Simple 360 Scene Rotation	3
1.2 Simple 30, 60, 90, 120, 180, Scene Rocking	4
1.3 Nutate	4
2. <u>Scene menu</u> - animations & transitions	4
2.1 Create a series of scenes:	4
2.2 Export the viewed scenes into a self-contained movie	7
Simple animation, PyMOL & PowerPoint	9
1. Automatic rotation within PyMOL viewer	9
2. Movie suitable for PowerPoint	10
3. Ray traced movies	13
Advanced movie commands	15
1. PyMOL movie set-up and frame	15
2. PyMOL movie motion commands	16
2.1 movie.roll	16
2.2 movie.rock	17
2.3 movie.zoom	17
2.4 movie.screw	18
2.5 movie.nutate	18
2.6 movie.tdroll	19
NMR models & animation	23
1. Import the 3D data	24
2. Dynamic movie	25
1. Displaying all models at once	26
2. Separating the models	26
3. Moving molecules independently with the mouse	27
4. HIV protease active site - NMR example	28
Multimeric structure: rhinovirus A16	31
1. Asymetric unit	31
2. Complete capsid: the biological unit	33
Scripts	36
1. Review of files, directories and path	36
2. Give the full path	37
2.1 Change default working directory	37
3. Text-only files: a review	37
3.1 Options to create text-only files:	38
3.2 Dealing with line breaks or end-of-lines problems:	38
4. Creating a small script	39
4.1 Running a script	40
4.1.1 set the default directory	40
4.1.2 Run the script: @ command	41
5. Orienting a molecule without the mouse	41

6. Automatic script making: Log menu.....	43
Subsets.....	45
1. Defining subsets by selection: example	46
2. PyMOL selectors: keywords to make atom selections	47
3. Putting it together: a fancier script.....	48
4. Molecule rotation relative to each other	50
Choosing a style: PyMOLWiki Gallery	52
Morphing PDB structures.....	53
1. Morph command	53
2. Morph web server (optional).....	54
2.1 Morphing software (INFO)	57
2.2 Published movie example	57
Where to go from here?	59

PyMOL Tutorials (2): Animations

1. Movie making for the impatient

✓ **OPTIONAL** This section is taken from the “Movie School” online tutorial at http://PyMOLwiki.org/index.php/MovieSchool_1 which states: “If you don't have time to or care to make more complex movies and want a movie now then read this section. It mostly involves the GUI for making movies, so get your mouse ready.”

1.1 Simple 360 Scene Rotation

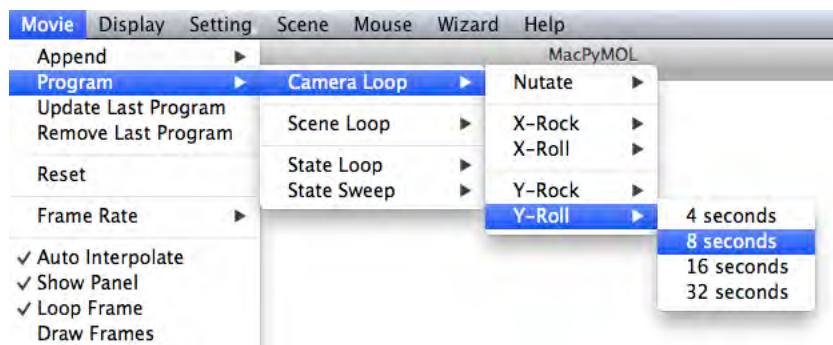
To make a movie that simply rotates around your scene 360 degrees, in the menu click on:

Movie > Program > Camera Loop > X-Roll > N Seconds

for an N-second movie rolling over the X-axis.

For the Y-axis roll chose:

Movie > Program > Camera Loop > Y-Roll > N Seconds



Done! Press **play**. [Triangle button at the bottom right - )

1.2 Simple 30, 60, 90, 120, 180, Scene Rocking

This will show up to a 30, 60, 90, 120 or 180 rocking 'wedge' of the scene. If you don't want to rotate all the way around, use this.

Movie > Program > Camera > X-Rock > X-Degrees over N-Seconds
Done! Press **play**.

1.3 Nutate

Nutating is like a wiggle-rock; try it and see!

Movie > Program > Camera > X-Rock > X-Degrees over N-Seconds
Done! Press **play**.

2. Scene menu - animations & transitions

The **Scene** menu has very nice features to save the current view of a structure in various states of representations and toggle between them. The transition from one scene to the next creates also a beautiful screen animation. This is one method for comparing different versions of structure illustrations, or for creating internal animations to tell a story about the structure.

Preliminary:

- **Open PyMOL**
- **fetch 2biw, type=pdb2**
- Change background: menu cascade **Display > Background > White**

2.1 Create a series of scenes:



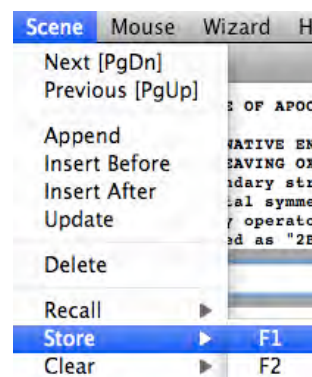
TASK

Create Scene 1: a cartoon representation colored by secondary structure

- Hide everything: **2biw > H > everything**
- Show as cartoon: **2biw > S > cartoon**
- Color by secondary structure: **2biw > C > by ss > pick-a-color.**
- Now **orient** the molecule with the mouse in a way you like.

Store Scene 1 under the F1 function key:

Scene > Store > F1



Create Scene 2: keep cartoon representation, color e.g. spectrum, and rotate the molecule in a different orientation. For a more stunning effect you can zoom in.

- Change cartoon color: **C > spectrum > rainbow**
- **Rotate** the molecule in a different orientation.
- **zoom in**.

Store Scene 2 under the F2 function key: **Scene > Store > F2**

Create Scene 3: Zoomed on ligand, and color

- Zoom on ligand: **2biw > A > preset > ligand sites > solid surface**
- change color to blue hue slate: **2biw > C > blues > slate**
- change ligand color to color wheat:
- **Click** on ligand: **2biw** (This places the ligand under name (sele))
- **(sele) > C > yellows > wheat**
- **Rotate** the molecule in a suitable, pleasant orientation.
- Alter clipping planes: menu cascade **Display > Clip > 12 Angstrom Slab**

Store Scene 3 under the F3 function key: **Scene > Store > F3**

Now you can recall any of those 3 scenes in any order with the **Scene > Recall** Menu.

Note: it is possible to simply click the **F1, F2** etc. **key** on the keyboard if it exists. However, if the key has other predefined function by the operating system, use the **FN** or **fn** key to alter the preset definition.

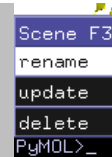


Update: the newest version of PyMOL adds a small icon at the bottom left of the display for each scene that is clickable to recall that scene!

Therefore you can click on **F1**, **F2** or **F3** to return to that defined scene.



Furthermore, the buttons can be renamed by right-clicking (control-click on Macs with one button mouse or trackpad.)



When choosing “rename” a dialog will open on the top left part of the 3D display to allow a new name to be entered. The button will be updated with the new name.

Renaming F3 to: ligand_scene_



✓ TASK

- Recall scene 1: **Scene > Recall > F1**

You should witness a beautiful animation and color change while the scene transitions and twirls to the new scene.

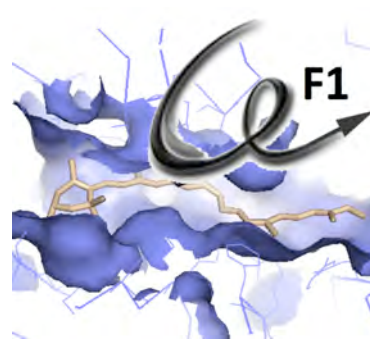
- Recall scene 2: **Scene > Recall > F2**
- Recall scene 3: **Scene > Recall > F3**
- Etc.



Click F2 to go from F1 to F2



Click F3 to go from F2 to F3



Click F1: go back from F3 to F1

2.2 Export the viewed scenes into a self-contained movie



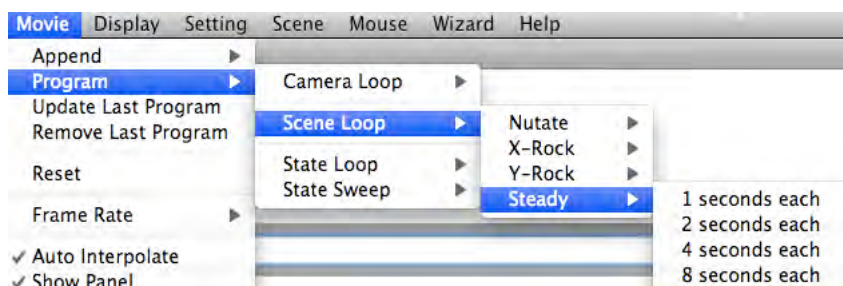
NEW! Newer versions of PyMOL can save the Scene animations into an independent movie, suitable for a Power Point presentation (more on this below!)

Rather than use the F1-F3 function keys, it is now possible to create a linear movie that will change from one state to the next in one click.

This is done with the menu cascade from the Movie menu:




TASK **Movie > Program > Scene Loop > Steady > 4 Seconds Each**



Note that a blue line with tick marks is created at the bottom of the window, titled “camera” on the right hand side. This is the time-line of the movie and the tick marks display when changes will occur.

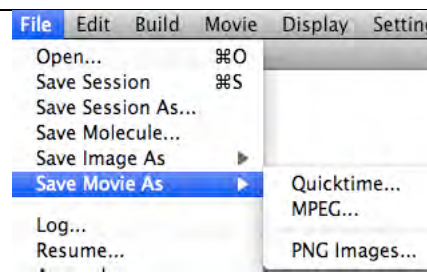


To play the movie, simply **click** the **Play** button at the bottom right (triangle - )



TASK To save the movie as an independent QuickTime or MPEG movie on the desktop use the following menu cascade and then follow the default instructions:

File > Save Movie As > MPEG...



Save the movie on the Desktop to find it easily. Double-click it to play it.



Advanced: Earlier versions of PyMOL (e.g. free version 0.99 and up to version 1.3 or 1.4) cannot export scene changes into an independent movie without additional external scripts. “Slerpy” was created to be used in these older versions.

Advanced users may want to explore the Slerpy.py PyMOL extension which “*creates a moderately easy to use environment for doing keyframe animation. The main function of slerpy is to record a series of PyMOL views. A movie can then be created by interpolating between these views.*”

See <http://www.pymolwiki.org/index.php/Slerpy>

Simple animation, PyMOL & 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.

Preliminary:

- If you are continuing from the previous part, zoom out of the previous exercise view to see the ligand within the pocket, or re-apply a simple preset such as (reload 2BIW.pdb2 if necessary):
- **2biw > A > preset > ligand sites > solid surface**
- recolor surface and side chains as above

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 PyMOL 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.

✓ TASK

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

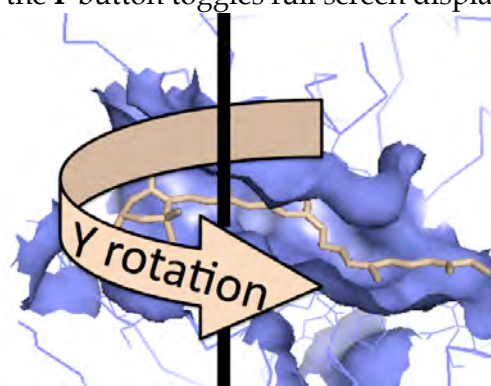
```
mset 1  
mdo 1: turn y,5;
```

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.



Note: The **S** button toggles on-screen sequence and the **F** button toggles full screen display.

- 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 (,) a syntax no longer valid. 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. Movie suitable 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 for a presentation.

✓ **READ** The creation of movies within PyMOL may be different depending on the version and on the operating system. The **MacPyMOL** version contains a built-in function to tap into the QuickTime™ engine and can be used to create movie files. On Mac and other systems, the movie can be saved as an MPEG file. On all systems the movie can also be saved as a series of images (in PNG format) that can be assembled into a movie by third party software.

Since we were using some movie/animation features above, the first command to use is `mclear` to remove functions and frames that 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 python command `util.mroll(start, finish, loop-flag)` does all of this if we have declared the number of frames with `mset`. The actual PyMOL command is: `movie.roll`

Note: A complete 360 degrees rotation is assumed by this command.

✓ TASK

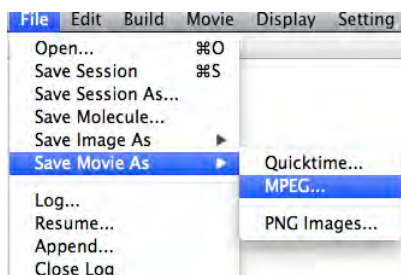
Within the top PyMOL> command line, type the following commands:

```
mclear
mset 1 x36
movie.roll 1, 36
mplay
mstop
```

`mplay` and `mstop` activate the movie manually. To save the movie as an MPEG movie follow these instructions:

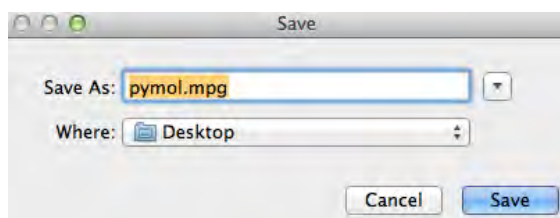
Use the menu cascade:

File > Save Movie As > MPEG...



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

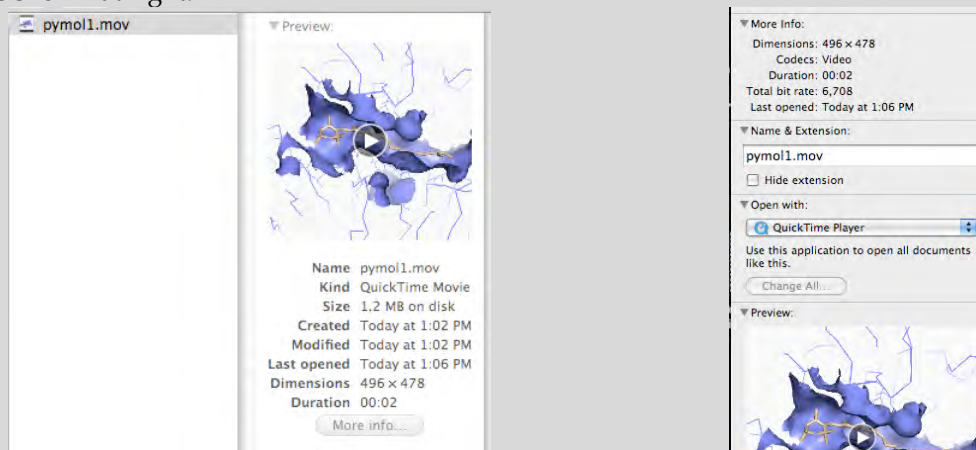
Press the Save button



Double click the **pymol.mpg** file and watch it.

✓ INFO

Within the Mac finder the movie can also be previewed. Additionally it tells us the dimensions of the movie: 496 x 478 in this example. (Your movie will have the size of the current Viewer window.) Clicking on the More Info... button opens the information data of the file, portion of which is shown at right.



Clicking the play circled triangular play button  will indeed play the movie within the preview window.

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.

✓ **TASK** 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
movie.roll 1, 36
```

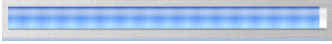
If you want to save the individual images use the following command:

```
mpng MyMovie
```

This command will create images MyMovie.0001.png, MyMovie0002.png etc. The images can then be assembled into a movie by an independent software such as QuickTime™Pro.

If you are using MacPyMOL and want to save the movie with QuickTime™Pro follow the same procedure as above with the menu cascade:

File > Save Movie As > QuickTime...

- Note 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) and the more complex its contents, the longer it takes. You can see which frame is being rendered by looking at the bottom right.

✓ **TASK**

Save this PyMOL **session** for later and **Quit PyMOL**.

From the top menu follow the menu cascade:

File > Save Session As...

Save file on the Desktop as *e.g.* **MySession2.pse**

Advanced movie commands

Preliminary:

- Open a new session of PyMOL.
- Load 2biw or any other molecule of your choice, unless you continue from above
- Change the default representation from green lines to *e.g.* ribbons colored by secondary structure

1. PyMOL movie set-up and frame

Some command are useful during the preparation of a movie, such as the command `rewind` or `frame`.



INFO

PyMOL "frame" commands	Description
rewind	goes to the beginning of the movie.
middle	goes to the middle of the movie.
ending	goes to the end of the movie.
forward	moves the movie one frame forward.
backward	moves the movie back one frame.
frame frame-#	sets the viewer to the indicated movie frame. <i>e.g.</i> <code>frame 5</code>
mmatrix {clear store recall}	sets up a matrix to be used for the first frame of the movie and orient the view in a specific way. Example 1: <code>mmatrix store</code> . Example 2: <code>mmatrix recall</code>

2. PyMOL movie motion commands

The purpose of this exercise is to explore the movie programs built within PyMOL.

Movie movement commands are summarized below from their definition within PyMOL software:

✓ INFO

Movie programs	Definitions and complete parameters
movie.rock	<code>movie.rock(first,last,angle=30,phase=0,loop=1,axis='y')</code>
movie.roll	<code>movie.roll(first,last,loop=1,axis='y')</code>
movie.zoom	<code>movie.zoom(first,last,step=1,loop=1,axis='z')</code>
movie.screw	<code>movie.screw(first,last,step=1,angle=30,phase=0,loop=1,axis='y')</code>
movie.sweep	<code>movie.sweep(pause=0,cycles=1)</code>
movie.pause	<code>movie.pause(pause=15,cycles=1)</code>
movie.nutate	<code>movie.nutate(first,last,angle=30,phase=0,loop=1,shift=math.pi/2.0,factor=0.01)</code>
movie.tdroll	<code>movie.tdroll(first,range_x,range_y,range_z,skip=1)</code>
movie.timed_roll	<code>timed_roll(period=12.0,cycles=1,axis='y')</code>
movie.load	<code>movie.load(*args,**kw)</code>

For many of these commands supplying the first and last frame is often enough, and all the other variables are used per the defaults listed in the table. We have already used `movie.roll` in the previous exercise.

Some of these commands are not documented. For example as of this date there are only 64 Google entries for “*movie.zoom pymol*” and only 4 entries for “*movie.screw pymol*” 2 of which are a listing of all PyMOL commands

2.1 movie.roll

We have already seen `movie.roll` in an exercise above. Here is an example where we take advantage of choosing the axis of rotation (default was y.)

✓ **TASK** Try the following mini movie commands. The `viewport` command is optional, it would set the image size to the standard of a 15 inch screen.

We have already encountered `mclear` and `mset` previously. `mplay` will start the movie and `mstop` will halt it. These actions can also be performed from the VCR controller as explained previously.

```
viewport 640,480      # optional, adjust screen size
mclear
mset 1 x360
movie.roll 1,120,1,axis=x
movie.roll 121,240,1,axis=y
movie.roll 241,360,1,axis=z
mplay
mstop # to halt the movie
```

2.2 movie.roll

The top right panel of PyMOL offers a button called “Rock” which rocks the molecule back and forth very slowly providing insight for depth perception. The command `movie.roll` provides the same functionality. The first 2 arguments are the first and last frame numbers, and the third is the degree of rotation. The movement is faster for larger, wider degrees of rotation. This movement is useful to present a molecule which interesting features are on one side, such as a cavity or an active site. The final movie can be exported to a QuickTime movie, an MPEG movie or a series of PNG files as well.



TASK

Type the following commands:

```
mclear
mset 1 x120
movie.roll 1,120,45
mplay
mstop # to halt the movie
```

2.3 movie.zoom

This function will zoom by default along the z axis (running perpendicular to the plane of the screen.)



TASK Type the following commands:

```
mclear
mset 1 x120
movie.zoom 1, 120
```

```
mplay
mstop # to halt the movie
```

Now change the default axis to x rather than the default z:

```
mclear
movie.zoom 1, 120, axis = x
mplay
mstop # to halt the movie
```

The resulting movement is in fact a translation along the x axis.

Note: since we are using the same amount of frames it was not necessary to restate the `mset` command again.

2.4 movie.screw

This commands operates a movement blending the rocking and the zooming options of the previous commands.



TASK Type the following commands:

```
mclear
movie.screw 1, 120
mplay
mstop # to halt the movie
```

2.5 movie.nutate

This commands is similar to the VMD molecular graphics software default movie movement called “lemniscates” and represents a rotation following the edges of a cone facing the viewer by its opening. The best is to simply view it...



TASK Type the following commands:

```
mclear
movie.nutate 1, 120
mplay
mstop # to halt the movie
```

2.6 movie.tdroll

This command is short for "Three-Dimensional roll" and rotates the molecule along multiple axes.

Adapted from the PyMOL source code:

AUTHOR

Byron DeLaBarre

USAGE

`movie.tdroll(first, rangex, rangey, rangez, skip=1)`
first is the first frame to apply command
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 1, 360, 360, 360, 5`

Note: in older versions of PyMOL, the sequence of number was different and did not include the "first" frame in arguments.



TASK Type the following commands for a rotation around the 3 axes.

Each axis requires $180 / 5 = 36$ frames.

Thress axis requires mset to be at $36 \times 3 = 108$

```
mclear
mset 1 x108
movie.tdroll 1, 180, 180, 180, 5
mplay
mstop # to halt the movie
```

PyMOL will echo:

```
(' tdroll: defined rotations for', 108, 'frames, starting at frame 1')
```

Note: if you want to change the skip, then the number of frames has to be altered also! For example to skip with only 2 since 5 is a bit fast change the mset command:

```
mclear
mset 1 x270
movie.tdroll 1, 180, 180, 180, 2
mplay
mstop # to halt the movie
```

In the following example multiple commands are used successively to create a slower paced rotation:

✓ **TASK** Type the following commands:

```
mclear
mset 1 x300
movie.tdroll 1, 0, 180, 0, 2
movie.tdroll 100, 180, 0, 0, 2
movie.tdroll 200, 0, 0, 180, 2
mplay
mstop # to halt the movie
```

✓ **INFO** PyMOL commands are sometimes found in two forms: the actual python commands which is built within PyMOL or the equivalent PyMOL command itself. Python is the programming language with which PyMOL is built.

For example, we saw above that the python command `util.mroll` is the PyMOL command `movie.roll`.

The python command is called an API command. *API is the abbreviation of application program interface*, a set of routines, protocols, and tools for building software applications.

Within the PyMOL wiki (<http://pymolwiki.org/>), commands are typically shown as both the API command and the PyMOL command. For example the command “`frame frame-number`” positions the movie to the designated frame number (an integer):

DESCRIPTION

`frame` sets the viewer to the indicated movie frame.

USAGE

`frame frame-number`

PYMOL API

```
cmd.frame( int frame_number )
```

However, some commands exist only as an API command:

DESCRIPTION

`count_frames` is an API-only function which returns the number of frames defined for the PyMOL movie.

PYMOL API

```
cmd.count_frames()
```

Typically the API-only command would only be useful to a programmer.



Advanced: Alternate methods and resources not covered in this tutorial:

- **eMovie:** Hodis, E., Schreiber, G., Rother, K., Sussman, J.L., *eMovie: a storyboard-based tool for making molecular movies*, Trends in Biochemical Sciences **32**, 199-204 (2007) - ~~<http://www.weizmann.ac.il/ISPC/eMovie.html>~~ web site now defunct, but archived: <http://bit.ly/2d1yCOo> (note: sample movies as well as download page archives are working.)
In addition, PyMOL Wiki page now hosts the code (top right of page) at <https://pymolwiki.org/index.php/EMovie>
- **rTools:** ~~http://www.rubor.de/bioinf/pymol_extensions.html~~ site defunct. Latest workable archive from 2009: <http://bit.ly/2d1zySX>
- **MovieSchool:** http://pymolwiki.org/index.php/Movie_school

NMR models & animation

Reminder: NMR structural data is typically published in 2 separate PDB entry: one entry is an average structure, the other is a series of models in slightly different 3D positions. Within the coordinates each model is numbered, starts with the keyword **MODEL** and is separated from the next with the keyword **ENDMDL**. Models are called “states” within PyMOL and can be used for animation.

✓ **READ** For this exercise we'll use the PDB structure entry 1SY4 (*“Refined solution structure of the S. cerevisiae U6 RNA intramolecular stem-loop.”*)

✓ **OPTIONAL EXERCISE: Find a pattern within a TEXT file.**

Example: find how many MODEL lines are present in 1SY4 PDB file

Windows

Open a DOS command window: Click Start button and type **cmd**
The DOS command is **FIND** and to make it case insensitive we add **/I** (capital i).
FIND will list all lines. (DOS commands are not case sensitive.)

```
C:\Users\jsgro\Downloads> find /I "model " 1SY4.pdb
----- 1SY4.PDB
REMARK 210 SPECTROMETER MODEL : DMX
REMARK 500 THAN 6*RMSD (M=MODEL NUMBER; RES=RESIDUE NAME; C=CHAIN
MODEL      1
MODEL      2
MODEL      3
MODEL      4
MODEL      5
MODEL      6
MODEL      7
MODEL      8
MODEL      9
MODEL     10
MODEL     11
```

MODEL 12

Macintosh / Unix / Linux

Open a TERMINAL (/ Applications/Utilities/Terminal on a Mac)

For pattern recognition we use `grep`. We can modify it with `-i` (lower case i) to make it case insensitive:

\$ `grep -i model 1sy4.pdb`

```
REMARK 210 SPECTROMETER MODEL : DMX
REMARK 500 THAN 6*RMSD (M=MODEL NUMBER; RES=RESIDUE NAME; C=CHAIN
MODEL      1
MODEL      2
MODEL      3
MODEL      4
MODEL      5
MODEL      6
MODEL      7
MODEL      8
MODEL      9
MODEL     10
MODEL     11
MODEL     12
```

Note: On a Unix/Linux/Mac it is also possible to search for the “BEGINNING of line” which is represented by `^` before MODEL. In this manner none of the header lines would show.

The command would need to be in quotes: **`grep -i "^model" 1sy4.pdb`**

We can also use the “word count” command to count how many there are:

`grep -i "^model" 1sy4.pdb | wc`

There are 12 models included in this entry, but only ONE structure will be visible at first. To see all models the user has 2 options:

- show all models at once, or
- split the models into independent entry copies listed separately on the “Names Panel.”

1. Import the 3D data



TASK Getting prepared:

- Quit PyMOL, then re-**Open** a new session of PyMOL or **MacPyMOL**.
Alternatively: use the top menu: **File > Reinitialize**

- **Get** the 3D data into PyMOL with the line command: **fetch 1SY4**
- Change the background to white (top menu: **Display > Background >**

White)

Alternatively type: **bg_color white**

The Name Panel will show a new entry with the current model/state displayed (1) and the total number of states (12) as illustrated here:





2. Dynamic movie

The 12 structures/models/states can be seen in rapid succession by clicking the play (triangle) and pause (square) buttons on the VCR control at the bottom right.



TASK

- Click on the **play** () and **pause** () buttons of the VCR control and **observe** the molecule movements between the solved states.

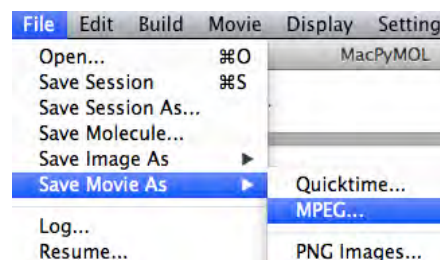
All structures are shown as the default representation (lines). Any changes (*e.g.* change to sticks) would apply equally to all 12 structures:

- Within the **Names Panel** follow the menu cascade: **1SY4 > S > stick**

This animation can be saved as a movie:

- Follow the menu:
File > Save Movie As > MPEG...
to save the movie in MPEG format¹.

- You can change the default name from the default PyMOL.mov to **nmr.mov** for example and save it on the Desktop.



¹ MPEG movies play on all computers. Saving as QuickTime is only possible on MacPyMOL.

This movie can be shown within Power Point presentations and gives an idea of the “wobble” of the molecule in solution.

1. Displaying all models at once

In some cases it may be useful to see all of the models at once superimposed on each other. This is accomplished with an line command switch on or off:



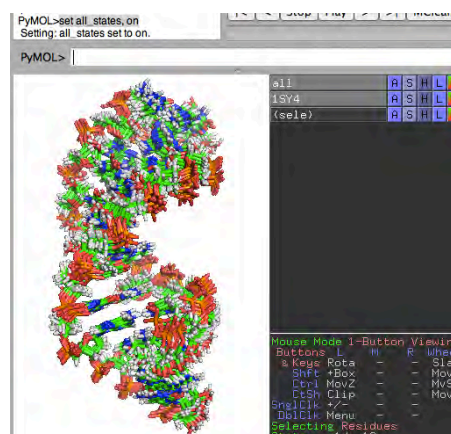
TASK

Type the following command:

```
set all_states, on
```

Note that all models are shown at the same time.

In addition, within the “Name Panel” there is no indication of the states shown/number of states next to the name 1SY4 since all states are visible.



Rotating with the mouse will move all models together.

Go back to the default by typing the following command:

```
set all_states, off
```

2. Separating the models

All models can be copied as individual objects with the command: `split_states`.



TASK

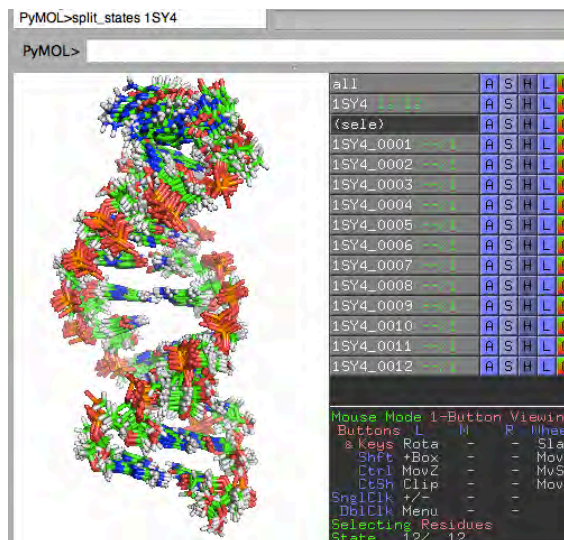
✓ - separate all the states with the line command: **split_states 1SY4**

This line-command will separate all the models as copies and make each one of them a separate numbered object assuming the name of the object is 1SY4.

When the command is given, 12 new objects are created, labeled from 1SY4_0001 to 1SY4_0012.

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.

Since all the models are now visible at the same time, the play button on the VCR control has no apparent effect.



Note: the original, single object named 1SY4 containing all 12 structures is still loaded.

It could be removed with the action menu cascade:

1SY4 > A > delete object

Alternatively type the line command:

delete 1SY4

3. Moving molecules independently with the mouse

It is possible to move molecules relative to each other by using the mouse in editing mode. This is a brief exercise to try this:



TASK

- **Click** on the word “**all**” within the names panel: all images will disappear from view

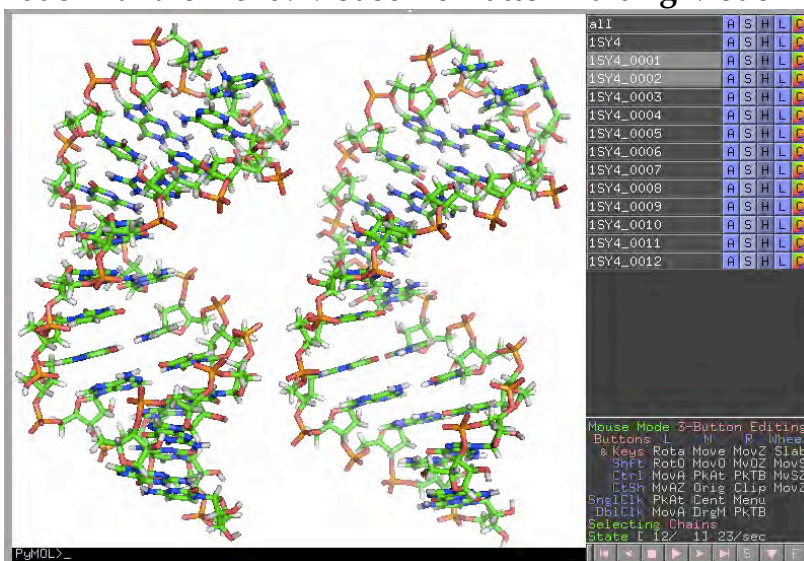
- **Click** on the **2** individual **objects** *e.g.* 1SY4_0001 and 1SY4_0002 to bring them to full view

- **Change** mouse editing mode with the menu: **Mouse > 3 Button Editing Mode**

- **Hold** the **SHIFT** key **and click** on any one of the 2 structures with the **middle button**.

This will allow you to move the structure to the right or left (translation).

- Now **move** the 2 structures apart.



Note: in this mode the left button still rotates and the right button still moves on the Z axis. Holding the SHIFT key restrict the movement to the clicked molecule.

✓ INFO- Summary of the mouse method:

Switch to 3-button 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).

Mouse Mode 3-Button Editing				
Buttons	L	M	R	Wheel
& Keys	Rota	Move	MovZ	Slab
Shift	RotO	MovO	MvOZ	MovS
Ctrl	MovA	PkAt	PkTB	MvSZ
CtSh	MvAZ	Orig	Clip	MovZ
SnglClk	PkAt	Cent	Menu	
DblClk	MovA	DrgM	PkTB	
Selecting Residues				

- **Click on the object** (molecule) you want to rotate or translate.

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

4. HIV protease active site - NMR example

This exercise is another example of multiple models within an NMR structure and is streamlined as mostly line commands. One new feature learned here is about atom selection within a specific distance. The **1BVE** entry contains 28 models.



TASK: Quit or Reinitialize PyMOL (File menu)

```
# fetch the 3D data (PDB ID: 1BVE; 28 models)
fetch 1bve

# make background white
bg_color white

# change presentation style
hide lines
show cartoon

# color by chain
util.cbc

# If you prefer your own colors you first need to know that
# there are 2 chains, named A and B. Click on any protein
# atom to find that information.
# Remove # below to use commands:
# color darksalmon, chain A
# color lightblue, chain B

# show the ligand
show stick, organic

# color the ligand by element color
color atomic, organic

# change the default green for Carbons to gray
color gray70, element c and organic

# Show atoms on the proteins that are within 3Å of ligand
# but select complete residues (byres). Ligand name is DMP
show stick, byres resn DMP around 3



# Hide all hydrogens to see better
hide (hydro)

# Show hydrogens again for ligand
show stick, (hydro) and organic

# Show protein surface (will calculate for all 28 models!)
```

show surface

```
# Make surface 50% transparent  
set transparency, 0.5
```

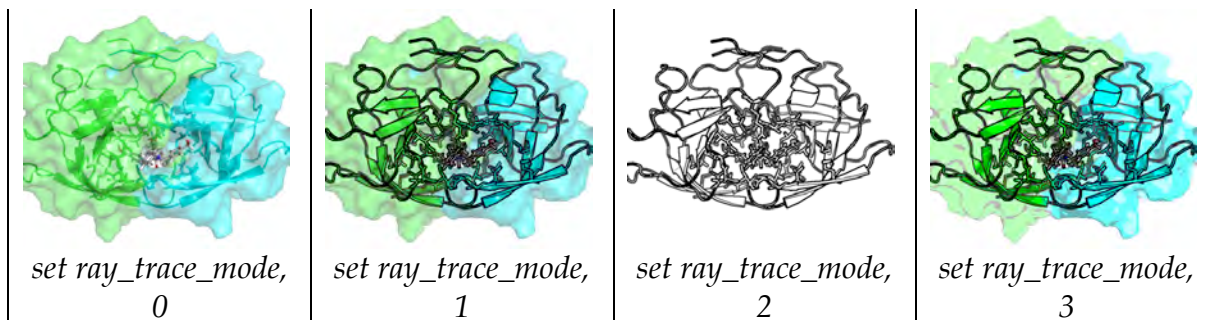
Watch models one by one with the VCR “frame advance” buttons forward  and back 

Note: these commands can be typed within a plain text file and activated all at once as a script. Assuming the file is called `abc.pml` in the current directory where PyMOL is looking (Home directory by default) the script would be activated with the simple command: `@abc.pml`

Play with the ray trace options from 0 to 3 as shown below. (0 is default.)

The commands to use one at a time:

```
set ray_trace_mode, 0  
set ray_trace_mode, 1  
set ray_trace_mode, 2  
set ray_trace_mode, 3
```



Multimeric structure: rhinovirus A16

Goal: to get familiar with virus data entries in the Protein Data Bank. We will use PDB entry 1AYM: human rhinovirus 16 coat protein at high resolution².

PubMed Abstract: Rhinoviruses belong to the picornavirus family and are small, icosahedral, non-enveloped viruses containing one positive RNA strand. Human rhinovirus 16 (HRV16) belongs to the major receptor group of rhinoviruses, for which the cellular receptor is intercellular adhesion molecule-1 (ICAM-1). **In many rhinoviruses, one of the viral coat proteins (VP1) contains a hydrophobic pocket which is occupied by a fatty acid-like molecule, or so-called 'pocket factor'.** Antiviral agents have been shown to bind to the hydrophobic pocket in VP1, replacing the pocket factor. The presence of the antiviral compound blocks uncoating of the virus and in some cases inhibits receptor attachment. A refined, high-resolution structure would be expected to provide further information on the nature of the pocket factor and other features previously not clearly identified.

1. Asymmetric unit

We will first look at one asymmetric unit to become somewhat familiar with the virus structure. The virus is composed of 4 proteins, all of which are the result of cleavage from a single protomer. The viral coat proteins are named vp1, vp2, vp3 and vp4. Vp1 often contains a hydrophobic pocket factor within a hydrophobic pocket. Vp4 often is myristoylated.

² Hadfield, A.T., Lee, W., Zhao, R., Oliveira, M.A., Minor, I., Rueckert, R.R., Rossmann, M.G. (1997) *The refined structure of human rhinovirus 16 at 2.15 Å resolution: implications for the viral life cycle. Structure* **5**: 427-441



TASK: Quit or Reinitialize PyMOL (File menu)

```
# fetch the 3D data (PDB ID: 1AYM)
fetch 1AYM

# make background white
bg_color white

# change presentation style
hide lines
show cartoon

# hide water molecules
hide everything, resn HOH

# color by chain name
color tv_blue, chain 1
color tv_green, chain 2
color tv_red, chain 3
color tv_yellow, chain 4

# show the ligand
show spheres, organic

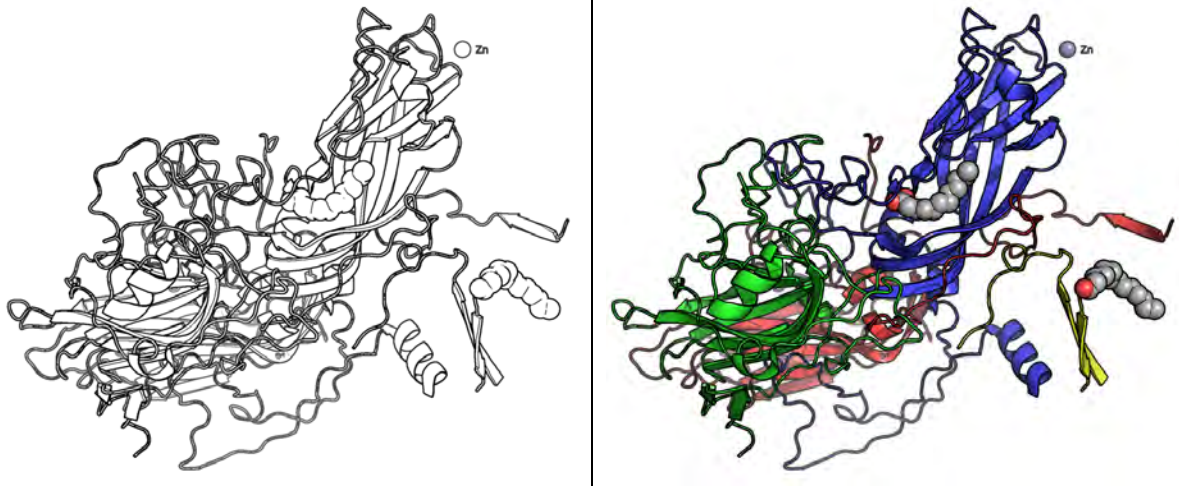
# show and label the Zinc sitting at the icosahedral 5-fold axis
show sphere, element ZN
label element Zn, "          Zn"

# change default orientation: 5-fold axis is at tip of vp1
turn x, -90
turn z, 45
turn y, 25
turn x, -25

# color the ligand and the Zinc by element color
color atomic, organic
color atomic, element ZN

# change the default green for Carbons to gray
color gray70, element c and organic

# Set ray trace method
set ray_trace_mode, 2 # or set ray_trace_mode, 1
ray
```



The 5-fold icosahedral axis is located at the top-left at the tip vp1 which contains the hydrophobic pocket factor.

2. Complete capsid: the biological unit

WARNING: rhinovirus has icosahedral symmetry and the biological assembly contains 60 copies of the asymmetric unit. Therefore some PyMOL functions such as fetching the 3D coordinates or calculating surface may take a long time or even a very long time (or might not be possible) and may depend on the amount of available memory on the computer.

✓ TASK: Reinitialize PyMOL (File menu)

```
# fetch the 3D data (PDB ID: 1AYM)
fetch 1AYM, type=pdb1

# remove water molecules
remove solvent

# Show all 60 models
set all_states, on

# color by chain name
color tv_blue, chain 1
color tv_green, chain 2
color tv_red, chain 3
color tv_yellow, chain 4
```

```
# hide lines and note the presence of 12 Zn atoms. They are located
# at each of the 5-fold axis vertices
hide lines
```

```
# show Zn atoms as spheres. Default color will be that given to vp1
show spheres, element zn
color gray70, element zn
```

```
# Show the proteins as cartoon
show cartoon
```

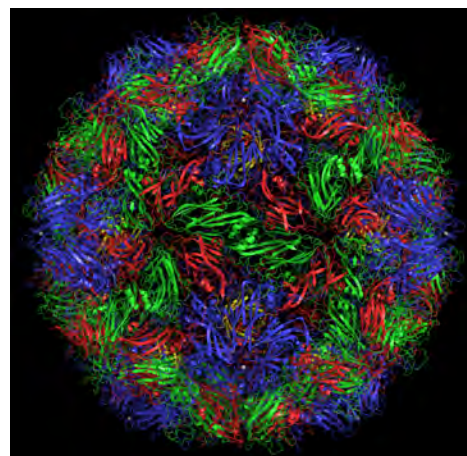
```
# Ray tracing this takes ~15 min
# on an 8-cores /8Gb Ram Mac to
# calculate an 874x846 pixel image
```

```
# DO NOT CLICK the Ray button!
```

```
# actual ray tracing echoed:
```

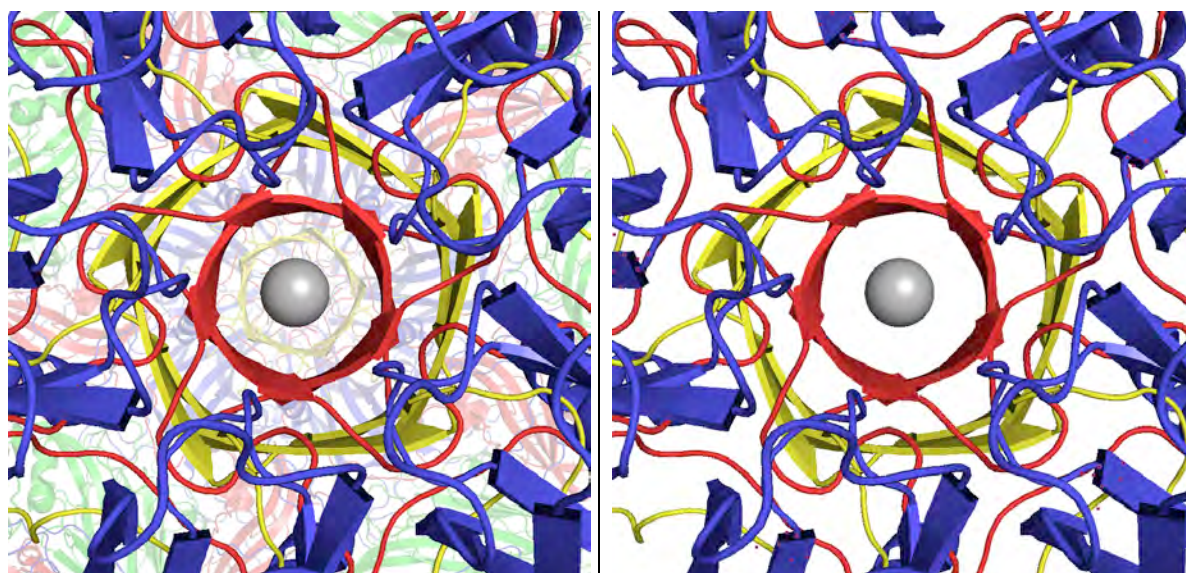
```
Ray: render time: 869.36 sec. = 4.1
frames/hour (923.47 sec. accum.).
```

(that's 14 and ½ minutes!)



✓ **TASK: Using the mouse zoom on a five-fold axis.** Note that the other side of the virus is visible faded in the background. To obtain a clearer picture we can issue a command that will bring the far clipping plane closer to us, here by 100 Å:

```
clip far, 100
```



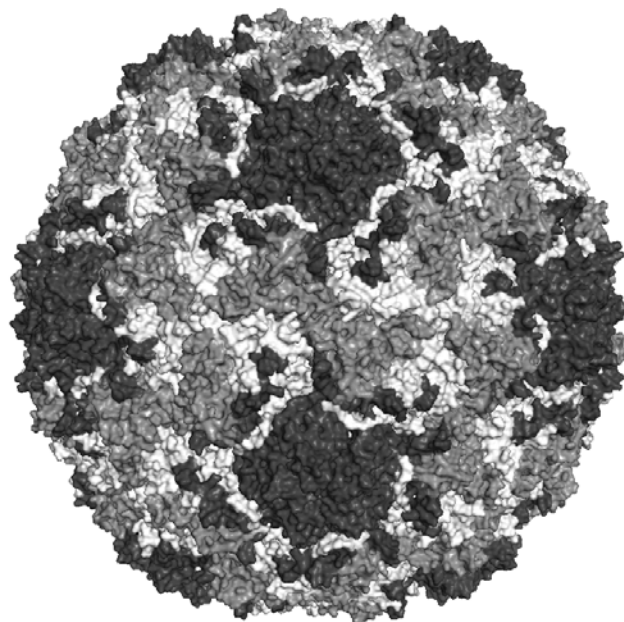
To return to the unclipped image simply move the far clipping back away 100 Å:

```
clip far, -100
```



TASK:

```
# Continue on to show surface  
  
# hide cartoon & show lines  
hide cartoon  
show lines  
  
# recolor chains name  
color gray30, chain 1  
color gray60, chain 2  
color white, chain 3  
color black, chain 4  
  
# Show surface (about 2 min.)  
show surface
```



Scripts

Harnessing the power of PyMOL: scripts

GUI interfaces are nice, but they are slow and cumbersome. PyMOL offers a very powerful and sophisticated 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 (or the menu File > Run...) 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. Review of files, directories and path

✓ **READ** 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 fundamental ways to make sure PyMOL knows where the script you want to use is located:

- Give the full path to the script file

- Change the default directory

2. Give the full path

For Mac OSX or Unix/Linux system, full path to the script may look like:

```
/Users/DMC/Desktop/PyMOL/myfirstscript.pml
```

On a Windows system it may be something like

```
D:\Data\My Datafiles\PMscripts\myfirstscript.pml
```

These long lines give the absolute path to the location of the script, and must be present at each invocation.

2.1 Change default working directory

The other option is to change the directory where scripts (and PDB files) are located. We did something very similar when we gave the command **cd desktop** in a previous exercise, 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.



READ

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.

3. Text-only files: a review



INFO

Although the following might appear trivial, it may be of vital importance!

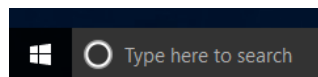
A file containing text that is **bold**, underlined, or *italicized* is NOT a plain text file.

A plain text file:

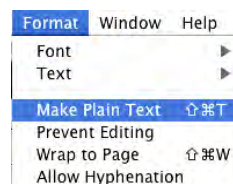
- contains ONLY printable characters that can be typed from the simplest of keyboards: letters, numbers and symbols such as !, @, and \$;
- is not specific to any particular word processor format and is displayed with the default font within any word processor;
- contains a “line break” and/or “end-of-line” code (specific to the operating system) that signifies that the line has terminated “here” and that a new line will begin “next” also known as carriage return (CR) and line feed (LF) respectively
- see also http://en.wikipedia.org/wiki/Text_file
and http://en.wikipedia.org/wiki/Plain_text

3.1 Options to create text-only files:

- Windows: *WordPad* or *NotePad* should be free, standard text editors within the Start (search) window at bottom left of Windows



- Mac OS X:
 - The included **Applications > TextEdit** can be used only after the proper menu has been called: **Format > Make Plain Text**
 - **Applications > Classes > BBEDIT** is a freeware program that offers many options and will be used in this class.



<https://www.barebones.com/products/bbedit/>

- General Unix/Linux/MacOSX

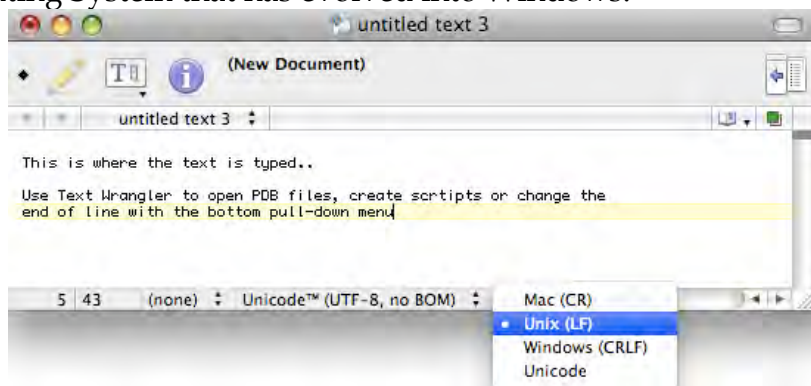
Any file created with a redirect command such as `cat > myfile.txt` or a text editor such as **pico**, **nano**, **xedit**, **vi**, **emacs** is created as a plain text file.

3.2 Dealing 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.

Hard-returns are coded with 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/Windows is for the old Disk Operating System that has evolved into Windows.

On Mac OS X BBEDIT can switch between the 3 types of hard-returns with a single mouse click at the bottom of the window as seen on this image.



<CR> means carriage return, <LF> means line feed and (CRLF) is both of them.

✓ **READ** - 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. This even includes the simple action of *copy/paste* from a PDF document!

Note: See also: <http://en.wikipedia.org/wiki/Newline>

4. Creating a small script

✓ TASK

Preliminary:

- For this exercise the script will be saved on the **Desktop**.
- If PyMOL is running, **Quit PyMOL** and **restart** it again, to have a fresh start.
- Open either **TextEdit** or **TextWrangler** (both located under / Applications)
With TextEdit remember to use menu: **Format > Make Plain Text**
- With the text editor **create** and **save** a **new script** that we shall call **abc.pml**.
- **Save abc.pml** on the **desktop**.

In a PyMOL script comment lines which start with #.

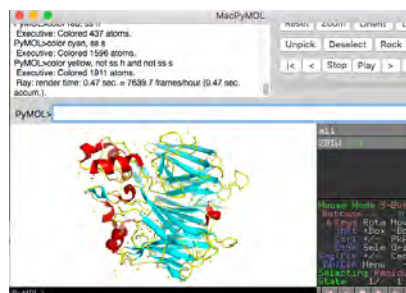
You can omit typing anything after the comment sign # for these exercises. However, placing comments within your scripts is a very good practice for your future reference or for sharing your scripts. This can literally save hours of frustrations!



TASK

Within abc.pml file type the following lines

```
# this is my first script:
fetch 2BIW, type=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 the loops or turns.

4.1 Running a script

4.1.1 set the default directory



TASK

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


```
cd Desktop
pwd
```

Note: You need to be connected to the Internet to `fetch` the coordinates. Alternatively download the coordinates into a file and change the script command from `fetch` to `load`.

4.1.2 Run the script: @ command

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

Example: @**abc.pml**

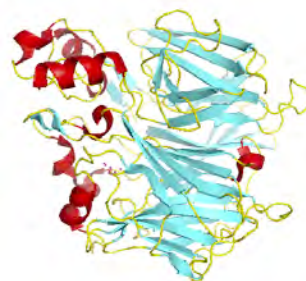
A screenshot of the PyMOL command line interface. The text 'PyMOL>' is followed by '@abc.pml' which is highlighted with a blue selection box.

Note: either the top or bottom line command can be used

A screenshot of the PyMOL command line interface. The text 'PyMOL>@abc.pml_' is visible. To the right, there is a small window titled 'Selecting State [1]' with navigation buttons.

✓ TASK

Type @**abc.pml** within either top or bottom line command.



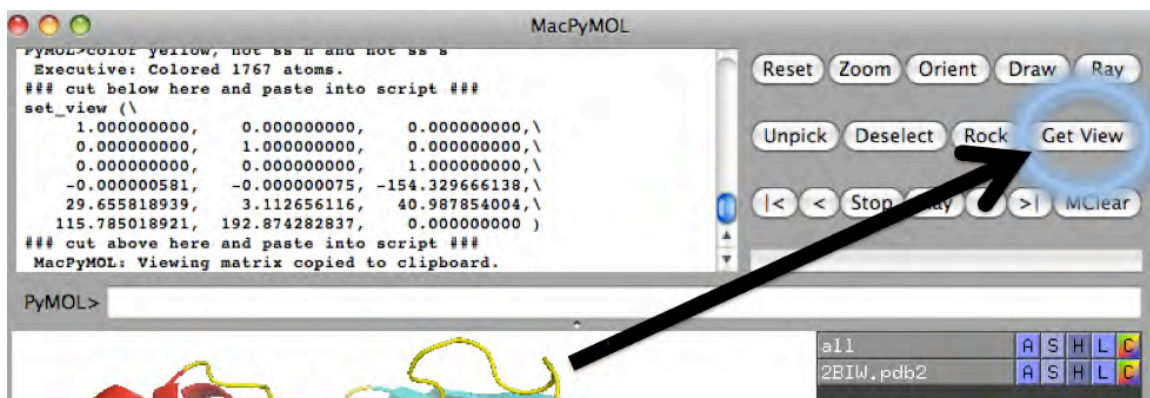
Activating the script will load the structure and create and modify the visual representation of the protein. The same image could have been obtained by either typing one by one each of the script commands within the line command, or by using some of the mouse options as we have done before.

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.

5. Orienting a molecule without the mouse

The image issued from the script was created with the default orientation that the molecule adopts when it is first opened. However, one important aspect of a structure illustration is the orientation, or even close up of the molecule(s).

Fortunately, PyMOL offers a very nice way to return to a previously determined orientation and inscribe it within a script: the “Get View” button, located at the top right.



TASK

- **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 **fetch** command to read: **# fetch 2BIW**
- **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 **fetch** 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.

Note: if the script appears NOT to function, make sure that the end-of-line is that appropriate to your operating system!



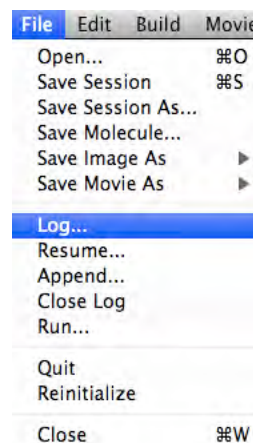
READ *The PyMOL way*: Simply create and modify a script as you go, commenting out the loading of the PDB file(s) after the first run. It is a preferred way of working with PyMOL that leaves the legacy of a finished script with complex commands. Therefore you can create your own repository of complex 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.

6. Automatic script making: Log menu

PyMOL offers an easy way to make scripts: save a log file of typed line-commands

✓ **READ** 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 only the typed commands. 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.



✓ TASK

On your own create a log file while typing the commands on the right hand side on the PyMOL line-command:

Reinitialize PyMOL (**File > Reinitialize**) 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 extension e.g. **test.log.pml** otherwise the file cannot be opened by this method.

```
cd desktop
fetch 2BIW
select hetatm
show stick
hide stick
show stick, sele
color yellow, sele
```

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

Subsets

Select command, parameters, scripting, and subsets.

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



READ The PyMOL “**select**” line-command has the special property to create atom selections with a chosen name 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 in the previous exercise, we used the command “**color red, ss h**” to color in red 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.

Note: When switching from Rasmol this can be confusing, as the command “select helix” is a valid command on the Rasmol command line.

1. Defining subsets by selection: example

Preliminary:

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

✓ TASK

Within the line command type: `PyMOL> color green, helix`

`color green, helix`

Selector-Error: Invalid Selection Name.

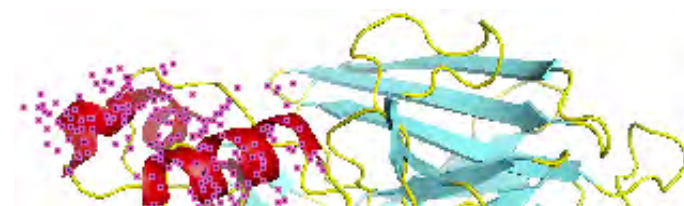
`helix<--`

Note the error echoed within the PyMOL text window.

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 and decorated with pink squares:



all	A	S	H	L	C
2BIW.pdb2	A	S	H	L	C
(helix)	A	S	H	L	C

Within the line command type:

`color green, helix`

Note: unlike the previous attempt, this time the command worked and the helices were colored green.

Note: The **(helix)** selection can be deleted with the line-command: `delete helix`. Alternatively the selection can be deleted with the "A" Actions menu "delete selection." The next menu allows to rename the selection.

all	A	S	H	L	C
2BIW.pdb2	A	S	H	L	C
(helix)	A	S	H	L	C

Actions:
delete selection
rename selection

**READ**

Important lesson: the word “**helix**” here could have been any other word such as “**MyHelixSelection**.”

all	A	S	H	L	C
2BIW.pdb2	A	S	H	L	C
(MyHelixSelectio	A	S	H	L	C

... any word can do ...

Using a “**My**” prefix before some of the selections can 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!

Another important difference to note is how a selection is shown in parentheses within the Names panel, while other objects are not.

2. PyMOL selectors: keywords to make atom selections



READ 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” 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 is a column 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 numbers are also in column before the XYZ *Cartesian* coordinates.

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 old on-line PyMOL manuals (<http://pymol.sourceforge.net/html/index.html>) or it’s echo as the wikipedia version (<http://www.pymolwiki.org/index.php/>). In summary, below are the most relevant selectors for working with macromolecules (proteins and nucleic acids).

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

Each definition below is followed by an example. Remember: as we have seen before, the word after the word “select” is of your own creation. Adding “My” to the word may be a good reminder of that fact.



READ Source:

http://www.pymolwiki.org/index.php/Property_Selectors

3. Putting it together: a fancier script

Lets make a last script to wrap it up. Create a script file called `abc2.pml` with e.g. TextEdit (Notepad on a PC), and save it on the desktop.

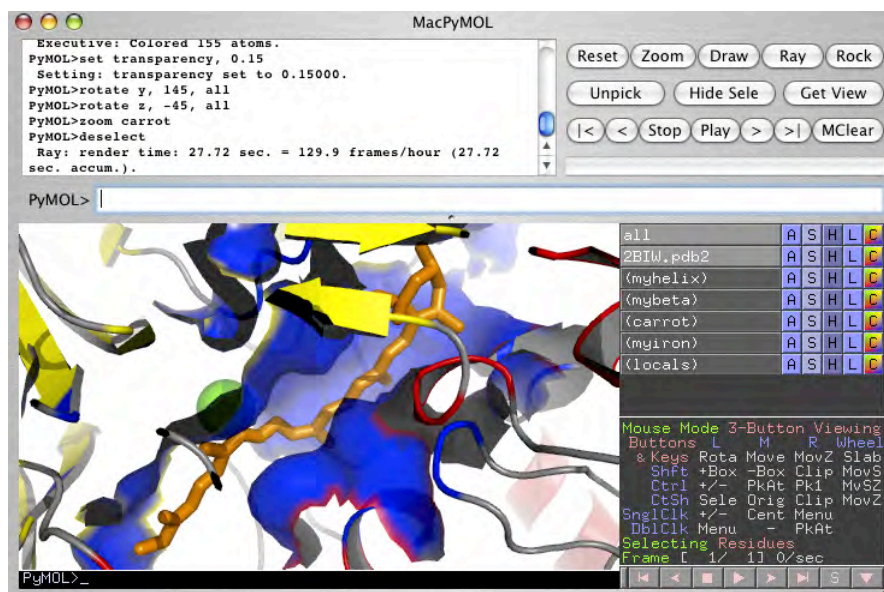
Within `abc2.pml` type the following script. *At this point you can omit typing the commented (#) lines*, but remember that placing comments within your scripts (annotation) can save (lots of) time and remove headaches in the future...

```
# 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
fetch 2BIW, type=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
# to make a fancy image: remove the “#” in the next line
# 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. Molecule rotation relative to each other

✓ **OPTIONAL** 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: reviewed with NMR, multistate structures exercise.

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.)

```
fetch 1FJ1
# 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
```

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 character `\` 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 designated object, which is then rotated around the y axis with the command `rotate`

```
# 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

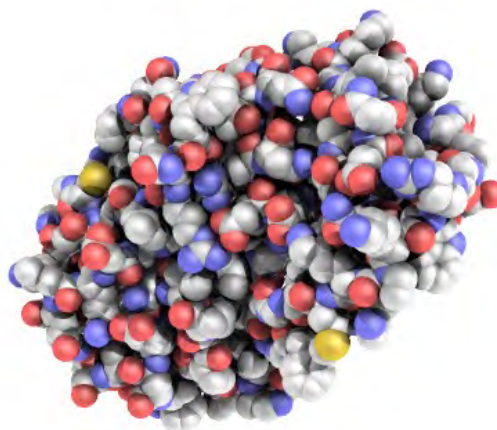
# zoom interface region
zoom inter
show sph,inter
disable inter
```

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`.

Choosing a style: PyMOLWiki Gallery

✓ **INFO** PyMOL offers many creative options for the display of molecular structures and we already encountered a few such as filled interior split surfaces and cartoon-like outline representations with options of the “ray” command. The PyMOLWiki Gallery³ offers a summary of various graphical styles that can be obtained with PyMOL. Here is one example from this gallery named “QuteMol like” from the style inspired by the QuteMol⁴ open source software. Note that the “striking” effect occurs only after the last command “ray” has been issued!

```
fetch 1hpv
set_color oxygen, [1.0,0.4,0.4]
set_color nitrogen,
[0.5,0.5,1.0]
remove solvent
as spheres
util.cbaw
bg white
set light_count,10
set spec_count,1
set shininess, 10
set specular, 0.25
set ambient,0
set direct,0
set reflect,1.5
set ray_shadow_decay_factor,
0.1
set ray_shadow_decay_range, 2
unset depth_cue
ray
```



³ <http://www.pymolwiki.org/index.php/Gallery>

⁴ <http://qutemol.sourceforge.net/>

Morphing PDB structures

✓ INFO

Morphing is a method that transforms a structure into another over a series of intermediate conformations. This can be useful to visualize and animate a hinge between 2 domains or an open/close enzymatic structure for example. There are linear interpolation methods as well as more sophisticated molecular dynamics methods, both of which are beyond the scope of these exercises.

There are 4 ways to create a morphing movie with PyMOL depending on the version of PyMOL that you have.

- 1) Starting with version 1.6 a new morph command makes the process very easy. We'll start with this method.
- 2) Use a morph web server to create the transition states.
- 3) Use a third-party software on the local computer (e.g. LSQMAN)
- 4) For older PyMOL version there are expert version of PyMOL containing the RigiMOL options (incentive software,) not covered in this tutorial (see below.)

The other, more elaborate methods are presented here as “optional” for completeness but are not expected to be addressed in class.

1. Morph command

This a command available starting with PyMOL 1.6. For older versions see below. The command is detailed on the pymolwiki:

<http://www.pymolwiki.org/index.php/Morph>

The provided example is straightforward and can be executed in a couple minutes:

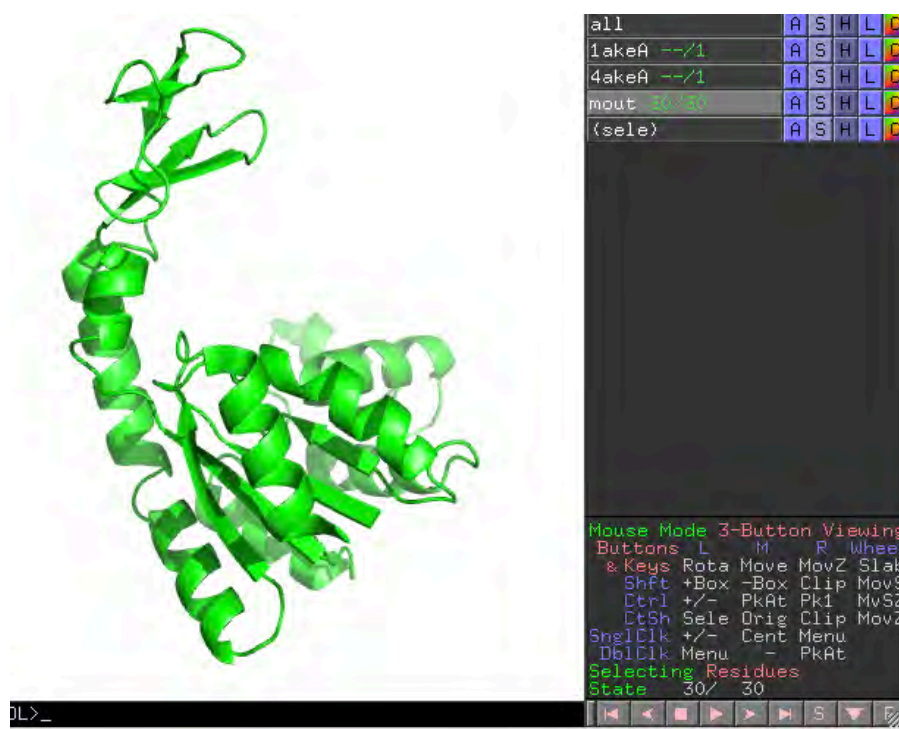
✓ TASK type the following commands to morph these 2 structures:

```
fetch lakeA 4akeA, async=0
align lakeA, 4akeA
morph mout, lakeA, 4akeA
```

The created object mout will have 30 “states” as shown at the bottom right pane.

The command `show cartoon` can be added to see the cartoon style, or the representation can be changed using the GUI under the S (Show) and H (Hide) buttons.

Note: The morph feature is available from the object menu panel: **A > generate > morph**



The remaining morph information below remains here mostly for informational purposes:

2. Morph web server (optional)

✓ OPTIONAL ✓ INFO

The Database of Macromolecular Movements (<http://www.molmovdb.org> and <http://www2.molmovdb.org>) host single chains and multi-chains versions of morph servers as well as a morph2 server as detailed in: http://www2.molmovdb.org/wiki/info/index.php/Morph_Server

For this exercise the file will be given to you by the instructor as the morph server can unpredictably slow or even not available.

The following example was created with the server morph2

http://www2.molmovdb.org/wiki/info/index.php/Morph2_Server

<http://morph2.molmovdb.org/submit.html>

<i>input</i>	<i>PDB ID</i>	<i>TITLE</i>
file1	2RGX	CRYSTAL STRUCTURE OF ADENYLATE KINASE FROM AQUIFEX AEOLICUS IN COMPLEX WITH AP5A
file2	2RH5	STRUCTURE OF APO ADENYLATE KINASE FROM AQUIFEX AEOLICUS

The server created `all.pdb`, a single file containing 9 structures named MODEL 1 to MODEL 9. To make it easier within PyMOL, the `ENDMDL` keyword was manually added at the end of each model so that the file would behave just like an NMR multi-model as we have just seen, renamed `all2.pdb` and supplied in class.

File: `all2.pdb`

```
ATOM 1612 CG1 ILE A 202 -5.467 -17.118 32.130 1.00 38.87
ATOM 1613 CG2 ILE A 202 -2.988 -16.950 31.502 1.00 39.19
ENDMDL
MODEL 4
ATOM 1 N MET A 1 -4.807 -22.979 23.599 1.00 28.85
ATOM 2 CA MET A 1 -3.748 -22.018 23.941 1.00 28.77
```

Preliminary:

✓ - Open a new session of PyMOL.

In the NMR file exercise we used the command `split_states` to split the PDB file into individual objects. Here we will use the `mset` command to indicate PyMOL to pass through each state one by one.

The manual definition of the `mset` command provides clues for animating states:



READ

DESCRIPTION

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

EXAMPLES

```
mset 1 // simplest case, one state -> one frame
mset 1 x10 // ten frames, all corresponding to state 1
```

```

mset 1 x30 1 -15 15 x30 15 -1
// more realistic example:
// the first thirty frames are state 1
// the next 15 frames pass through states 1-15
// the next 30 frames are of state 15
// the next 15 frames iterate back to state 1

```

The mset command is followed by an arbitrarily list of statements which defines the entire movie. Each statement takes on one of three forms:

A number simply indicates a state is to be played next.

x# A lowercase "x" immediately followed by a number (no space) indicates that the previous state should be repeated that many times total.

-# A hyphen immediately followed by a number (no space) indicates that a numeric sequence of states are to be appended onto the movie starting with the previously played state going to indicated state.

Once a movie has been defined, the red "VCR" controls in the lower-right-hand corner of the viewer can be used to step or play through the movie.

Examples

```

mset 1 x30      # creates a 30 frame movie consisting of state 1 played
                 30 times.
mset 1 -30      # 30 frames movie: states 1, 2, ... up through 30.
mset 1 -30 -2    # 58 frames: states 1, 2, ... , 29, 30, then 29, 28,
                 ... , 4, 3, down to 2
mset 1 6 5 2 3   # 5 frames: states 1, 6, 5, 2, 3 in that order.

```

✓ OPTIONAL ✓ TASK

✓ - Open file all2.pdb

✓ TASK

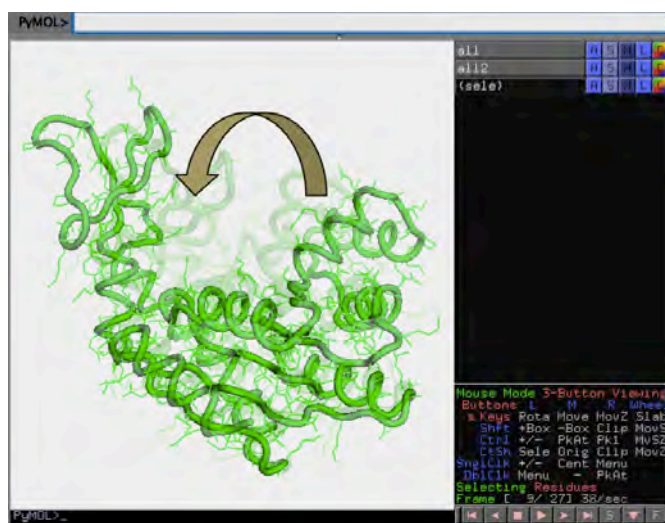
Type the following commands:

```

mclear
mset 1 -9 -1
mplay
mstop # halt movie

```

You should see the molecule "animate" between various positions



We can add a pause at state 1 : stay 10 frames on state 1 by adding **1 x10**
Type the following commands:

```
mclear
mset 1 -9 -1 1 x10
mplay
mstop # to halt the movie
```

✓ TASK

The following set will make the movement even slower by staying 10 frames on each state:

```
mclear
mset 1 x10 2 x10 3 x10 4 x10 5 x10 6 x10 7 x10 8 x10 9
x10
mplay
mstop # to halt the movie
```

✓ INFO

If the data is not in a single file with MODEL and ENDMDL keywords, each PDB file can be imported one by one into a single object with the `load` command:

```
load foo1.pdb,mov      # loads foo1.pdb into state 1 of "mov".
load foo2.pdb,mov,2    # loads foo2.pdb into state 2 of "mov".
load foo3.pdb,mov,3    # loads foo3.pdb into state 3 of "mov".
load foo4.pdb,mov,4    # loads foo4.pdb into state 4 of "mov".
```

2.1 Morphing software (INFO)

✓ Advanced

One such software is **LSQMAN** from the Uppsala Software Factory
http://xray.bmc.uu.se/usf/mol_morph.html
http://xray.bmc.uu.se/usf/lsqman_man.html

2.2 Published movie example

✓ INFO

Example movies are in the Supplementary information located at the web address:
<http://www.nature.com/nature/journal/v450/n7172/supinfo/nature06526.html>

From the article: *Reaching for high-hanging fruit in drug discovery at protein-protein interfaces*. James A. Wells & Christopher L. McClendon, *Nature* **450**, 1001-1009(13 December 2007) doi:10.1038/nature06526

Where to go from here?

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

Philosophy: *Writing scripts is the PyMOL way!*

Learning by example: more PyMOL web resources

Learning by example 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 : Archived** <http://bit.ly/1rwHr5x>
- **Brief PyMOL tutorials: *defunct* but archived link provided:**
Archived: <http://bit.ly/1vk7rB9>
http://www.mrc-lmb.cam.ac.uk/rlw/text/pymol_tutorial.htm
Archived version: <http://bit.ly/1mvU1RO>
- **Google or other search engines:** Great way to find new resources.



Advanced and Extremely Advanced. Alternate methods and resources not covered in this tutorial:

Python Scripting with Python language:

- **Tutorial:** http://www.pymolwiki.org/index.php/Script_Tutorial

- **My PyMOL Script repository**

<http://adelie.biochem.queensu.ca/~rlc/work/pymol/>

Archived: <http://bit.ly/2cAwij8>

This server is now replaced, as shown below.

New server name:

<http://pldserver1.biochem.queensu.ca/~rlc/work/pymol/>

Archived: (Aug 27, 2017) <http://bit.ly/2xObZtF>



tutorial notes

http://www.gutenberg.org/files/24518/24518-h/images/v2_image64.png