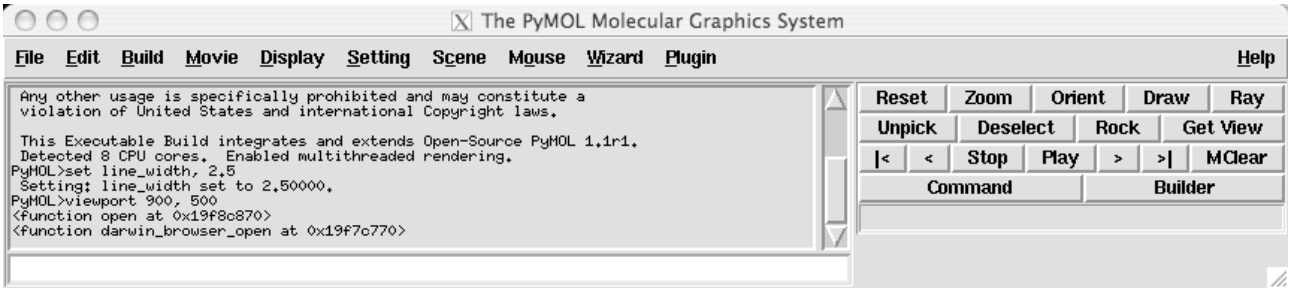
TUTORIAL – INTRODUCTION TO PYMOL

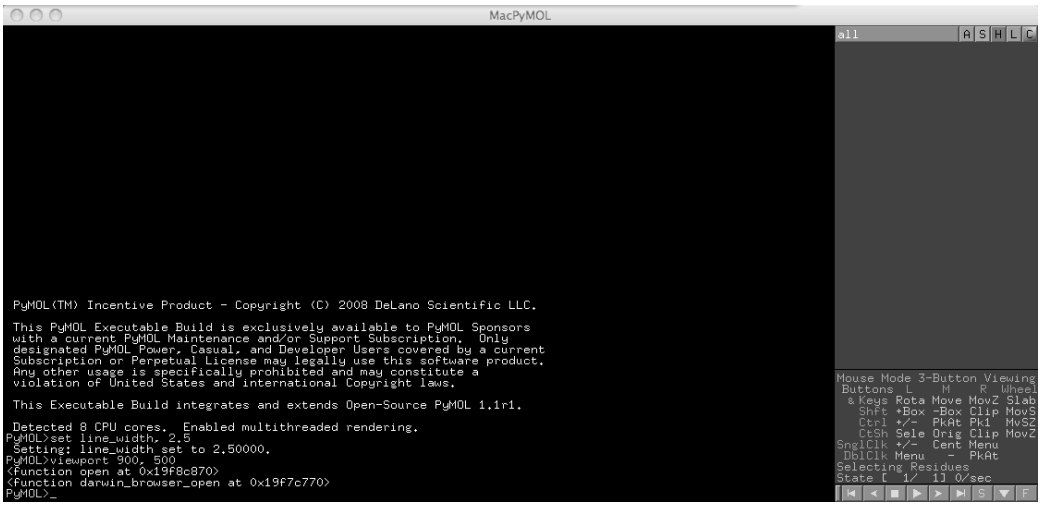
PYMOL
*(version used: **2.3**)*

(Note: Perform only bold-written statements during the tutorial, which are preceded by the >>)

PyMol is a molecular graphics program and modeling, free, *open-source,* and extensible through the implementation of Python language *scripts.* With its many features, ease of use, excellent graphics rendering and ability to generate high-quality images and movies, PyMol has established itself as a *standard* in molecular graphics and modelling. PyMol is distributed free of charge to students and teachers at ***https://pymol.org/edu/?q=educational/*** (after registration), from which you can obtain installation packages for Linux, MAC OS X and Windows operating systems. Once installed, you can start PyMol by double-clicking on the icon. At startup, two sub-windows will appear:



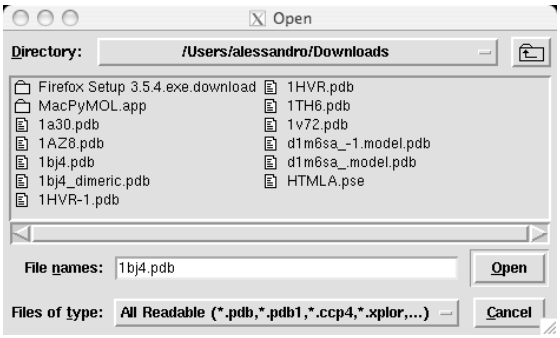
The first window, which is called the ***"PyMol External GUI"*** (**GUI** stands for Graphical User *Interface*) contains: at the top a menu through which you can access command windows; in the middle a report ("***log***") of the commands executed; at the bottom an area for inserting text commands (***Command Input Area***; the latter option, which allows advanced use of the program, will not be considered in this introductory tutorial; for more information on this topic, please refer to the official documentation of the program) and on the right a button with some commonly used commands.



The second window is divided into two sections: in the main one, the graphical **environment** (***viewer),*** the display of the molecules loaded within the program; in the second one, called ***PyMol Internal GUI,*** there is a top panel with the list of objects present in the *viewer,* any selections made, and the operations that can be carried out on them; 2) a lower panel, related to what you can do with the *mouse* and the actions on the movies*(play, stop,* and so on).

PYMOL COME VIEWER

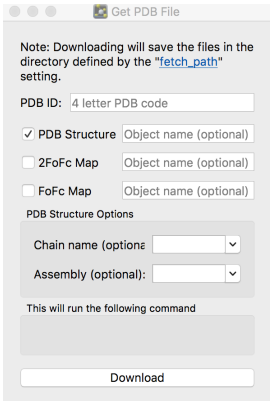
PyMol is, primarily, a program for molecular visualization, albeit it is possible to carry out with it (and especially with the python *scripts* through which it can be extended) also several structural and functional analyses on the molecules investigated. In this tutorial, which has absolutely no claim to fully explore all of the PyMol's features, we'll only discuss a few commonly performed operations.
It is possible to load a molecule, in the example that follows a protein, in several ways. One of these is that the *file* containing the coordinates of the protein (usually, a *.pdb file,* although with PyMol you can read many other formats) is already present in our *computer* (that is, downloaded, for example, from Protein Data *Bank)* and does not require an active connection to the *Internet* network. In this case, select the *File* entry from the External *GUI* (top left), then *Open...*
In the window that appears, similar to the one shown below:



you can select a *file* and/or browse to the directories of our PC.
The second approach, faster and more convenient if you have an active connection and already know the PDB code associated with the macromolecule of interest (remember, the PDB code is a 4-symbol identification code through which the three-dimensional structures of macromolecules deposited in the database "Protein Data Bank"- **www.rcsb.org/** are uniquely annotated) is to select the *File* entry from the External *GUI* (top left), then *Get PDB...*

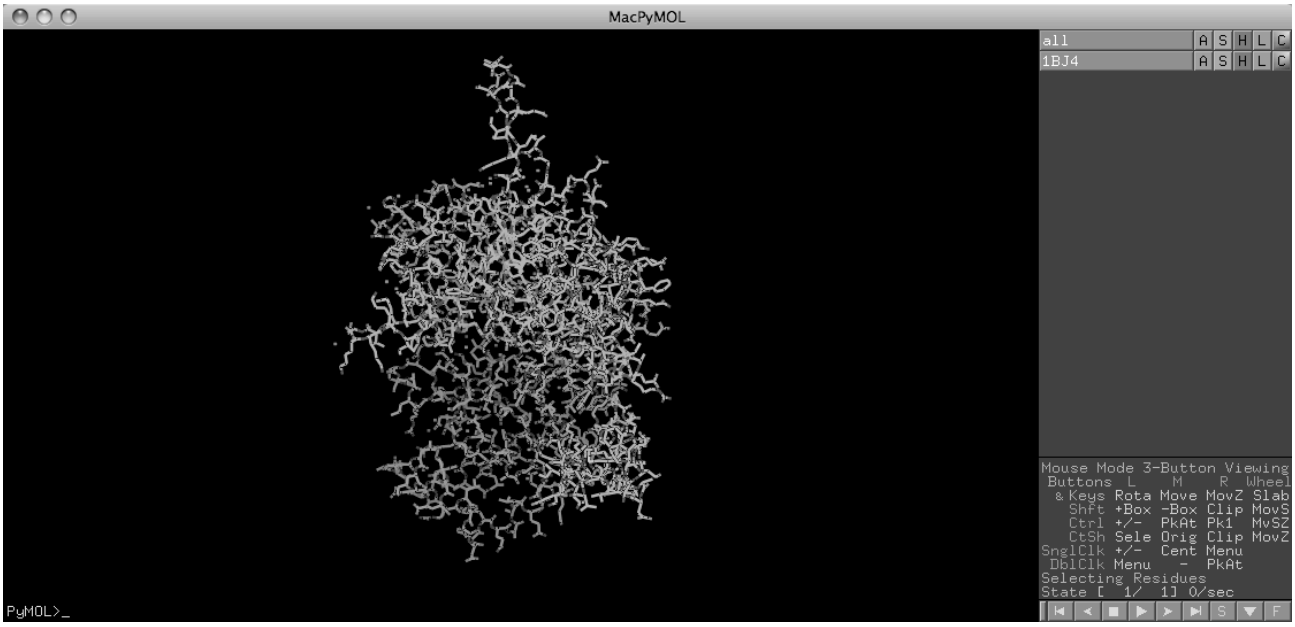
>> select the *File* entry from the External *GUI* (top left), then *Get PDB*

Once selected, a window similar to the one shown below will appear:



>> Let's type in it, for example, the code "1BJ4" (the crystallographic structure of the human serine hydroxymethyltrasferase with the PLP cofactor) and type the "Enter" button.

After a few seconds, depending on the speed of your connection, the structure will appear on the *Viewer*:

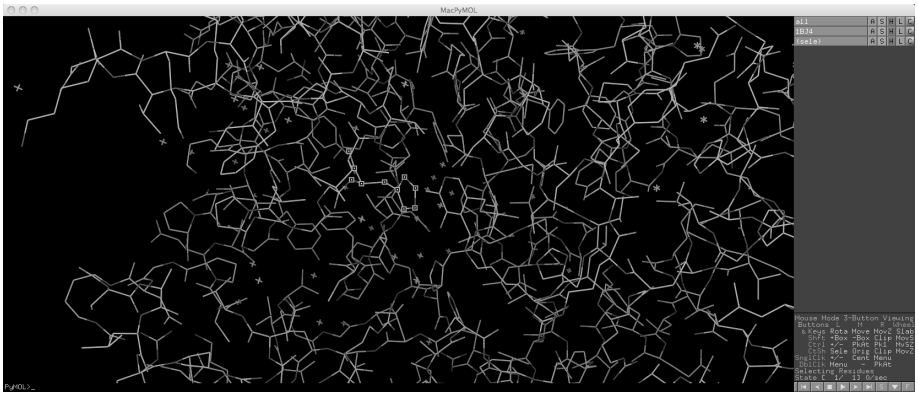


Note that, along with the structure, a new line, called "**1BJ4**" (the top line, "*all*", always present, indicates all objects that are present) has appeared in the upper panel of the *Internal GUI.* Before exploring the meaning of this line, let's train yourself with mouse movements (to make the most of PyMol, it is highly recommended to use a three-keys mouse; in this introductory tutorial, only the main mouse commands will be analyzed).

>> Let's put the cursor in the *Viewer* window and:

- **holding down the left button, we rotate the molecule**
- **holding down the middle button, we translate the molecule**
- **holding down the right button, we *zoom in* on the molecule**
- **If the *mouse* has got a wheel, by rotating it, we can hide sections of the molecule through the use of "*clipping planes*", imaginary planes in front and behind the molecule**

>> We click with the left button any atom of our molecule, the corresponding residue will be selected, and in the upper panel of the *Internal GUI* will appear a new entry, "sele":

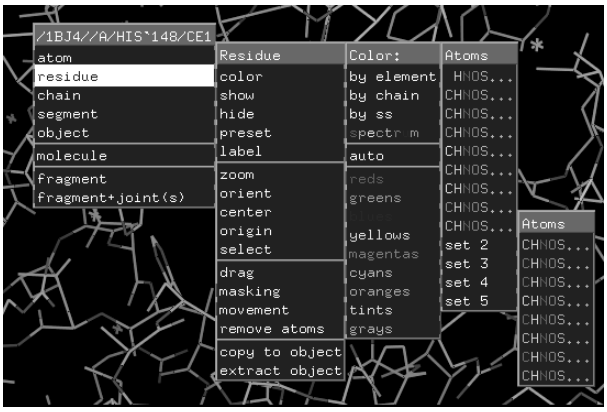


Note that the whole residue selection occurs because the *mouse* is set to "*Selecting Residues*" (voice in the bottom right). By repeatedly clicking on this field (on *Residues)* you can redefine the selection as Molecules*, Chains, Atoms, Segments,* and so on.
Clicking instead with the middle button on a residue will center the view on it.

>> Let's click a residue with the middle key and try to rotate the molecule with the left mouse button. What's happening?

You will notice that the center of rotation of the molecule is set to the point you have just clicked.

If we right-click, finally, on an unselected atom, a drop-down menu will appear, with a series of actions that can be performed (these are the same that you can find in the top panel of the *Internal GUI,* and they will be discussed when talking about this tool):



>> Right-click a residue, and select "residual -> zoom". What happened?

>> To return to the original view, right-click on a residue, and select "molecule -> orient". What happened?

>> To deselect residues, we click with the left button in the blank screen (black part of the external GUI).

Let's now return to the description of the lines that appear (*all, 1BJ4, sele,* and so on) in the top panel of the Internal *GUI.* **These lines describe the objects present and the actions that can be performed on them.** For example, if we want to make the 1BJ4 molecule disappear momentarily from the *viewer,* it is sufficient, with the left *mouse* button, to click on its name in the upper panel of the Internal *GUI.* To bring 1BJ4 back, click on the name again.

>> with the left *mouse* button, we click the name "1BJ4" in the top panel of the Internal *GUI.*

>> To bring 1BJ4 back together, let's click the name again.

There is a row of buttons next to each object in the *Internal GUI.* Thanks to them:

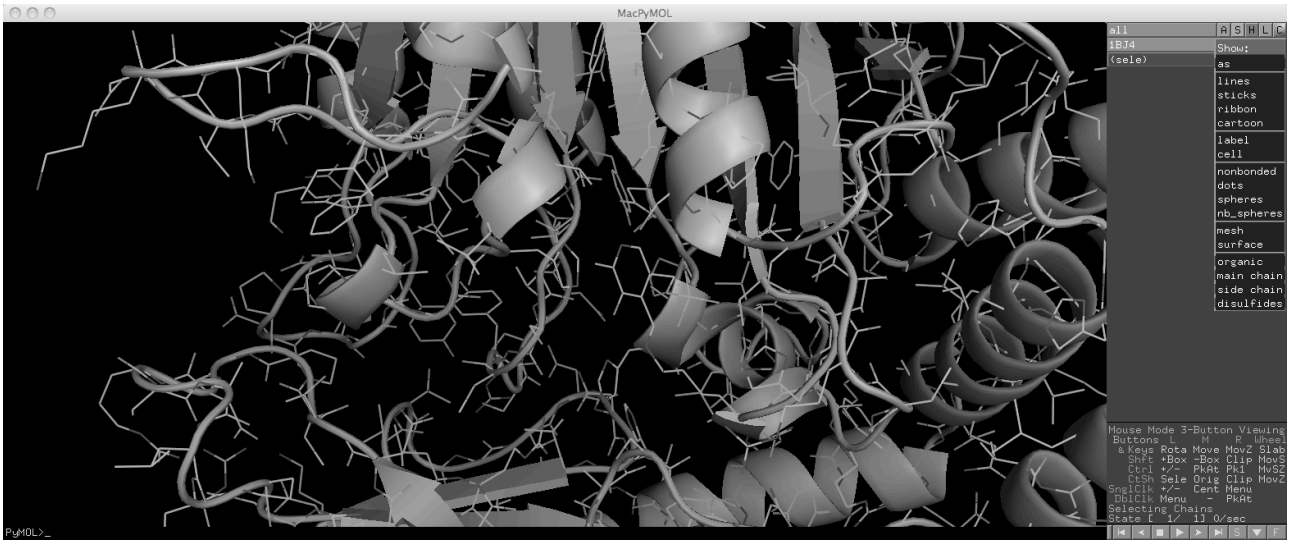
- you can access the actions that are performed on the object (**A** key, *actions*),
- change the object representation **(S** button, *show*),
- hide object representations **(H** key, *hide*),
- apply "labels" on the object **(L** key, *label*),

- color the object (**C** key, *color*; note that the basic colors of the proteins are green for Carbon, red for Oxygen, blue for Nitrogen, yellow for Sulfur, white for Hydrogen atoms and orange for Phosphorus, if any)

Of course, it's impossible to explore in detail all the actions and changes that can be made by these buttons in this tutorial. We will then see, below, only a few common actions.
Let's start with the representations: PyMol allows you to visualize the molecule in many styles, each customizable through the Setting entry from the *External GUI* (in particular, *Edit all...*). For example, we represent 1BJ4 as "***Cartoons***", selecting the related item from the S key menu:

>> We select the Cartoons entry by pressing the "S" button (show) next to the name "1BJ4"

This mode displays the molecule's secondary structures, helices, beta filaments and loops, as "cartoons", which follow the progress of the polypeptide chains.

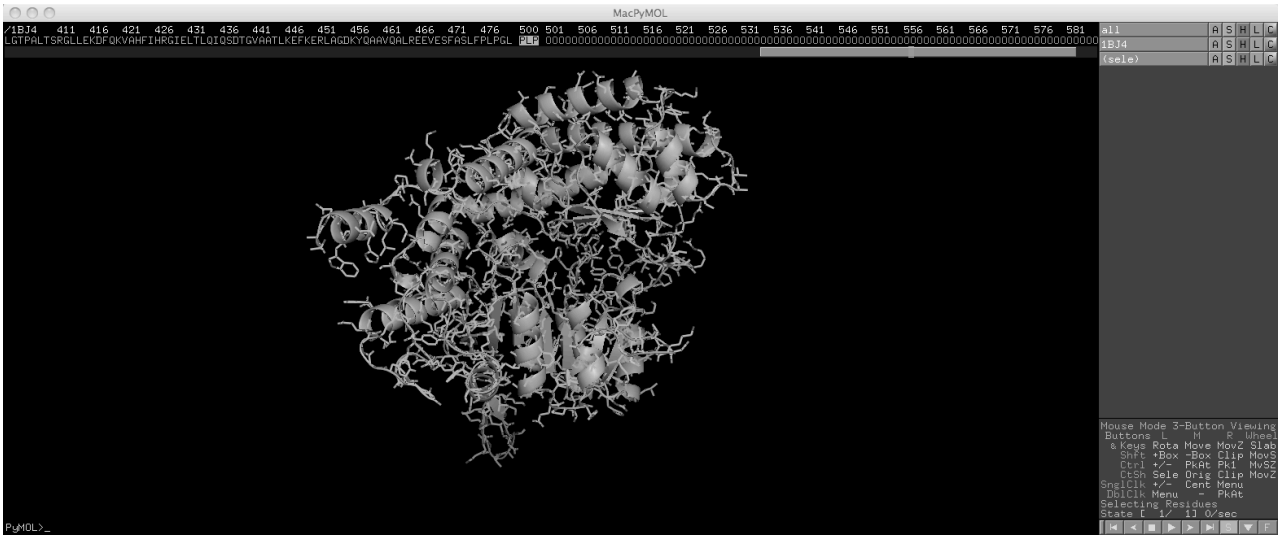


We now want to represent the PLP cofactor as sticks. First, to help identify the PLP, we open the window with the sequence of 1BJ4, pressing the "S" key in the lower right, in the lower panel next to the actions on the movies:

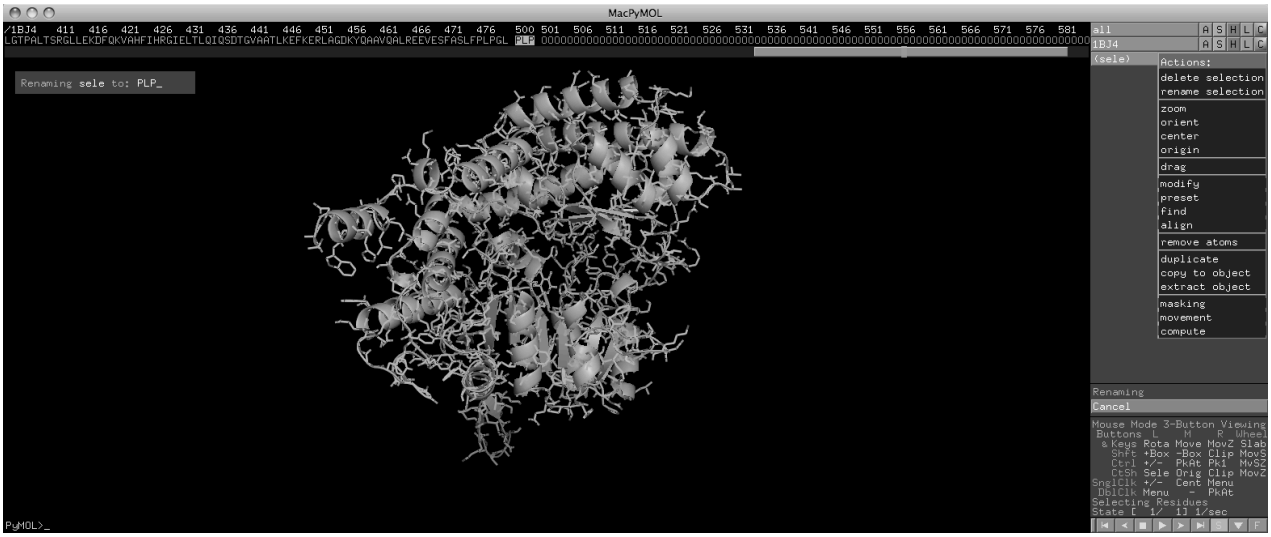
>> We click with the left mouse button the letter "S" at the bottom right of the external GUI. What's going on?

>> Let's scroll to the right by clicking on the gray bar, and holding down the mouse button, until we identify the residue indicated as "PLP" (position 500).

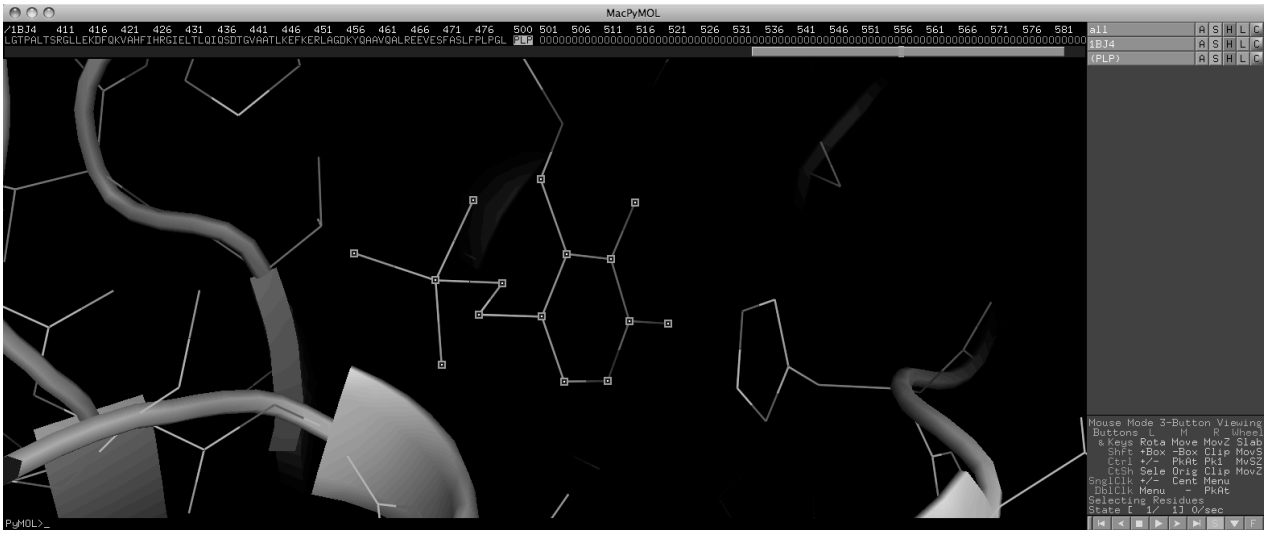
>> At this point, we select the residual "PLP" with the left *mouse* button:



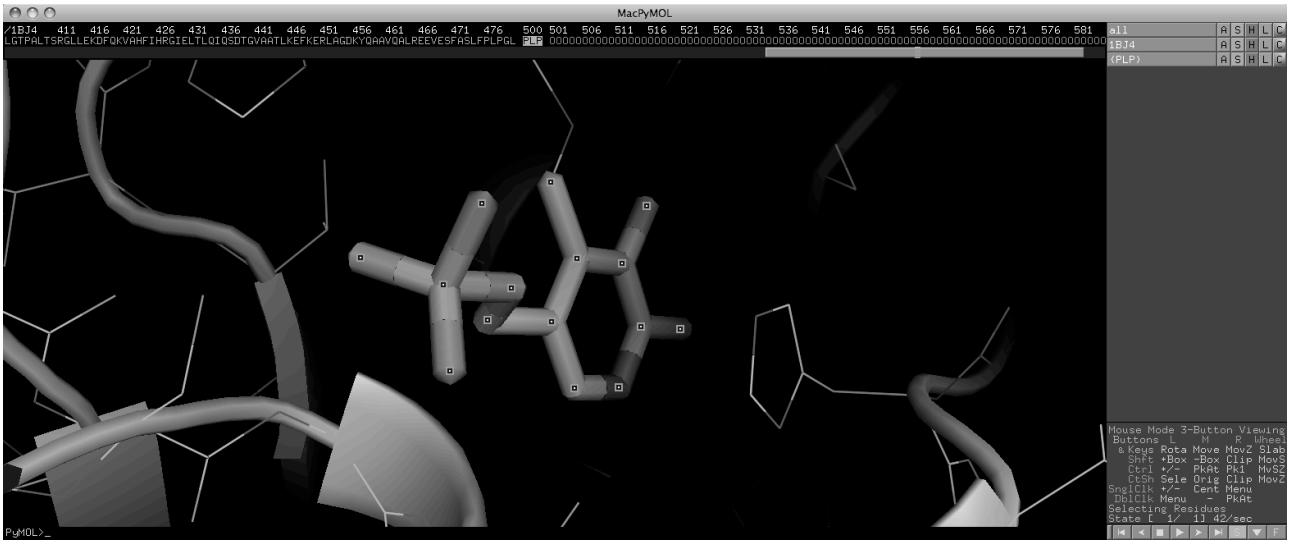
>> We rename the selection that appeared in the top panel of the *Internal GUI* as "PLP", clicking on the "A" button next to the *"sele"* item and selecting the "*rename selection"* option. In the field that appears, we enter PLP and type "Enter":



>> Now, by clicking on the "A" button next to the "PLP" entry in the top panel of the *Internal GUI* and selecting the *"zoom"* option, we focus on the PLP cofactor:

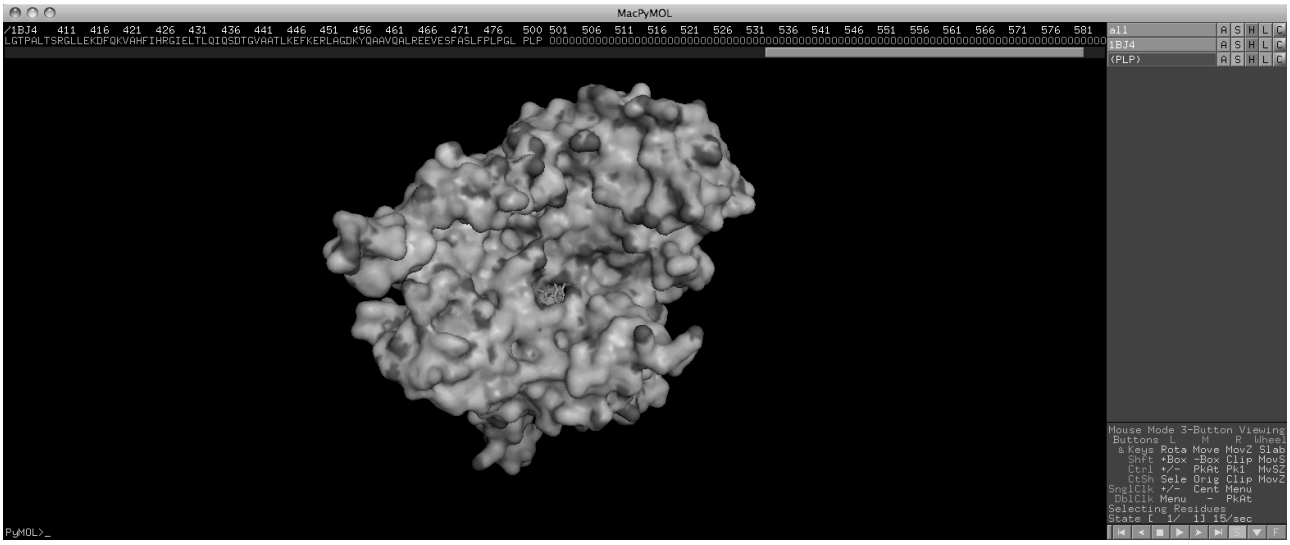


>> Finally, by clicking on the "S" button next to the "PLP" entry in the top panel of the*Internal GUI* and selecting the *"sticks"* option, we change the representation of the PLP:



PyMol also allows you to represent the Van der Waals spheres of molecules and the surfaces accessible to the solvent.

>> By clicking on the "S" button next to the "1BJ4" entry in the top panel of the *Internal GUI* and selecting the *"surface"* option, we change the macromolecule's representation:

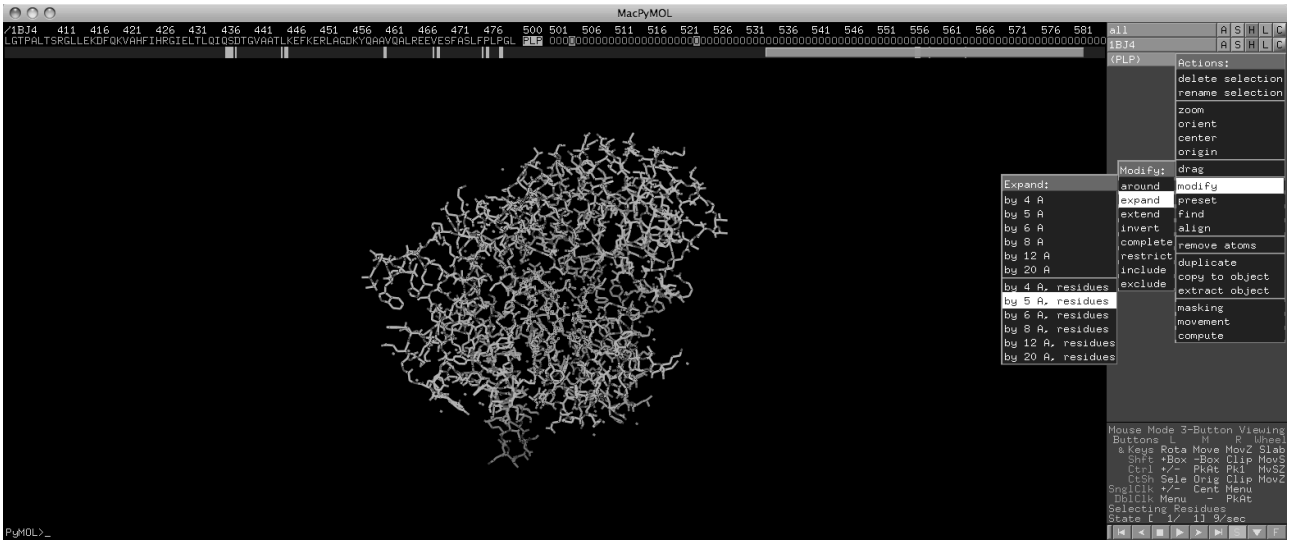


>> We click the "S" button next to the "PLP" entry and selecting the*"spheres"*option, we change the PLP representation.

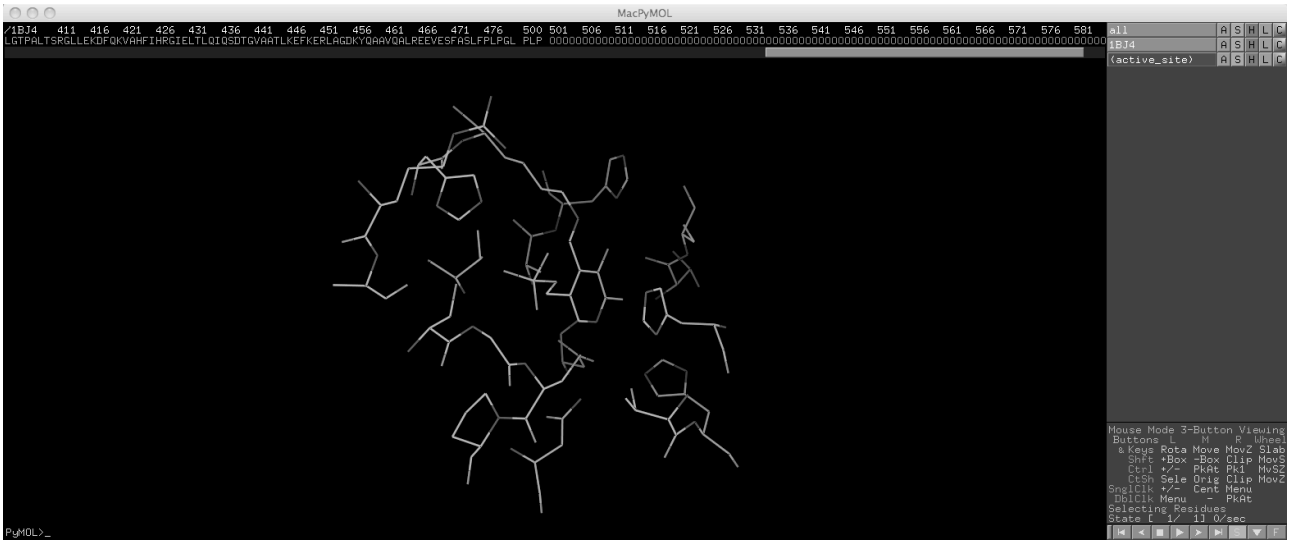
>> Now, to return to the initial view, we click the "A" button next to "1BJ4" and select the *preset->default* options (the *preset* panel contains other common display types. I invite you to explore them. When you are finished, return to the *default* view.)

Especially when studying enzymes, it can be very useful to identify and analyze in detail the **active site and the residues present in it**:

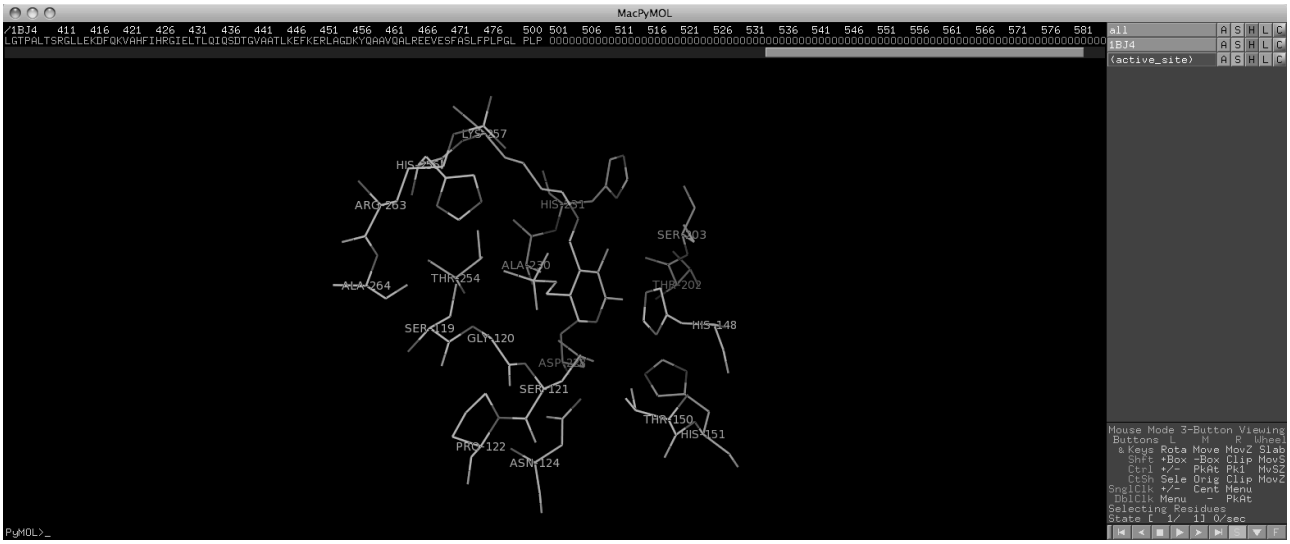
>> By clicking on the "A" button next to the "PLP" entry and selecting the option *"modify"*, then *"expand"*, and next *"by 5 A, residues"* we modify the selection "PLP" so that it includes the neighboring residues (at least one atom of them must be no more than 5th from the PLP). We can then rename the selection as "*active_site*":



>> Now, we hide 1BJ4 by clicking on the "H" button next to "1BJ4" and selecting the *"everything" option*, and only display *"active_site"* (by clicking on the "S" button next to the "active_site" entry and selecting the "lines" option). Then, we frame only *"active_site"* ("A" key next to the entry *"active_site"* and selection of the *zoom* option):

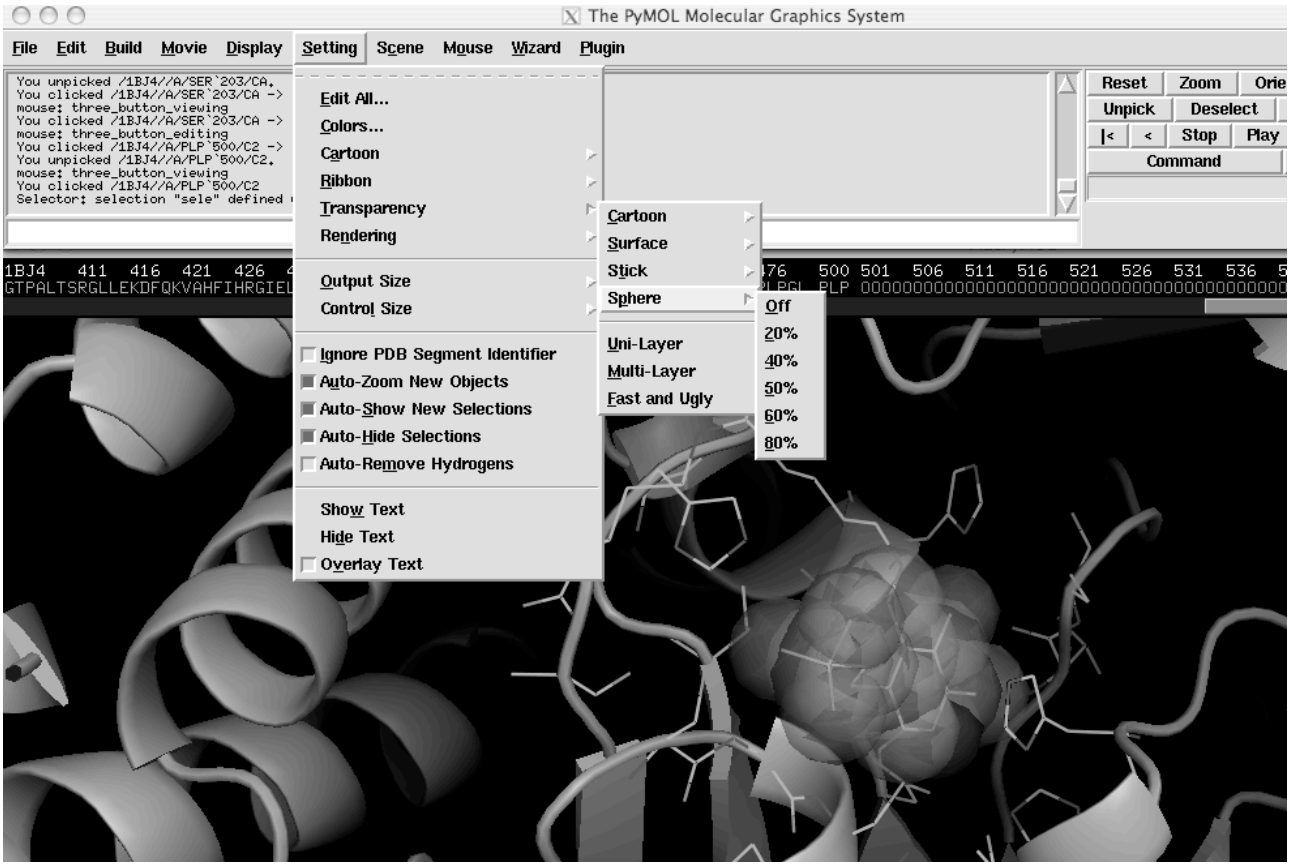


>> What's left on the active site? To answer this question, we click the "L" button next to *"active_site"* and select the *residues* option:



We now want to make a beautiful image of the active site for a presentation or publication.

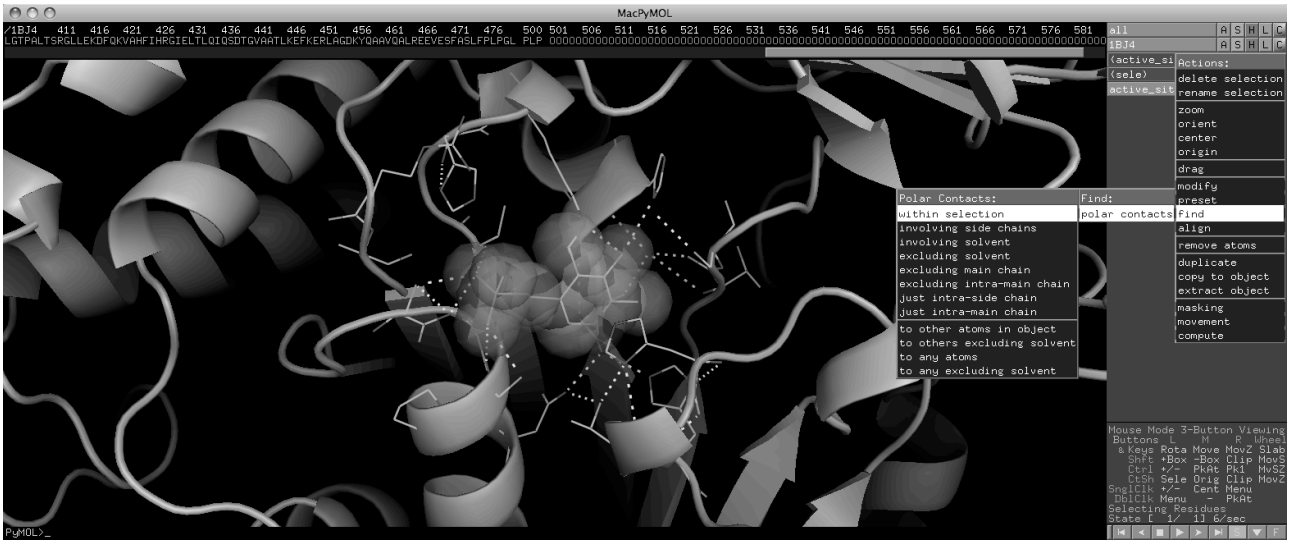
>> Let's re-icon 1BJ4 *cartoons* ("S" -> *cartoons*), hide the residual labels ("L" -> *clear*), we reselect the PLP with the left mouse button, represent it as *spheres* and finally from the *Setting* item of the Menu of the *External GUI* we select, as shown below, the transparency of the spheres to 50%:



If we wish, we can also highlight the polar contacts that the PLP establishes with the residues of the active site. To this purpose:

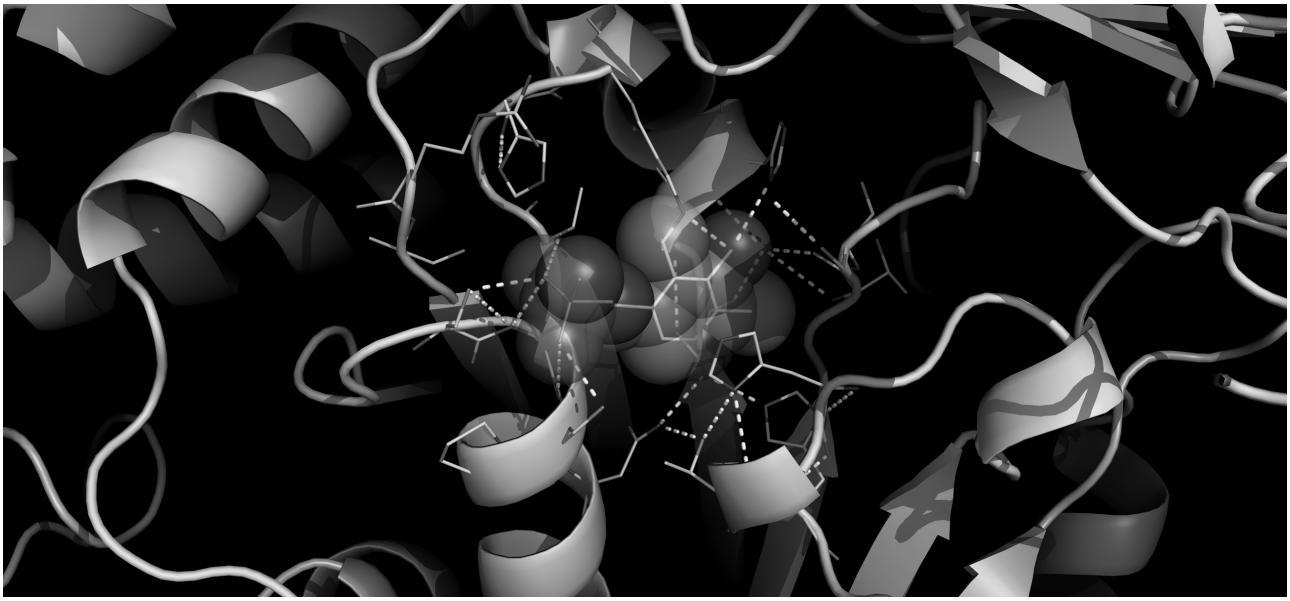
>> We click the "A" button next to the *"active_site"* entry and then find -> *polar contacts -> within selection*).

Interactions will appear as yellow dashes, and a new object will be created in the menu on the right:



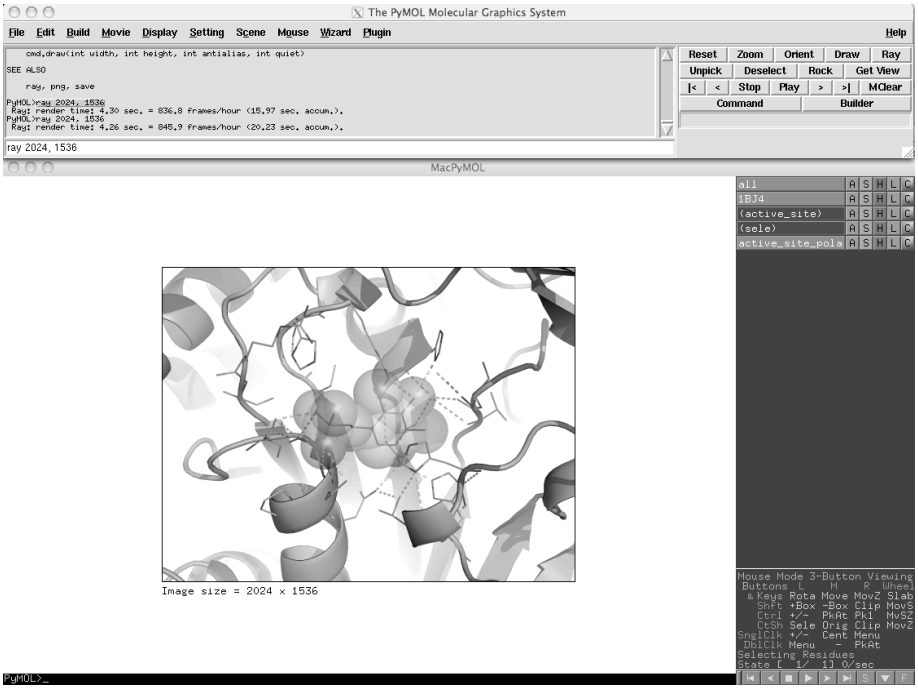
If we are satisfied with our representation, we save the work done to avoid having to repeat it:
>> We select *"Save Session as ... "* from the *File* entry of the External *GUI* (top left), then we choose the *file* name in the window that will appear, for example "work.pse" (files with PyMol's sessions have a ".pse" extension).

>> Finally, we generate a figure: from the buttons at the top right of the *External GUI* we press the *"Ray"* button and wait a few seconds. The image will be subject to a*"rendering"* process that will refine its details. We then save the image by selecting Save Image *as... -> PNG...* from the *External GUI Files* entry and then choosing the *file* name in the window that appears:



If the image must be high-resolution, you can also use the *text-ray* command in the *Command Input Area* (for more information about using the command, refer to the official guide, or type within the

help ray area). For example, if we wanted to generate an image of 2024 x 1536 *pixels,* we would write the command *ray 2024, 1536*:



The white background was generated through the *Background -> white* command accessible from the *Display* menu.

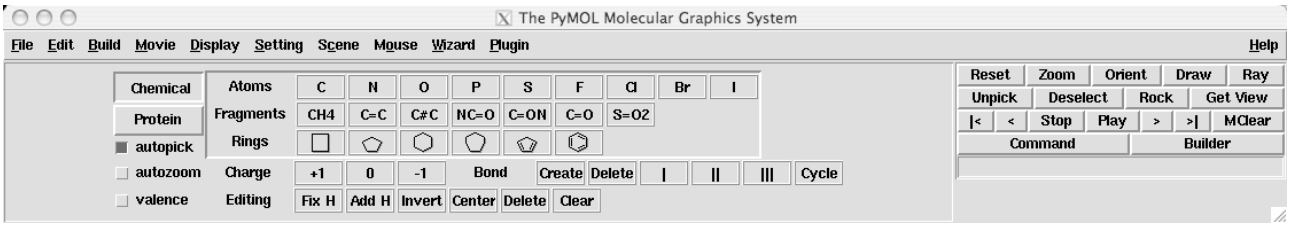
>> To return to the initial window and clear the molecules on the screen, we select the *Reinitialize* command accessible from the *File* menu.

In the next paragraphs, some PyMol functions commonly used in molecular modelling will be described.

PYMOL AS BUILDER

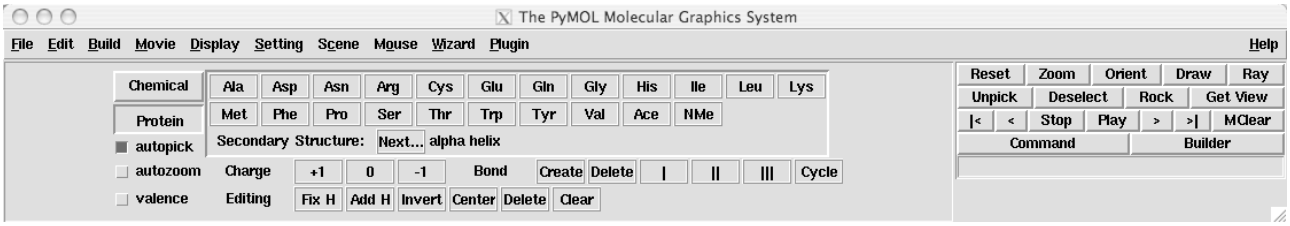
Pymol can also be used to build and model peptides and small organic molecules. In the following example, we will build a beta strand made up of Alanine residues.

>> We click the *"Builder"* button located to the right of the External *GUI.* The latter will be modified, and the *mouse* will enter *Editing* mode (as indicated at the bottom of the Internal *GUI*):

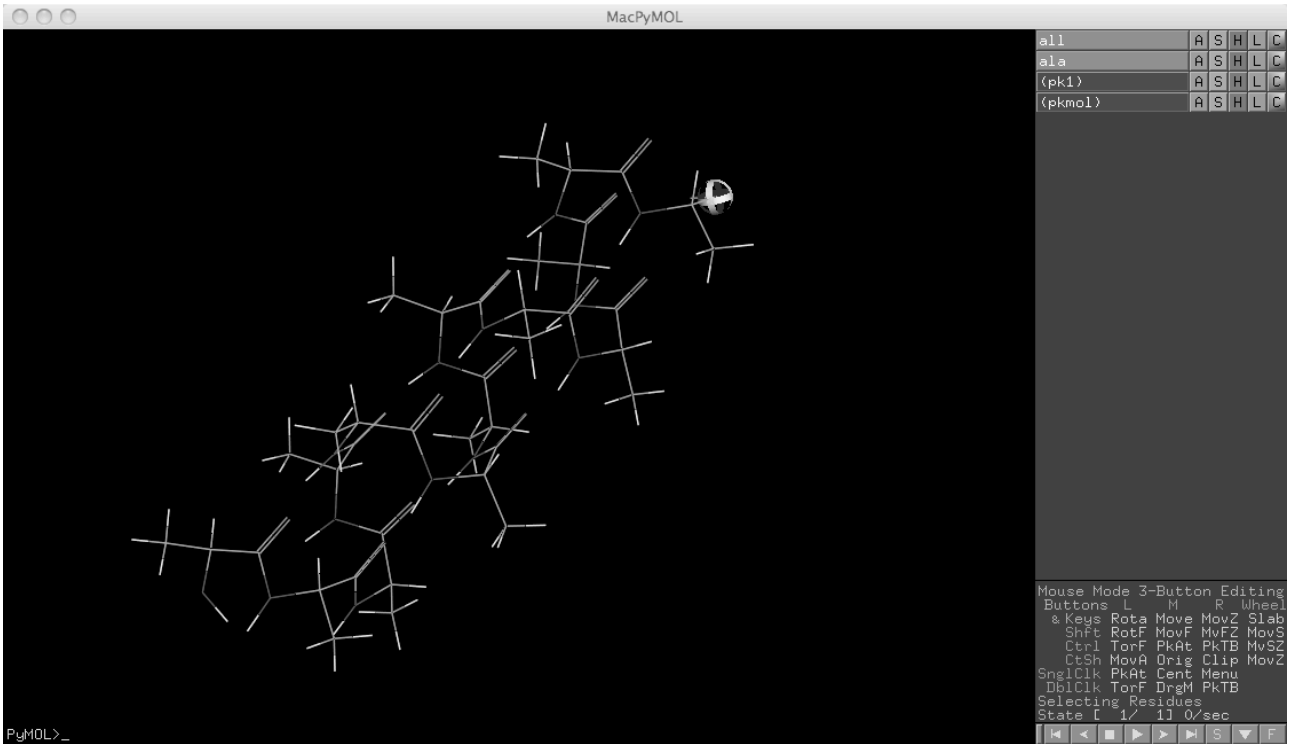


In the window that appears, you can select common atoms and fragments for the construction of small organic molecules. It is also possible to change the charge, binding types and protonation of the generated structures.

>> By clicking on the "Protein" button at the top left, we instead access the menu for the construction of polypeptides:

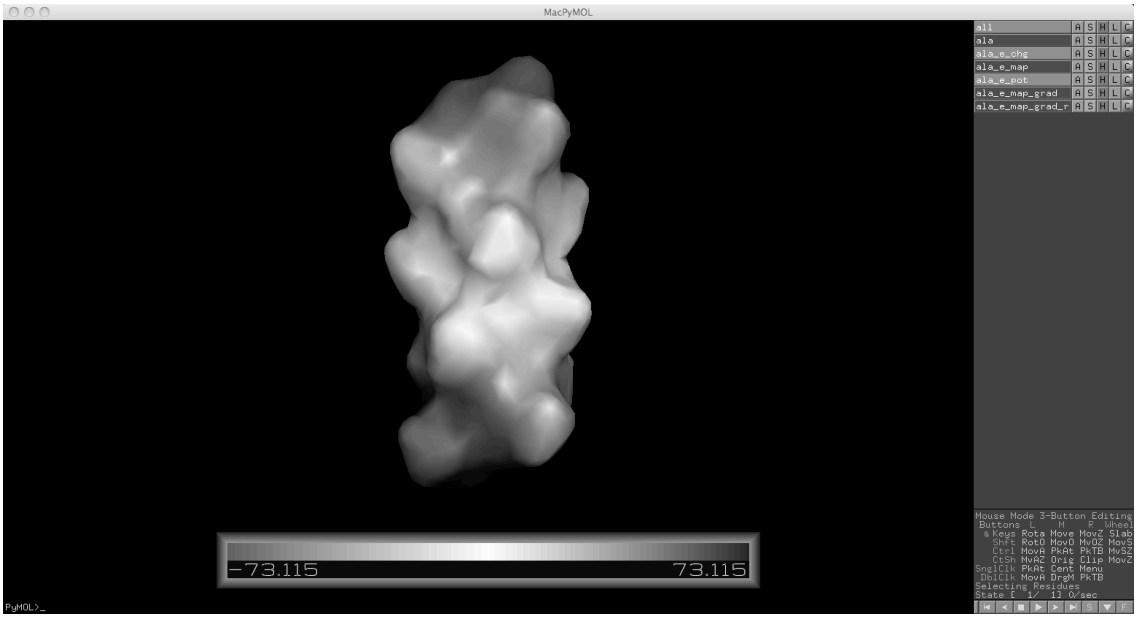


>> Click the *"Next... "*, under the "Secondary *Structure*" entry, and select "*strand*". Then, by repeatedly clicking on the molecule, we generate a strand consisting of Alanine residues (we add at least five residues):



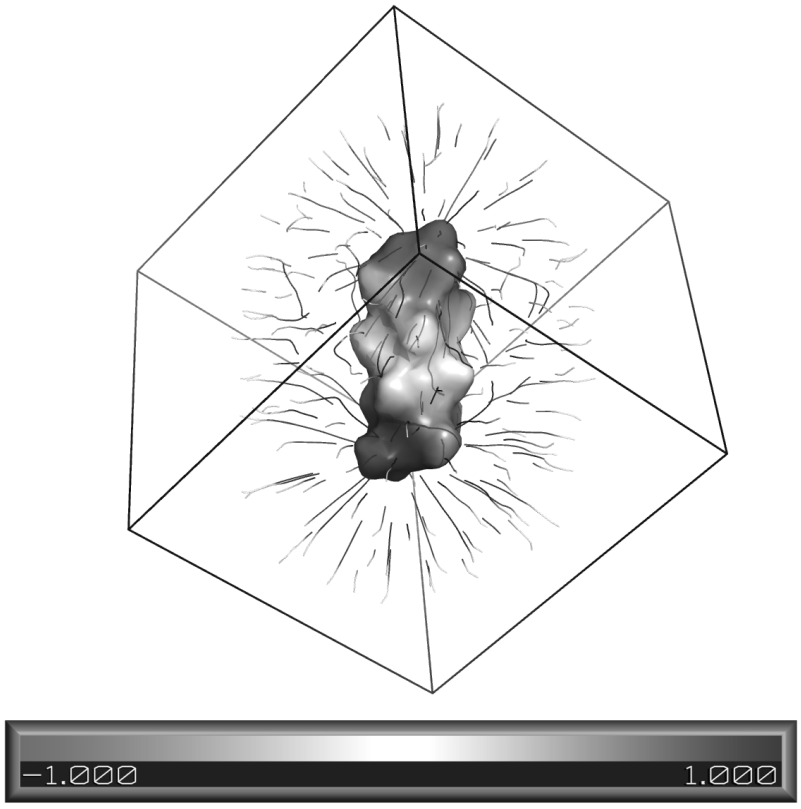
The sphere that appears in the *viewer* indicates where the new residue will be attached, through peptide binding. Along with the newly created object, two selections are automatically generated, *pk1* (*pick1*, relative to the atom indicated with the sphere) and *pkmol* (*pick* molecule , relative to the entire molecule).
The molecule is characterized by a moment of **dipole,** due to N- and C- termini. You can visualize this separation of charge by mapping its distribution on the surface accessible to the solvent of the molecule:

>> We select the "A" key at the strand and generate -> *vacuum electrostatic -> protein contact potential*:



The positively charged surface is colored blue, the one negatively charged in red. The scale, in units of *kT*/e, is shown atthe bottom, and a new object, "*ala_e_pot*", is associated with it. By clicking with the middle mouse key, holding down the *"ctrl"* key on the keyboard on the scale and moving left and right, you can vary the minimum and maximum values of the scale. Another object created is "*ala_e_map*", which allows you to analyze the electrostatic field generated by the molecule with different representations.

>> For example, we select the "A" key at *"ala_e_map"*and gradient -> *default* below to display the gradient of the electrostatic field:



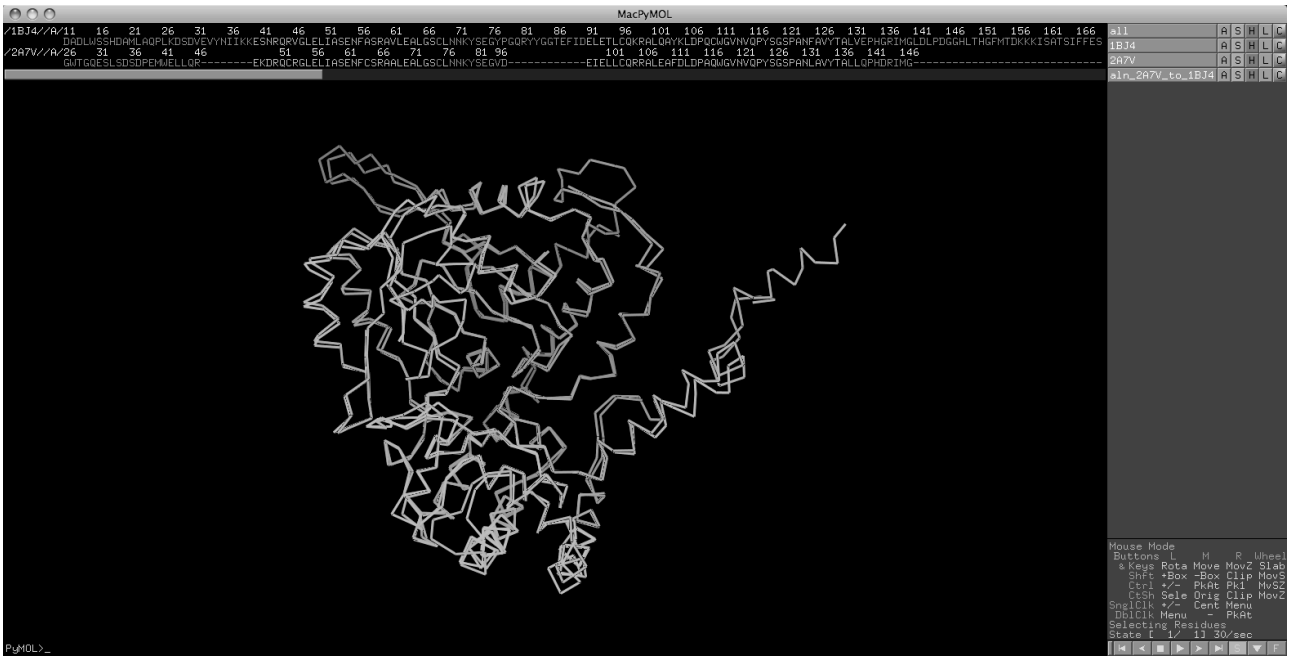
ALIGN AND SUPERPOSE TWO MOLECULES WITH PYMOL

A PyMol function that is often useful in comparing homologous proteins (related proteins, which have an ancestral protein in common) is the alignment of two sequences and, on the basis of the latter, the overlap of corresponding protein structures.

>> We import into PyMol structures with pdb codes 1BJ4 and 2A7V (human hydroxymtyltraspherese human cytosolic and mitochondrial, respectively), using the *PDB Loader Service* "Get PDB…".

>> We open the window with the sequences of 1BJ4 and 2A7V, pressing the "S" key in the lower right, in the lower panel next to the actions on the movies.

>> Next, we select the "A" key at "1BJ4" and below align -> *to molecule* -> *2A7V*, to display the aligned sequences and overlapping protein structures (to simplify the representation, we can hide the lines and display only the skeleton of the α-carbons, selecting from the "S" of *"to*" the item "*Ribbons*":



To perform a sequence-independent structural overlap, which is useful when proteins have dissimilar sequence, you can also use the *cealign* text command in the *Command Input Area* (for more information on using the command, refer to the official guide). For example, if we wanted to generate an overlap between 1BJ4 and 2AV7, we would write the *command* cealign *1BJ4, 2AV7,* and type the "Enter" key.

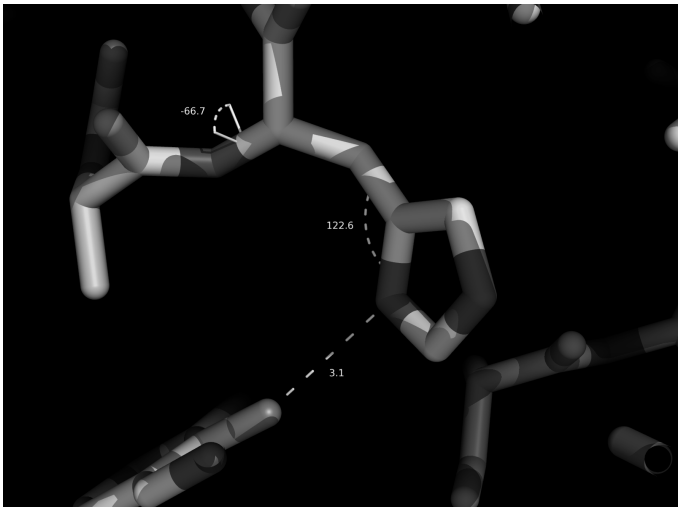
PYMOL'S "WIZARD" menu **(OPTIONAL)**

Through PyMol's *"wizard*" menu, advanced structural analysis tools can be accessed, including, for example, the measurement of binding lengths, distances, angles, the overlap of three or more pairs of atoms, the in silico mutagenesis of a residue, or the *sculpting* of a molecule.

>> Selecting, for example, "*Measurement*", a new menu appears in the Internal *GUI.*

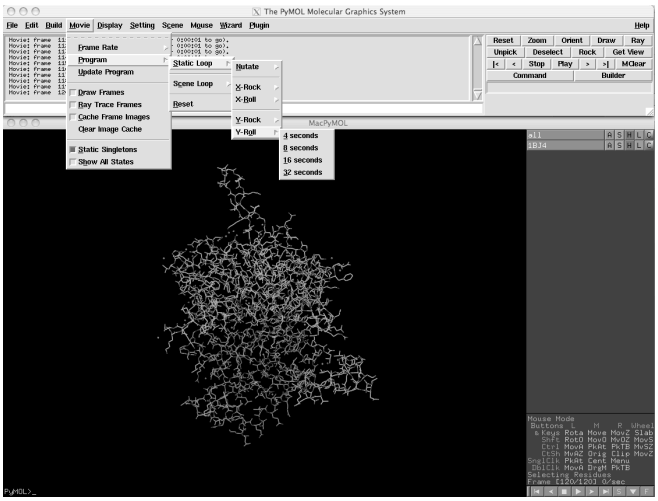
>> By clicking with the left *mouse* button on the first item of the new menu, you can select the type of measurement to make (distances, angles, dihric angles and so on). After selecting one of these entries, we will be guided by PyMol in the choice of atoms in relation to which you want to make the measurement.

For example, the following figure shows measurements of distances, angles and dihedral angles:



GENERATE MOVIES WITH PYMOL (OPTIONAL)

Using *scripts,* you can generate quite complex movies with PyMol. In the case of simple rotations or oscillations, however, PyMol provides tools that do not involve the use of *scripts.* PyMol allows you to export the movie directly as a .mov (Quicktime) or MPEG *file.* In other operating instances, PyMol can generate a series of consecutive static images *(frames),* which can then be assembled into movies using external programs (**ImageMagick,** for example).
If, for example, we want to generate a movie with the rotation of a molecule (we will use 1BJ4 again), we select, from the *external GUI*menu, *Movie -> Program -> Static Loop -> Y-Roll -> 4 seconds*:



At this point, we select the *"Play"* button from the actions on the movies, at the bottom right. The molecule will start to rotate relative to the Y-axis, making a full turn in 4 seconds. To save the movie, we select "Save *Movie...* "file" of the *External GUI* menu.

USING PYMOL SCRIPTS (OPTIONAL; REQUIRES PYTHON LANGUAGE KNOWLEDGE)

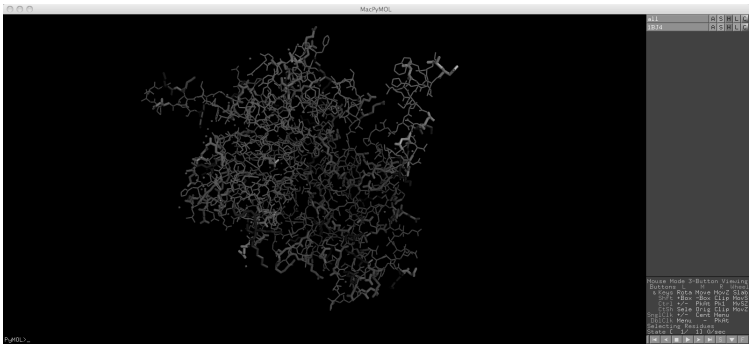
You can add new features to PyMol, through the Python scripting lnguage. In the simplest cases, simply embed a Python *script* as a PyMol command, which can be run from the Command *Input Area.* In the following example, we'll use a short *script* that represents certain types of residues (Arginine, Lysine, Aspartate, and Glutamate) with user-determined colors.

>> Let's open a text *editor* and write the following Python code:

```
def colora_carichi():
        cmd.set_color("rosso", [1.0,0.0,0.0])
        cmd.set_color("marrone", [0.7,0.4,0.0])
        cmd.set_color("blu", [0.0,0.0,1.0])
        cmd.set_color("azzurro", [0.0.0.4,0.7])
        cmd.set_color("gray", [0.3,0.3,0.3])
        cmd.color("gray", "all")
        cmd.color("rosso", "resn glu")
        cmd.color("marrone", "resn asp")
        cmd.color("blu","resn lys")
        cmd.color("azzurro","resn arg")
        carichi = "resn glu+asp+lys+arg"
        cmd.show("sticks", carichi)
cmd.extend("colora_carichi", colora_carichi)
```

>> Save the *file* as "colora_carichi.py".
As you can see, the *script* defines a function, *colora_carichi()*, within which some PyMol commands (*cmd.set_color*) are defined, which defines a color on a scale of type RGB; *cmd.color*, which colors a selection of PyMol with the color given as an argument; *cmd.show*, which shows the selection as *sticks).* When the *script* completes, the *cmd.extend* command allows you to use the newly generated function as a Pymol command.

>> To import the command into PyMol, we select *"Run Script...* "from the *"File"* item on the *External GUI*menu, and select the *file* "colora_carichi.py".
At this point, simply use the text command *colora_carichi* in the *Command* Input *Area* to launch the newly written function:



CONCLUSIONS
PyMol is a great tool for molecular graphics and modelling. Perhaps the most fascinating aspect of PyMol is the simplicity with which you can extend it with your own *scripts* and *plugins.* This attribute has given rise to a vibrant community of users and programmers who provide their time and skills to constantly extend and improve PyMol. You can get to know and join this community by visiting the *http://www.pymolwiki.org/* website. Good navigation!

Warren L. DeLano, creator of PyMol and a firm supporter of the "Open Source" philosophy, died on November 3, 2009 at the age of 37. A heartfelt thank you to Warren DeLano, for the remarkable contribution made to scientific research and teaching.

Alessandro