Moviemaking in PyMOL

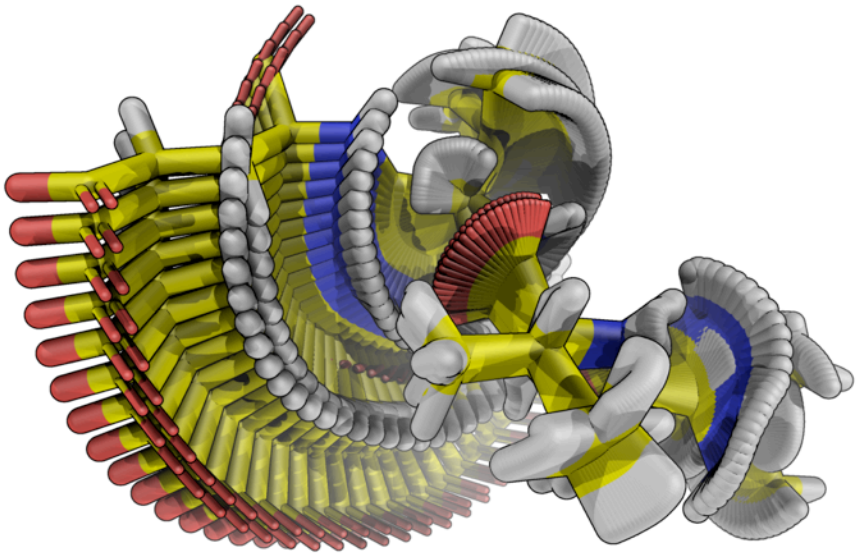

Table of Contents

Introduction

Copyright Notice

About This Booklet

This is a follow-along guide for the *Moviemaking in PyMOL* classroom tutorial taught by Schrödinger. It covers moviemaking through scenes, object motions, molecular dynamics trajectories, and molecular morphing. Upon completion, the user should feel comfortable with the basics of moviemaking in PyMOL including creating and export movies to MPEG or AVI files.

This tutorial was created for PyMOL version 1.2 or greater running under Windows, Mac, or Linux.

Requirements

Before starting this tutorial, please be sure that you have the following.

- PyMOL version 1.2 or greater
- A 3-button wheel mouse
- Tutorial-specific data files found in the **Moviemaking** subfolder of the **PyMOLTutorials** archive supplied by Schrödinger. This archive is typically provided as a compressed “ZIP” file, which should be extracted on to your Desktop

Read and understand, or attend the Introduction to PyMOL classroom tutorial taught by Schrödinger.

Be sure that your keyboard Caps Lock key is turned off when using PyMOL.

Intended Audience

This tutorial is intended for users who have completed the one-hour Introduction to PyMOL course or are familiar enough with PyMOL to type commands, read and run simple scripts, and easily navigate with the mouse.

Creating a Movie from Scenes

PyMOL allows us to create visual snapshots, called scenes. In the *Introduction to PyMOL* tutorial, we created scenes that we stored and displayed in **show** files.

The user defines the order of the scenes in **shows.** Then PyMOL creates a smooth transition by interpolating a camera motion that takes us from each scene to the next. This interpolation of the camera's viewpoint as it moves between scenes maintains the visual context of the molecular structure, which is important information for the user. Scenes, and interpolations of motions between scenes, are also useful for creating movies. However, there are differences between PyMOL **shows** made from scenes and PyMOL movies made from scenes.

Recall that a **show** is a series of scenes that the user displays by pressing the `PgUp` and `PgDn` keys. The user controls the length of the intervening time between the scene transitions. During a **show**, the user may even interrupt the ordered scenes by just interacting with PyMOL in the normal way, say by zooming in on another part of a structure. This is possible because a **show** file is identical to a PyMOL session file, except that it opens with a full screen display. After an interruption, a PyMOL **show** can be continued unperturbed. A movie, on the other hand, can be rendered out and stored as a PyMOL-independent movie file, a simpler type of file that the user just watches. There is no interaction with the molecules in a movie file because it is just a collection of images. Nonetheless, PyMOL movies are developed in PyMOL sessions, and reside in development format within PyMOL sessions.

A Brief Review of Scenes and Scene Buttons

A **scene** in PyMOL is simply a memory-snapshot, in which PyMOL saves current colors, visualizations, representations and the position of the camera. In PyMOL, we store scenes by selecting **Scene→Append** from the Upper Control Window or by pressing `CTRL-PgDn` on the keyboard. We recall scenes by selecting **Scene→Previous/Next** from the Upper Control Window or by pressing `PgUp` or `PgDn` for the previous and next scene, respectively.

PyMOL has a new feature called Scene Buttons. Please click on **Scene→Buttons** from the Upper Control Window. When you create a scene, a button will now appear for each scene. You can click on a Scene Button to immediately jump to its scene. You can also rename, update and delete scenes by right-clicking on the corresponding buttons. Figure 1 shows three Scene Buttons; they appear in the lower left hand corner of the Display Area. The first scene is named "001," the second "zoomProtein," and the last "zoomLigand." To move a scene's position in the stack, right-click and drag the scene button to its new location. Because the zoomProtein scene is active, it appears in light grey. These buttons act as a simplified storyboard for your movies.

Figure 1: Scene Buttons.

Creating a Scene-based Movie

Here we show you how to make a movie using scenes in a few simple steps. First, setup and save each scene you want to visit. Then, from the menu cascade, **Movie→Program→Scene Loop** in the Upper Control Window, select your desired movie type. That's it—now just press the play button to watch your movie. You can even export that movie to an MPEG (or additionally, a QuickTime file with MacPyMOL). Let's take a look at that in detail.

For our movie, we create a series of scenes in which each one focuses the camera on a different part of a protein complex. We begin by viewing an entire protein structure shown as cartoon. Then, we focus on the ligand binding pocket and highlight its features by showing it as a surface. And, to distinguish the ligand from the rest of the structures, we show it as sticks. Once our scenes are prepared, we create the movie with just a couple of clicks of the mouse. With a couple more clicks, we save the movie as an MPEG file to watch or insert into PowerPoint.

Scene 1: Full View

Please launch PyMOL and load the file titled "2zvj.pdb" from the **PyMOLTutorials/Moviemaking/pdbs** directory. Please show it as a cartoon by selecting **S→Show→As→Cartoon**. Please save this as our first scene by selecting **Scene→Append** from the Upper Control Window. The Scene Button "001" appears. Please right-click on the scene button and rename it to "fullView."

Scene 2: The Binding Site

Now, please make a selection for the ligands, which are organic small molecules, by selecting, in the Viewer Window, **A→Action→Generate→Selection→Organic** for the 2zvj object. PyMOL creates the "(2zvj_organic)" selection. Orient on this selection by clicking **A→Action→Orient** and then disable it by clicking on its name. Save this as the second scene, again by clicking on **Scene→Append**. The Scene Button "002" appears. Please right-click on this scene button and rename it to "zoomSite."

Scene 3: Showing the Ligand

Let's turn on the sticks representation for the organic selection. To do so, please click **S→Show→As→Sticks** for the organic selection. Save this as our third scene. Please rename this scene to "ligSticks." While we did not move the camera, this is still a new scene because we have a change in representation: the ligand is now being shown as sticks.

Scene 4: Showing the Binding Pocket

Now we want to show the binding pocket as a surface in which the ligand atoms are embedded. To do this, we make a new selection, by copying, renaming, and modifying the original 2zvj_organic selection into a new selection that contains only those *protein atoms* within 4 Å of the ligand.

Please duplicate the organic selection and rename it to "(pocket)". Now, let's modify "(pocket)" to contain the residues within 4 Å of the ligand. For "(pocket)" please click on **A→Action→Modify→Around→Residues within 4 Å**. Please turn on the surface representation for the (pocket) selection by choosing its **S→Show→Surface**. Part of the ligand is hidden by the surface, so let's turn on transparency for the surface: click **Setting→Transparency→Surface→40%**. Please color the pocket "grey60" by clicking on **C→Color→Grays→Gray60** for the (pocket) selection. Using your mouse, please rotate the view to your liking. Please **Scene→Append** this last scene and name it "viewPocket". This step corresponds to the far right image in Figure 2.

At this point you should have four scenes that look similar to those in Figure 2, and you should be able to switch between them by pressing the `PgUp` and `PgDn` keys on your keyboard.

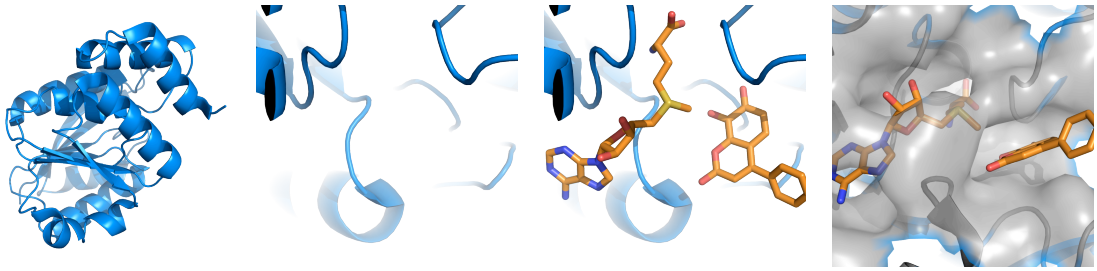

Figure 2: Screenshots of the four scenes created for our first movie.

Create the Scene-based Movie

At this point, we use PyMOL to stitch the scenes together into a movie. To seamlessly loop through our scenes, pausing to display each one with a motion that helps us to see the three dimensional space, please select **Movie→Program→Scene Loop→Nutate→2 Seconds Each**. Now press the play button in the movie controls. PyMOL plays the movie, nutating at each scene for two seconds. Please stop your movie.

Saving A Movie

Before we export our movie to an MPEG file, let's ensure the safety of our work by saving this PyMOL session to disk. Please select **File→Save Session As** and save the current session to your **Desktop** as **movieSession.pse**. Now, if PyMOL crashes during the movie production step we can just reload the session file and continue where from where we left off.

To export the movie, select **File→Save Movie As→MPEG** from the Upper Control Window. Please name the file "bindingPocket" and save it on your Desktop. Don't worry, PyMOL will add the ".mpeg" extension for us. While PyMOL produces your movie, please do not interact with PyMOL or the temporary folder in which PyMOL is working. This could corrupt your movie. Once PyMOL finishes compiling your movie, it will write the message, "`produce: finished.`" At that point, you may continue working in PyMOL.

If an appropriate media player is installed on your computer, then double-clicking on the icon should play the movie. You can also import this movie into PowerPoint. PyMOL writes the movie to the current working directory unless you specify the full path for the movie. To find the current working directory for PyMOL, please type `pwd` into the command line.

The Movie Timeline & Frames

While your movie played inside of PyMOL, you may have noticed a little gray rectangle passing along the bottom of the PyMOL Viewer Window. That is the movie slider, located in the Movie Panel. Figure 3 shows an annotated Movie Panel. The movie slider indicates which **frame** PyMOL is currently displaying. A **frame** in PyMOL is just like a frame of a movie reel. It is snapshot taken after a small unit of time in which the scene may have changed. Some frames are special, and they can influence what happens in surrounding frames. These **key frames** are defined as frames in which the camera or object matrix undergoes a change. (The object matrix encodes the differences in translation and rotation for an object. Discussion of motions matrices is beyond the scope of this text.) Key frames are shown as blue boxes on the timeline. Key frames are still just single frames, so they only take up the space of

one key frame on the timeline. Do not be confused into thinking that multiple key frames adjacent to each other, like the first 40 frames of our movie, are actually one large key frame.

You can jump to any given frame by typing `frame X`, where X is the frame number you want to visit, for example `frame 45`. You can find the frame number and object state (discussed in section, "Animating a Trajectory") directly above the movie controls. See Figure 3. The default frame rate in PyMOL is 30 frames per second, but can be changed with the **Movie→Frame Rate** menu options in the Upper Control Window. So, one frame by default lasts 1/30th of a second.

Left-clicking to the right of the slider advances it by one frame. **Left-clicking** to the left of the slider will step back one frame. You may also directly drag the slider by **Left-clicking** on it and dragging it. **Middle-clicking** on the timeline will place the slider on that frame. **Right-clicking** on the timeline will display a pop-up menu of movie options for that frame. In addition to the camera timeline, a timeline can be assigned to a molecular object if it is moving independently in relation to other objects. This allows us explicit, independent control of the camera and the object at each frame of the movie. If you right-click on the timeline *for the camera*, you get Camera Motion options. Likewise, if you right-click on the timeline for a *molecular object*, PyMOL displays a Object Motion pop-up menu for that given frame. Figure 3 shows these two menus.

When you use the **Movie→Program→Scene Loop,** PyMOL creates a movie by stitching together scenes. Interpolating from one scene to the next requires the camera to move so the camera timeline is made visible and keyframes shown. As long as the objects don't move with respect to each other, no timeline is displayed for any object. Again, a key frame is a just a frame where the given object or camera undergoes a change. They exist to help us control the transitions in our movies.

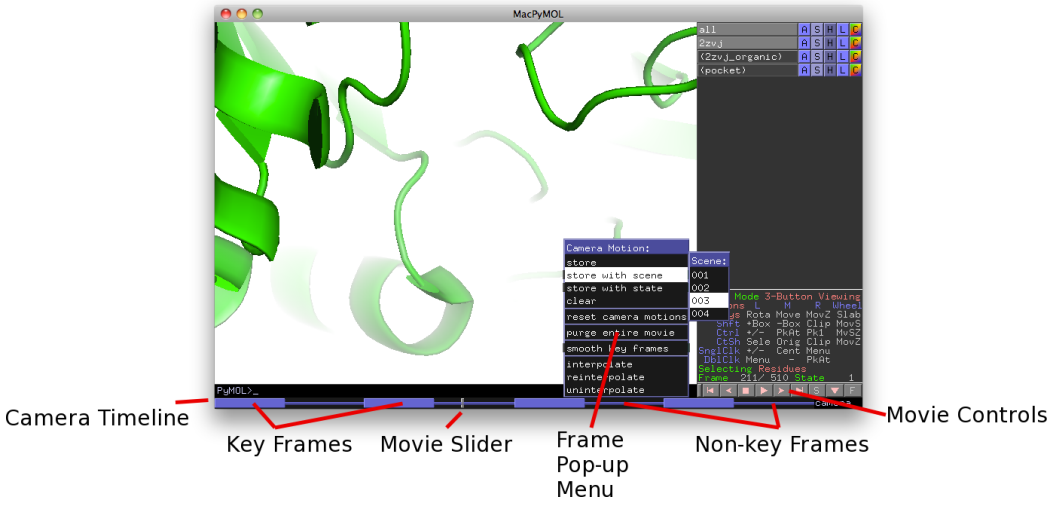

Figure 3: Annotated Movie Panel. The camera has its own timeline. The thick, blue colored frames in the timeline are series of individual keyframes. A key frame is an important frame where the object in question, here the camera, undergoes a change. Contiguous key frames, like in this timeline, look like one wide key frame, but are actually a series of small key frames. Single key frames are shown in Figure MM.2 in the Motions Chapter. Non-key frames are not highlighted, they are just shown as a thin blue line. Right-clicking on the timeline will pop-up a menu for that frame. The movie controls at right are used to rewind, stop, play, fast forward the movie. The frame and state counters are directly above the Movie Controls.

Using the Timeline to Insert and Delete Frames

If your movie is playing, please stop it. Now, let's practice interacting with the timeline and use it to modify a movie. We can insert time into the timeline—or more accurately inject a number of frames into the timeline at a given point. We do this by positioning our mouse over the frame after which we want to insert the time and then pressing and holding `CTRL` on the keyboard and dragging to the right with the left mouse button. This action is indicated in Figure 4B. The farther you drag the mouse, the more frames are added.

Insert Frames

Let's try that now. Please position your mouse over one of the first non-key frames, such as frame 65. Now, press and hold the `CTRL` key on the keyboard and press and hold and the left mouse button while dragging the mouse to the right. PyMOL outlines the new frames in green as you move the mouse. When you release the mouse button, new frames will be inserted. Now, replay your movie. The zooming in on the pocket will now be slower. PyMOL has many more frames in which it can zoom in on the pocket, so it takes longer.

Delete Frames

Please stop your movie. We will now remove frames (see Figure 4C). Please place your mouse near the last non-key frame that you just added. Now, press and hold the `CTRL` key on the keyboard and press and hold the left mouse button while dragging the mouse to the left. PyMOL highlights these frames in red. When you release the button, the frames in red will be deleted. Please leave a few non-key frames for the transition. Replay your movie. Now, because it has fewer frames in which to zoom, PyMOL zooms in much more quickly.

Figure 4A. Your timeline should look like this immediately after programming your movie.

Figure 4B. Adding time to the timeline. Left-click near the first non-key frame and `CTRL` left-click-drag to the right. PyMOL highlights the new time in green.

Figure 4C. Removing time from the timeline. To remove time from a movie, CTLR left-click-drag to the left. PyMOL highlights the frames to be removed in red.

Figure 4D. Frames have been removed. Notice how close the first two sections of key frames are.

Inserting Scenes Before/After Other Programmed Scenes

PyMOL allows us to insert scenes before and after the scene we are currently viewing. This is helpful in case we forget to insert a scene or want to add information when developing a movie. In our case, let's investigate the hydrogen bonds of the ligand to the protein. Let's insert this scene before the "viewPocket" scene.

Create the New Scene

First, please use `PgUp` or `PgDn` until you arrive at the “viewPocket” scene. Now let’s find hydrogen bonds. To do this, please click on **A→Action→Find→Polar Contacts→To other atoms in object**, for the (2zvj_organic) selection. Now, using your mouse, please zoom in on a hydrogen bond (shown as yellow dashed line) and clip out other atoms until you have a view you like.

Insert the New Scene

Now, insert this scene by selecting **Scene→Insert (before)**. Stepping through the list of scenes will now show that this scene has been inserted before the ”viewPocket” scene, however we need to reprogram our movie to include the new scene.

Reprogram the Movie

Please reset your movie with **Movie→Reset** and then reprogram your movie with **Movie→Program→Scene Loop→Nutate→2 Seconds Each**. Please play your movie. PyMOL has inserted the new scene and displays it in order. Please stop your movie.

Save the Movie

As with any PyMOL movie you may now save it with **File→Save Movie As→MPEG** or save the current session with **File→Save Session As.**

Custom Movie Camera Looping (Optional)

We used the **Movie→Program→Scene Loop→Nutate→2 Seconds Each** in this movie. This nutates for 2 seconds at each scene. If we wanted to say, nutate at scene 1, and then x-rock at scene 2, and y-roll at scene 3, we would have to program that by hand; here’s how.

Scene 1: Program First Scene Plus Transition Time

To program that by hand first, define the three scenes. Click on the first scene’s scene button. Then, select **Movie→Program→Camera Loop→Nutate→15 deg over 4 Sec**. Then add 2 seconds to transition to scene 2 by selecting **Movie→Append→2 Seconds**. If we didn’t add this time, the change from scene 1 to scene 2 would be instantaneous, but now it’s two seconds.

Scene 2: Program Second Scene Plus Transition Time

Now, fast forward to the last frame by clicking on the fast forward button. Please save your second scene to this frame by selecting **M→Store with scene→002** (If you have named the scenes, their names will be available in the Store with Scene submenu) for the **all** object line. If the **M→** menu button is missing, please right-click on the mouse mode table and choose “3-Button Motions.” Then select **Movie→Program→Camera Loop→X-Rock→30 Deg. over 4 Sec**. Add 2 more seconds to transition to scene 3 by selecting **Movie→Append→2 Seconds**.

Scene 3: Store Final Scene Plus Transition Time

Fast forward to the last frame. Save the third scene to the last frame by right-clicking on the camera’s timeline at the last frame, and selecting **Store with scene** and choose your third scene. Lastly, program the y-roll at scene three by selecting **Movie→Program→Camera Loop→Y-Roll→30 Deg. over 4 Sec**. Add 2 more seconds to transition to back to scene 1 in a smooth loop by selecting **Movie→Append→2 Seconds**.

Moving Objects Around in Movies

Until now we have focused on moving the camera around the visual field and changing the representation of objects. So far, the objects themselves never moved—just the camera. Now we consider objects moving independently of each other. We also show you more ways to interact with the timeline and control movies. The movie in this section animates the dissociation of a dimeric complex to gain a better view of the dimer interface.

Creating the Dimer Dissociation Movie

Please start by finding the PyMOL script named "dimer_script.pml" in the **PyMOLTutorials/Moviemaking/motions** directory. Double click on that script to launch it in PyMOL. Your screen should look like Figure 5, except with a black background. This script loads the protein 3f9f, extracts chain A and chain B into two separate objects, and colors them, and creates the two selections (AnearB) and (BnearA), which include the residues in each chain that are within 5 Ångstroms of the other chain. The script also creates six seconds of blank movie frames, represented in a camera timeline and a total frame count of 180 (30 frames per second, by default).

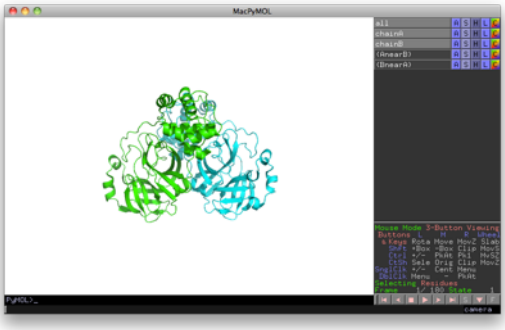

Figure 5: Screenshot of PyMOL after loading dimer_script.pml.

Step 1A—Simple Object Motions

Here, we demonstrate how to drag an object to a new location and have PyMOL stitch together a smooth path from the object's starting position to its ending position. Dragging an object changes its coordinates with respect to other objects, whereas nutating, rocking, rolling and zooming between scenes are computed more simply as camera motions.

Storing chainA's Initial Object Position

Please enable 3 Button Motions mode, by right-clicking on the mouse mode table and selecting 3-Button Motions. PyMOL can now be used to drag objects. Please make sure your movie is on frame 1. Next, please click **M→Store** for chainA. This stores the object's initial position into frame 1.

Dragging chainA to a New Location

Now, position the slider to frame 90. We will drag chain A away from its current position using the **middle** mouse button in motions mode. To do this, please click **M→drag** for chainA to enable "dragging mode" for the chainA object. In this mode, whenever we change chainA's position or orientation, PyMOL will automatically save that change and perform any necessary reinterpolations. Hold down the `SHIFT` button on your keyboard and drag with the **middle** mouse button to move the green chain, chainA, to the left. Dragging changes the object's coordinates in a "motions matrix," saved to frame 90. That is, PyMOL has learned where your object will be positioned in frame 90. It has calculated and interpolated one seamless motion from the position in frame 1 to the position in frame 90. Please press play to see the object motion. ChainA should be sliding in and out as if it's traveling along a line. Please stop your movie.

You may adjust the position of the object at frame 90 to some new location. Please left-click on the movie slider and drag it to frame 90. You are still in "dragging mode" for chainA so you can move chainA to a new location by holding down the `SHIFT` button on your keyboard and dragging with the **middle** mouse button. Press the rewind button. Press play. The object moves to and from the new location. PyMOL knows where the protein should be at each key frame and it smoothly interpolates the transitions.

Multiple Positions for chainA

You may also insert new locations at other frames. Please go to frame 45 by typing `frame 45` or by dragging the movie slider to frame 45. Please drag the chain to a new position and store it. Then, press play. PyMOL ensures that your protein visits each of the locations in each key frame that you specify. If you are satisfied with the object motions, please press Done.

Clearing Motions from the Timeline

Let's clear the motion at frame 45. Please move the slider to frame 45. You may now either right-click on the timeline for chainA and choose **clear,** or you can select **M→clear** for chainA. Please play the movie. PyMOL has cleared the position of chainA that was inserted in frame 45 by dragging. Another way to clear a motion is to delete the frames in which it occurs, as shown in Figure 4C. Please press and hold the `CTRL` key and the left mouse button and drag the mouse to the left to erase frame 90's motion. Now, press play. There should be no motions. Please stop your movie.

Step 1B—Object Motions with Rotations

Please go to frame 90 and choose **M→Drag** for chainA to enable "dragging mode" for the chainA object. You might notice two timelines. The top is for the camera motions and scenes, the bottom is for chainA's motions. Now, please rotate the molecule by dragging with the **left** mouse button while holding down the `SHIFT` key. Your protein rotates. Press the play button. Your molecule rotates in place. Please stop the movie and clear that rotation motion, by right-clicking on the timeline at key frame 90 and selecting Clear. At this point you should have no rotations stored.

We can combine rotations with movement. So, please return to frame 90. First, rotate chainA (left mouse button). Second, move it to a new location (middle mouse button), and press play. You will notice that PyMOL now uses a curved path to combine the rotation and translation of your protein. When rotations are involved, the protein no longer follows a straight path, it moves along an arc. Different rotations define different arcs.

Viewing the Binding Residues

Please remove motions you have for chainA by right-clicking on key frame 90 and selecting clear. Add frames to the timeline if you have less than 100 frames (`CTRL`-left-mouse-button-drag the mouse to the right, over the timeline). Please go to frame 50 and click **M→Drag** again for chainA. Enable the (AnearB) selection to see the indicators. Now, rotate it in place, by pressing the `SHIFT` key and dragging the mouse with the left mouse button depressed, so that the interface residues, (AnearB) are facing the camera. Next, drag chainA away from chainB. We are done moving chainA and will next add visualizations to the movie. So, press Done and then press Play. Now the movie should show chainA moving away from chainB and rotating to display its interface residues.

Step 2—Add the Scene Visualizations

Now let's keep the interface residues exposed for 4 seconds to give the viewer a chance to look at them. To have the open state visible for four seconds, we need to copy the key frame for the first open state to a frame four seconds down the timeline. Remember, PyMOL transitions between key frames. So, if we have two identical key frames separated by four seconds, there is no visual change because the two key frames encode the same information.

Copying a Key Frame to Make a Pause in the Motions

Recall that the interface residues are exposed at frame 50. So, at 30 frames a second (the default frame rate), adding 4 seconds to frame 50 puts us at frame 170 = (50 + 4*(30)). We should add another two seconds (60 frames) or so after that for a smooth final transition. So please inject frames into your timeline if you have fewer than 230 frames (170+60=230). Next, please move the movie slider to frame 50, or just type `frame 50`. The **M** lights up for chainA indicating stored motions for chainA at frame 50. Now, press `SHIFT` and drag with the right mouse button frame 50's key frame identifier near to frame 170. `SHIFT`-right-click-dragging a key frame copies it. right-click-drag of a key frame moves it. If you copy and pasted the key frame too far, then use right-click-drag to move it (instead of `SHIFT`-right-click-drag which would make another copy). If you move or copy a key frame on the timeline, PyMOL automatically stores the information for that frame and performs any necessary reinterpolations so you do not have to click **M→Store** repeatedly. Now, play your movie. ChainA should expose its interface residues for four seconds and then close. Figure 6 illustrates the idea of duplicating key frames on the timeline to pause a view.

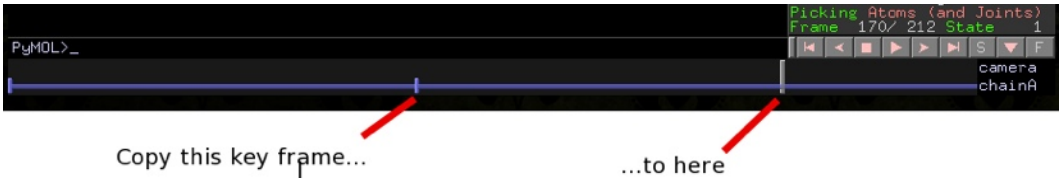

Figure 6: Copying a key frame to a new location to create a visual pause. If we copy the first key frame to the second location, then when the movie plays through that part of the timeline the view stays the same because, as they are the same key frame, no interpolations are required between the two frames. Thus, a visual pause is created.

Adding Scene Visualizations

During the four seconds we added, we would like to zoom in on the interface residues and highlight their structure with a surface and change in color. This is a change in representation, not orientation or position, so we need scenes again. Please store the current visualization as scene 1, using **Scene→Append**. Now, with the slider at frame 1, please assign scene 001 to frame 1 with: **M→Store with Scene→001** for the **all** object line. Please go to your last frame and also choose **M→Store with Scene→001**. This ensures our starting and ending scenes will be the same, for a smooth visual transition. Now, please also store scene 001 at frame 50, after which our next scene will start.

Now, we would like to setup our next scene, one that highlights the interface residues. Please turn on the AnearB selection and for it choose: **A→Action→Orient**. Next show AnearB as surface and sticks. Color AnearB marine. Set the surface color to grey80 with: `set surface_color, grey80, chainA`. Lastly, make the surface somewhat transparent by selecting **Setting→Transparency→Surface→40%**. Your screen should resemble

Figure 7. Please disable the AnearB selection and then append this scene, again by choosing **Scene→Append** from the Upper Control Window. At some frame half-way through the first and second key frames, please store the camera with scene 002 by right selecting **M→Motions→Store with Scene→002** for the all object line. Please repeat this for your last key frame for chainA. That is, advance to the frame for chainA's last key frame. Then store the *camera* motions with scene 002. The **M→Motions** menu for the **all** object line is the same as right clicking on the *camera* timeline. You can store motions and scenes from each, for the camera.

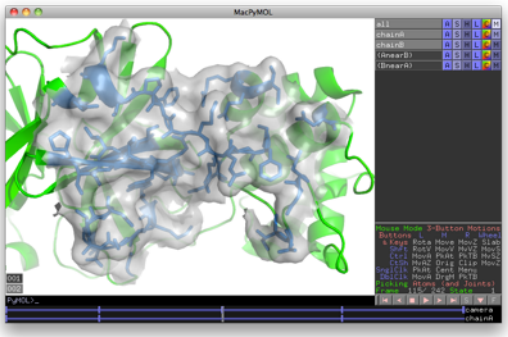

Figure 7: Scene 002 setup for object motions movie.

At this point, if all went well, you should have a movie that show the separation of two chains with one chain's interface residues exposed toward the camera. Then, the camera zooms in on these residues while surface, sticks and colors representations change. There is a pause. Finally, the camera zooms back to the original scene while the chain returns to its original location.

Animating a Trajectory

Molecular Dynamics (MD) trajectories show the motions of molecules in a solvent with a time-dependent force field applied. MD trajectories contain a very large amount of information. Due to the discrete time steps and high-frequency vibrations, most MD movies are "jumpy," making the information hard to interpret. PyMOL comes with tools that allow us to smooth trajectories, stop the molecule from drifting, and even watch bond formations.

Reviewing States in PyMOL

A "state" of an object in PyMOL is a set of spatial coordinates for that object. In an MD trajectory of a protein changing conformations, a "state" is one specific conformation of the protein (and solvent or other molecules, if included), at a given time step, say time step #4. Contrast a state with a scene: scenes in PyMOL do not save molecular coordinates, just the representation, colors and camera position. In PyMOL, you can load a trajectory of states and play the state-sequence, or trajectory, as a movie.

Making a Movie from an MD Trajectory

Please load the file **SampleTrajectory.gz** in the **PyMOL-Tutorials/Moviemaking/trajectories** directory, by selecting the file and dropping it on the PyMOL icon. PyMOL can load pdbs and trajectory files directly from the gzipped file. This file is a small subset (501 time steps) of a trajectory and contains just a few peptides from the original protein. Your screen should look similar to Figure 8. Please ensure that you see "**1/501**" to the right of the word "State" in the Viewer Window. Please press the play button. You will see the familiar motions of an MD trajectory as PyMOL loops through each of the 501 states. The high-frequency vibrations of the trajectory make the information in the movie difficult to detect. Can you see where possible bonds might be forming and breaking? Probably not. Let's clean up the movie to allow for easier interpretation.

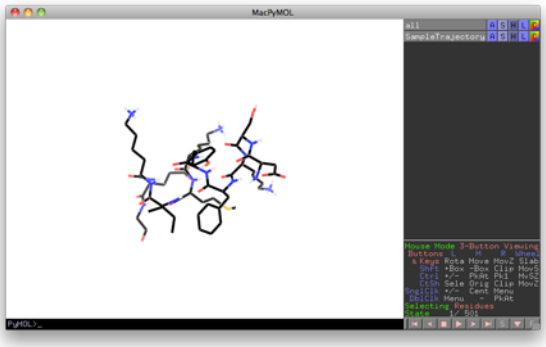

Figure 8: Screenshot of PyMOL directly after loading the sample trajectory file.

Making Visual Improvements to the Trajectory

Let's smooth the trajectory using the `smooth` command. Please press the play button, if you haven't already. Now, please type `smooth` into the PyMOL command line. The motions of the atoms are now smoothed to reduce the high-frequency vibrations. (See the `smooth` command in the PyMOL documentation (http://delsci.info/) for more information on smoothing.)

We can also use PyMOL to stop the peptide from drifting around the screen. We achieve this by telling PyMOL to fit all the states to each other. To do this please type, `intra_fit SampleTrajectory`. The molecule is now centered in the Viewer Window and does not drift off. Please orient the molecule by choosing **A→Action→Orient** for the **SampleTrajectory** object.

Bonding interactions are still difficult to detect. Let's add the hydrogen bonding visualization by selecting **A→Action→Find→Polar Contacts→Within Selection** for the SampleTrajectory object. Now we can see the forming and breaking of hydrogen bonds in the peptide as the trajectory progresses. Figure 9 shows a series of screenshots taken from the movie showing the hydrogen bonds. We improved contrast of hydrogen bonds on the white background by coloring them magenta, with the command: `set dash_color, magenta`.

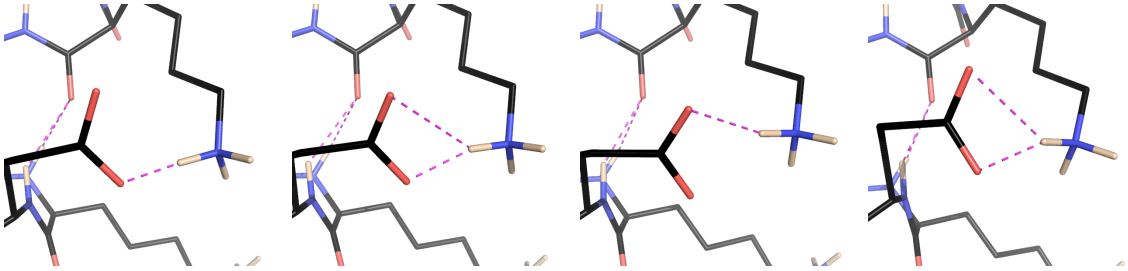

Figure 9: Screenshot of hydrogen bonds changing in the trajectory.

To make the multi-state object into a movie of the states, please use the pre-programmed **state loop** movie option by selecting **Movie→Program→State Loop→Full Speed→No Pause** from the Upper Control Window. PyMOL now shows the timeline, movie slider, and frame count. At this point you can save your movie with **File→Save Movie As→MPEG** to save what is typically shown as an MD trajectory. We can, however, take our movie one step further by combining scenes into the movie.

Let's combine our knowledge of scenes and states to make a movie that focuses on a specific bonding network. Please stop your movie and rewind it. Please turn on Scene Buttons, by selecting **Scene→Buttons**, from the Upper Control Window. Please also enable the 3-Button Motions mouse mode by selecting **Mouse→3 Button Motions**. Now, please save this current view as a scene using **Scene→Append** from the Upper Control Window. We need to store this view in the first frame by selecting **M→Store with scene→001** for the **all** object line. Press the fast-forward button and store that last frame with the scene 001, too, by clicking on the **M→Store with scene→001**, for the **all** object line.

Now, let's find a hydrogen bond to zoom in on. Please drag the movie slider back and forth along the timeline. Locate a hydrogen bond that is forming and breaking between two atoms. Once you've found a bond to target, middle-click on one of its participating atoms to center on it. Next, zoom in very closely on the bond and use clipping to hide other non-relevant bonds. Please aim for a view like that shown in Figure 10. Append this scene using **Scene→Append** from the Upper Control Window. PyMOL stores it as **002**.

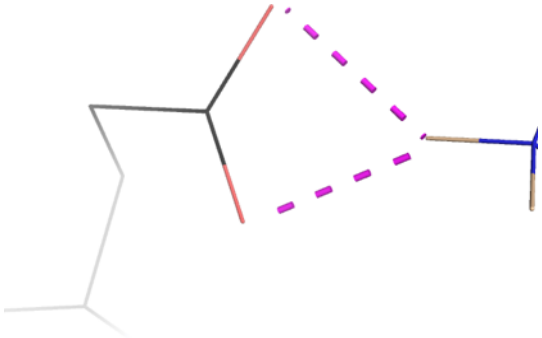

Figure 10: Example bonding network. Note, your colors may be different.

Now we save this scene in the movie in two places. Remember if we save it in only one place, PyMOL will smoothly transition to that scene and then immediately transition away from that scene. So, to stay at this scene for a few seconds, we right click on the timeline where we want to start viewing this scene, say frame 100, and choose

Store with scene→002. Farther down the timeline, maybe about frame 220, please right click on the timeline and again choose **Store with scene→002**. Press the play button. Your movie should now start out in the original view, that of the whole protein. It should then smoothly zoom in on your chosen bonding network. Lastly, it should zoom out again to the original view. You may now save this movie using **File→Save Movie As→MPEG**.

Warnings

Please be warned that moviemaking in PyMOL is a relatively new feature and it can sometimes be buggy. At the moment, you may encounter problems with states and frames in a multi-state movie.

Animating a Molecular Morph

Many molecular systems undergo conformational changes as part of their natural function. In some cases, crystal structures can reveal distinct conformations for similar or even identical molecules under varied conditions. Molecular morphing can help the users to compare how two or more conformations relate to one another by creating a smooth and concerted 3D trajectory between them. It is important to understand that these artificial trajectories, commonly known as "molecular morphs" do not represent, and are not meant to suggest, actual physical pathways of inter-conversion. Rather, they are simply tools for illustration and comparison of complex conformational differences.

Molecular morphing in PyMOL follows the following steps.

1. create or load the first molecular structure
2. create or load the last molecular structure
3. align the first and last structures to each other
4. create the morphing input object from the aligned start and final states
5. call the morphing routine to create an output trajectory object

Morphing Between Two Conformations

Please quit PyMOL. In a plain-text editor, such as Microsoft Notepad, please open the **mtase_morph.pml** script in the **PyMOLTutorials/Moviemaking/morphs** folder. A copy of **mtase_morph.pml** appears in Code 1. Here we walk you through the code, line by line.

Code 1: Methyltransferase morphing script.

```
# load start and final conformations
load ../pdbs/2zlb.pdb, startConf
load ../pdbs/2zvj.pdb, finalConf

# remove solvent atoms, they don't map 1-1 across structures
remove solvent

# align the two structures
align startConf, finalConf, object=aln

# make 2-state morph_in from state 1 of startConf and state 1 of finalConf
create morph_in, startConf in finalConf, 1, 1
create morph_in, finalConf in startConf, 1, 2

# import the rigimol module
from epymol import rigimol

# compute the morph and store the resulting
# trajectory in a new object called "morph_out"
rigimol.morph( "morph_in", "morph_out", refinement=2, async=1 )
```

Line 2 loads our starting object, and names it “startConf” instead of its default pdb file name “2zlb”. Line 3 loads the pdb file representing the final conformation, and names it “finalConf”. Line 5 removes the solvent atoms. While the structures of the two proteins have the same number of atoms, the waters in the pdb files don’t have a 1-1 correspondence so we remove them. Line 7 aligns the two proteins and then stores the alignment in the *alignment object* called “aln.” Line 9 creates the “morph_in” object by inserting state 1 of the startConf object into state 1 of the morph_in object. Line 10 adds finalConf’s 1st state into morph_in’s second state. So, morph_in is now a two-state object where state 1 is our start state and state 2 is our final state. Line 12 is a Python command that imports the module rigimol from the package epymol. Inside rigimol is the morph function, which we call in line 15. We pass the input structure, the output structure, the number of refinement steps, and a flag controlling responsiveness to the morph command.

Please find the icon for this script and drop it on the PyMOL icon. PyMOL launches and runs the script. When PyMOL prints the message “Morph: Refinement complete.” in the Upper Control Window, the morphing calculation is complete. When you see this message, please save your morph to a multi-state pdb file by typing, `save morph_out.pdb, morph_out, 0`. Now, again, quit PyMOL.

Making a Movie with the Morph Output

Please open the **mtase_movie.pml** script from the **PyMOLTutorials/Moviemaking/morphs** directory into your simple text editor. The script is shown in Code 2. The goal of this movie is to visualize the morph. As the morph approaches the final conformation, to which a ligand is bound, we show the ligand in its binding pocket. Here we walk you through the script that creates this movie.

Line 3 loads our multi-state pdb file into PyMOL. This file was created when we ran the mtase_morph.pml script. Line 5 loads the original holo-structure. To remind ourselves that the holo-structure is being used temporarily just for its ligands and waters, we assign it the name “tmp.”

Next, on line 9, we copy the ligands from the tmp object to a new object that we call ligands. We copy from tmp’s first state and insert the object into ligands’ 30th state. We do this because we want the ligands and nearby waters to turn on when our movie reaches state 30. Line 11 is very much like line 9. However, in line 11 we pick up solvent atoms near the binding pocket, within 5 Ångstroms of the ligands. We are done with the tmp object, so we remove it in line 13.

Next, line 17 uses the `mset` command to program a movie. That command takes two types of arguments: a state-hold argument, which comes first, “1 x30” and a state-sweep argument, “1 -30” which follows. The state-hold argument, “1 x30” translates to “hold state 1 for 30 frames.” The state-sweep argument, “1 -30” translates to, “sweep through states 1 to 30 at one frame per state.” Line 17 continues with another state-hold argument, “30 x30” and another state sweep argument, “30 -1”. Our movie is now programmed to view state 1 for 30 frames. Then, sweep from state 1 to 30 over 30 frames. Next, it waits at state 30 for 30 frames before descending back down to state 1, taking 30 frames. Table 1 illustrates the usage of the `mset` command.

Command	Comment
`mset 1-30`	*Invalid*, no space after the 1.
`mset 1 -30`	Valid, sweep from state 1 to 30.
`mset 1x40`	*Invalid*, no space after the 1.
`mset 1 x40`	Valid, repeat state 1 for 40 frames.
`mset 10 -20 20 x30 20 -1`	Valid: sweep from state 10 to 20, then stay at state 20 for 30 frames, then sweep down to 1.

Table 1. Example usage of the `mset` command.

Please note the special syntax. There is a space before the “x’ and “-” in the state-sweep argument, “1 -30” so, “1-30” will cause an error. “1 -30” is correct.

Lines 19–24 make the ligand the center of our attention for the entire movie. Specifically, line 19 starts the movie. Line 20 resets the view to the original position. Line 21 focuses in on the ligand and makes the center of the view the center of mass of the ligand. Line 22 shows the ligand as sticks. Line 23 zooms on the center just defined and then zooms out to a 12 Ångstroms sphere. Lastly, line 24 turns the scene for a slightly better view of the binding pocket.

Please run the **morph_movie.pml** script by dropping its icon onto the PyMOL icon. Once your movie is made, you can save it with **File→Save Move As→MPEG** from the Upper Control Window.

Troubleshooting

When morphing fails, it will leave output for us to investigate called “rigimol.inp.” The most common cause of a morphing failure is that the two structures are incompatible. In order to successfully morph from a starting to a final conformation, you must have very high, if not perfect, sequence identity between the two structures.

Alternate coordinates will cause RigiMOL and `smooth` to fail. Please remove all atoms with alternate coordinates, leaving only one consistent set of atoms per structure.

Code 2: Making a movie from the methyltransferase morph trajectory

```
# Use a morph and holo structure to make a more biologically interesting movie
# Load the multi-state pdb file
load morph_out.pdb

# load the holo structure
load ../pdbs/2zvj.pdb, tmp

# we want to show the ligands bound to the protein in the final conformation, which is
# now state 30 of the morph_out object. So, we create a new object called ligand from
# the holo structure's ligand and put that into state 30 of the new object.
create ligand, tmp and organic, 1, 30

# do likewise for the waters nearby the binding pocket
create waters, tmp and (solvent within 5 of organic), 1, 30

# remove this object, we only needed the ligands and waters
delete tmp

# Use the mset command to make a movie. This starts in state 1 for 30 frames (1 x30),
# then transitions up to state 30 (1-30). Next, it stops at state 30 for 30 frames (30
# x30). Then, it goes back to state one, taking another 30 frames to do so (30 -1).
mset 1 x30 1 -30 30 x30 30 -1

# play the movie and setup the scene.
mplay
reset
orient ligand
show_as sticks, ligand
zoom center, 12
turn x, 90

# This is the command line version of File->Save Movie As->MPEG.
cmd.movie.produce("morph_movie.mpeg", quiet=0)
```

Morphing Large Structures

PyMOL provides another method for calculating morphs. This method simply calculates a linear combination of coordinates of the specified structures. If we opt to, we can refine the resulting morph using RigiMOL in "refinement" mode. This technique allows us to: morph large structures that RigiMOL alone could not handle; very quickly morph smaller structures; and easily make a morphs from multiple input conformations. The trade off for this added capability is that these morphs are visually less appealing; they're based off simple linear combinations of coordinates not RigiMOL-like geometry-aware calculations.

Molecular morphing, following this technique in PyMOL follows the following steps:

1. create or load the first molecular structure
2. create or load the last molecular structure
3. (optional) align the first and last structures to each other

4. create a multi-state target object from N-copies of the first structure and N-copies of the second structure
5. call smooth to linearly combine the coordinates
6. (optional) refine the morph using RigiMOL in “refinment” mode.

Let’s use this technique to create a morph. Please open the **linear_morph.pml** script from the **PyMOLTutorials/Moviemaking/morphs** directory into a simple text editor, like Microsoft Notepad. The script is shown in Code 3. We will use the same two proteins as the last morph so we can compare the results and differences in speed.

Code 3: Linear Morphing and Refinement

```
# load start and final conformations
load ../pdbs/2zlb.pdb, startConf
load ../pdbs/2zvj.pdb, finalConf

# remove solvent atoms, they don't map 1-1 across structures
remove solvent

# align the two structures
align startConf, finalConf, object=aln

# make 100-state morph_in from state 1 of startConf and state 1 of finalConf
for x in range(51): cmd.create("m_out", "startConf", 1, x)
for x in range(51,101): cmd.create("m_out", "finalConf", 1, x)

# perform the linear smoothing across all states
smooth m_out, 1, 100, 1, 100

# import the rigimol module
from epymol import rigimol

# compute the morph and store the resulting
# trajectory in a new object called "morph_out"
rigimol.refine(2, "m_out")
```

Next Steps

Other Courses

Congratulations on finishing the Intermediate PyMOL tutorial. We hope you enjoyed the lesson and are motivated to use PyMOL for your molecular visualization, movie-making, scripting, and editing tasks. This isn’t the end, though: Schrödinger has other courses available covering Molecular Editing and Cleanup, and Intermediate PyMOL. Please contact us for more information on those courses via help@schrodinger.com.

Joining the PyMOL–Users Mailing List

As of Summer 2011, well over 1,500 PyMOL users subscribe to the pymol–users mailing list, which is where the community exchanges tips on how to use the software effectively and how to solve any problems that come up. To join the mailing list, please click on the **Mailing List** link that appears at the top of the PyMOL Home Page, (`http://www.pymol.org`). Or, to quickly search the mailing list archives for posts on a specific topic, click the **Mailing List Archive** link.

Visiting the PyMOLWiki Community Web Site

The community also runs a "Wiki" dynamic content site that aggregates information from the mailing list and provides other kinds of PyMOL–related documentation (`http://www.pymolwiki.org`). PyMOLWiki users can create their own pymol–related pages, on this site or elaborate upon useful information that is already posted. The site boasts over 1,300 users, over 7,000,000 page views, is typically accessed between 4,000–6,000 times a day, and has many useful scripts, plugins and tutorials free for download.

Accessing the Official Documentation Site

PyMOL subscribers access Schrödinger's Official Documentation site (`http://pymol.org/dsc`) using the subscription credentials shown from their most recent PyMOL receipt. This site contains the latest chapters from an updated PyMOL Manual, an assortment of narrated ScreenCasts, and plenty of reference information to help users on their way becoming PyMOL experts.