PyMOL Scripting

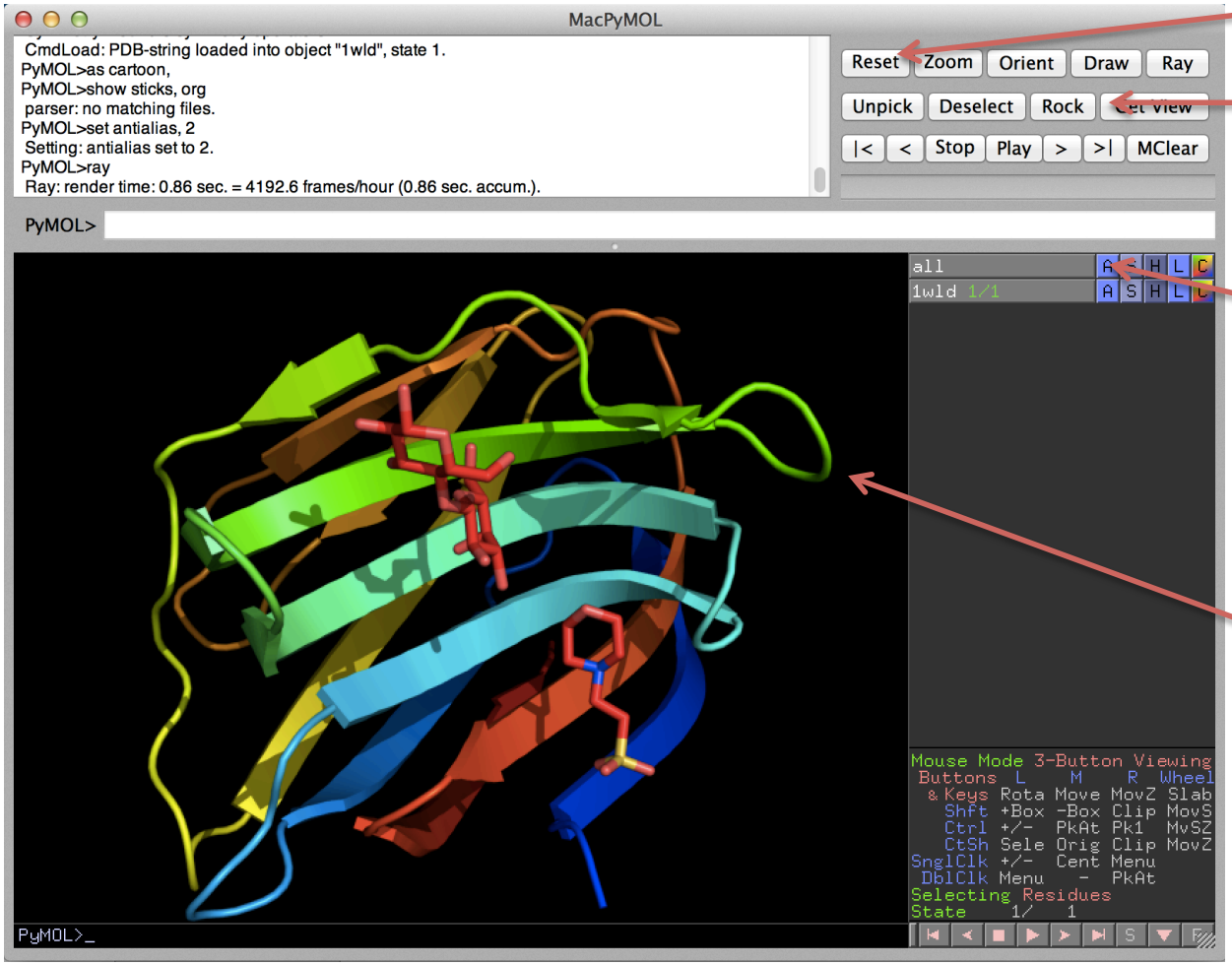
python™

Python & PyMOL

- Arbitrary Python code possible within PyMOL scripts
- PyMOL functionality available by importing PyMOL's modules cmd, cgo, stored etc.
- Great for repeating tasks (but different input)
  - Structural alignments of two proteins
  - Movies of the binding site
- Similar concept in Chimera, Yasara, VMD, …
  - they all have Python APIs

From GUI to Scripting

Every button, setting, etc. in the GUI has an equivalent command!



cmd.reinitialize()

cmd.rock()

cmd.select('all')

cmd.show('cartoon')

Selections & Representations

```
cmd.show(<representation>, <selection>)
```

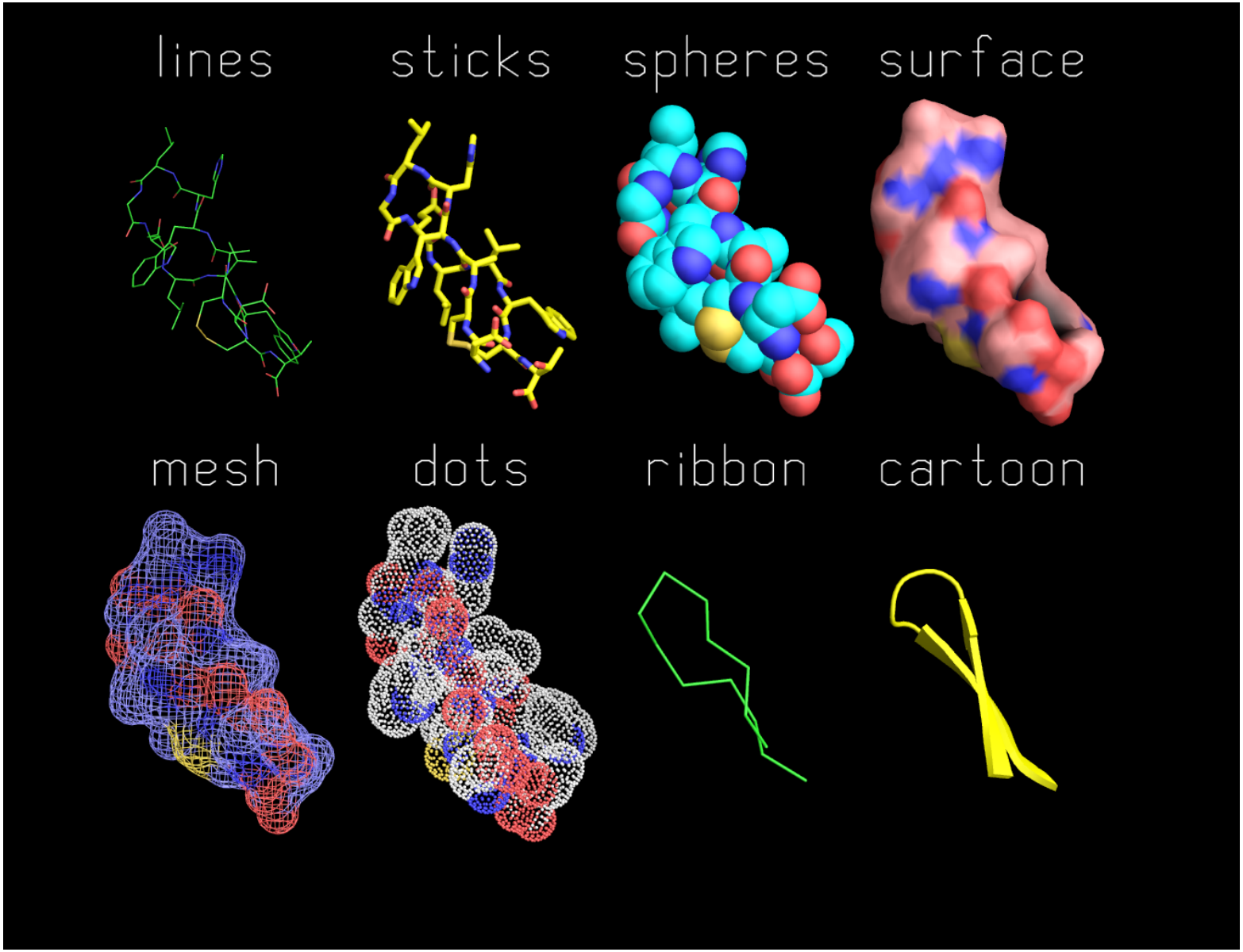
Syntax

```
cmd.show(“sticks”)
cmd.show(“cartoon”, “1vsn”)
cmd.show(“lines”, “2reg and chain A”)
cmd.show(“everything”, “all”)
cmd.show(“sticks”, “!(chain B)”)
```

Examples

Same syntax for cmd.hide()

PyMOL Representations



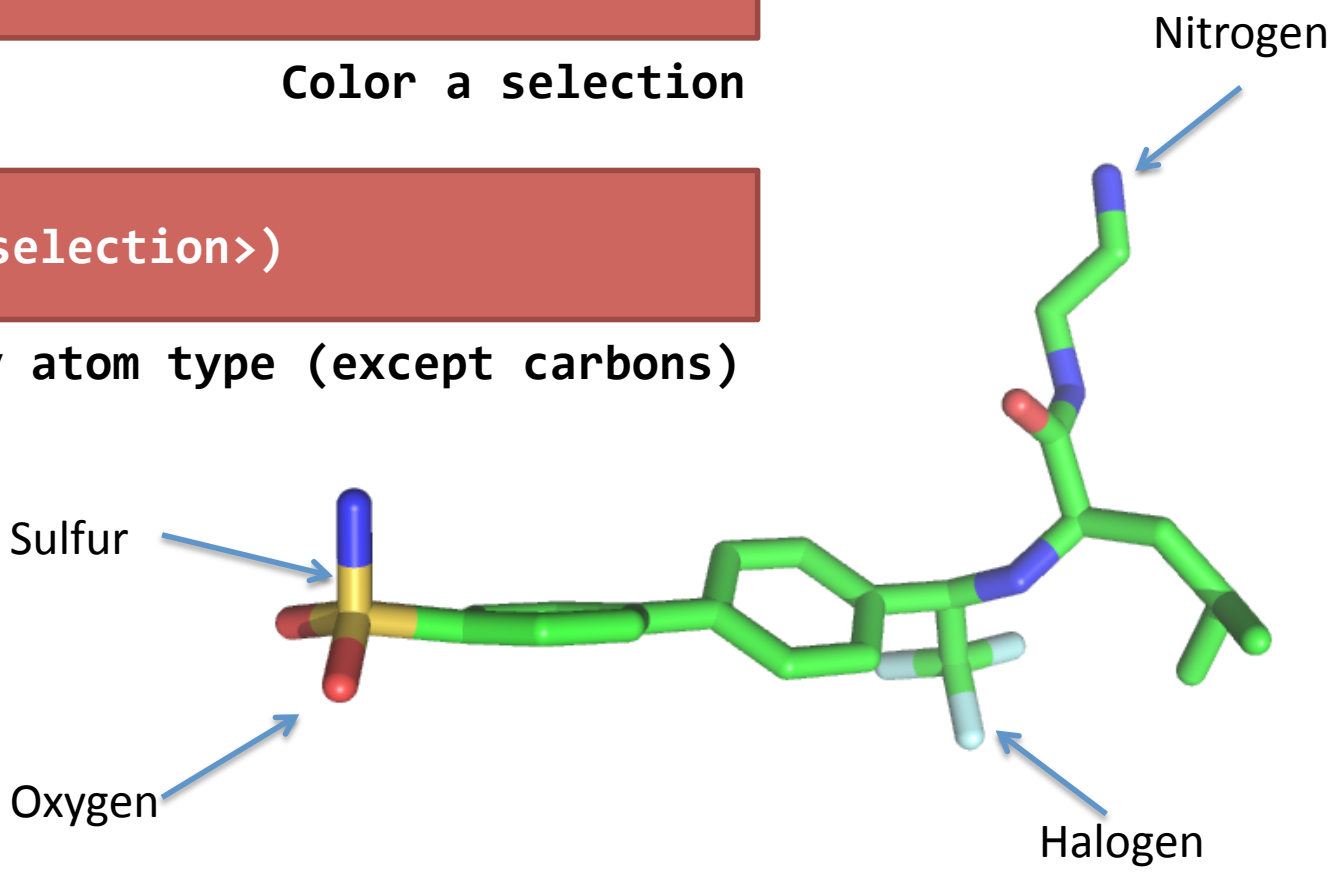
Further Commands

```
cmd.color(<color>, <selection>)
```

Color a selection

```
cmd.util.cnc(<selection>)
```

Color by atom type (except carbons)



Loading and saving data

```
cmd.load(<filename>)
```

Load a (PDB) file from disk

```
cmd.fetch(<pdbid>)
```

Get a structure from the PDB server

```
cmd.png(<filename>)
```

Save the current image as PNG

```
cmd.save(<filename.{pdb,pse}>)
```

Save a PDB or PSE (binary session) file

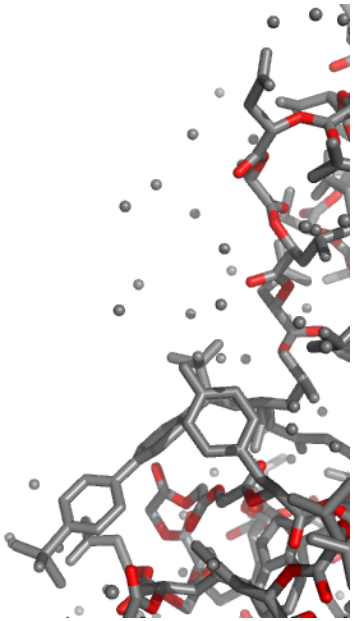
Advanced Selections I

het

Special identifier
For heteroresidues

name ca

Alpha-carbon atoms



not het and name o+n

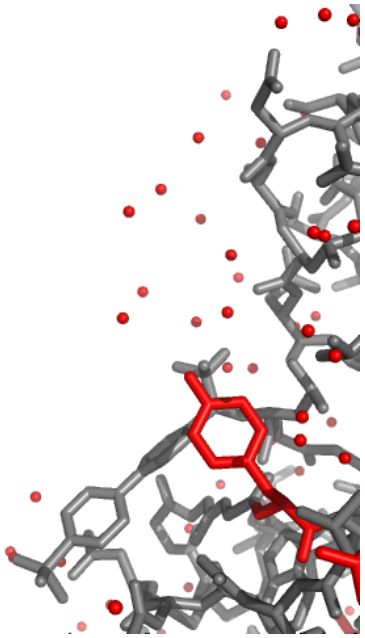
Just protein oxygen
or nitrogen atoms

See http://pymol.sourceforge.net/newman/user/S0220commands.html for a detailed listing

Advanced Selections II

resi 1-100

Residues 1-000
(resI -> ID)



resn TYR or resn HOH

Only tyrosin or water “residues”
(resN -> name)

See http://pymol.sourceforge.net/newman/user/S0220commands.html for a detailed listing

Named Selections

```
cmd.select(<name>, <selection>)
```

Named selection

```
cmd.fetch("1vsn")
cmd.select("near_atoms","1vsn within 4 of resn NFT")
cmd.select("near_aa, "byres near_atoms & !(resn NFT)")
cmd.color("grey")
cmd.color("red", "near_aa")
```

Expands atom selection to complete residues

Expands atom selection with specified radius in Angstrom

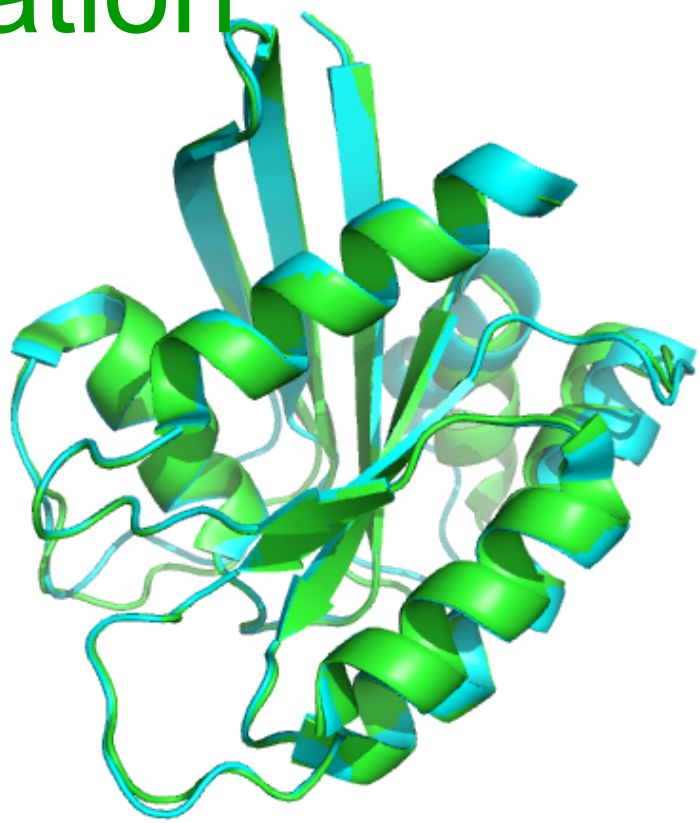
The script colors all binding site residues whose closest distance to the ligand is 4 Angstrom or less

See http://pymol.sourceforge.net/newman/user/S0220commands.html for a detailed listing

Beyond visualization

Scripted alignment

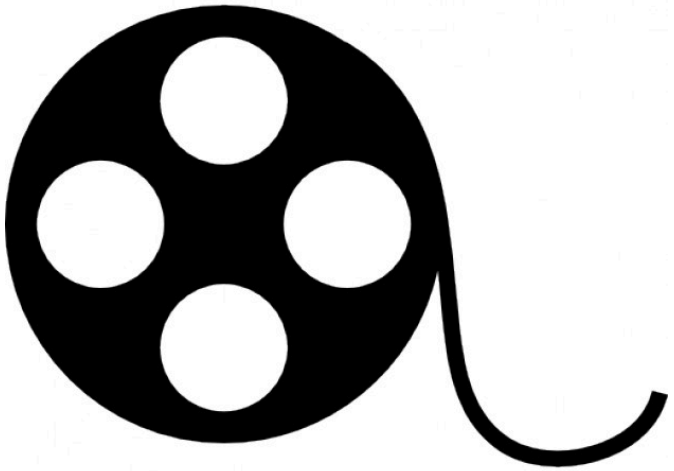
```
import sys

import pymol
from pymol import cmd
pymol.finish_launching()

prot1, prot2 = sys.argv[-2], sys.argv[-1]
cmd.fetch(prot1)
cmd.fetch(prot2)
cmd.hide("everything")
cmd.show("cartoon")
cmd.set("bg_rgb", "white")
aligndata = cmd.super(prot1, prot2)
print "The RMSD is %.3f" % aligndata[0]
```



python align.py 1lf0 alf5

The RMSD is 0.251

Movie School

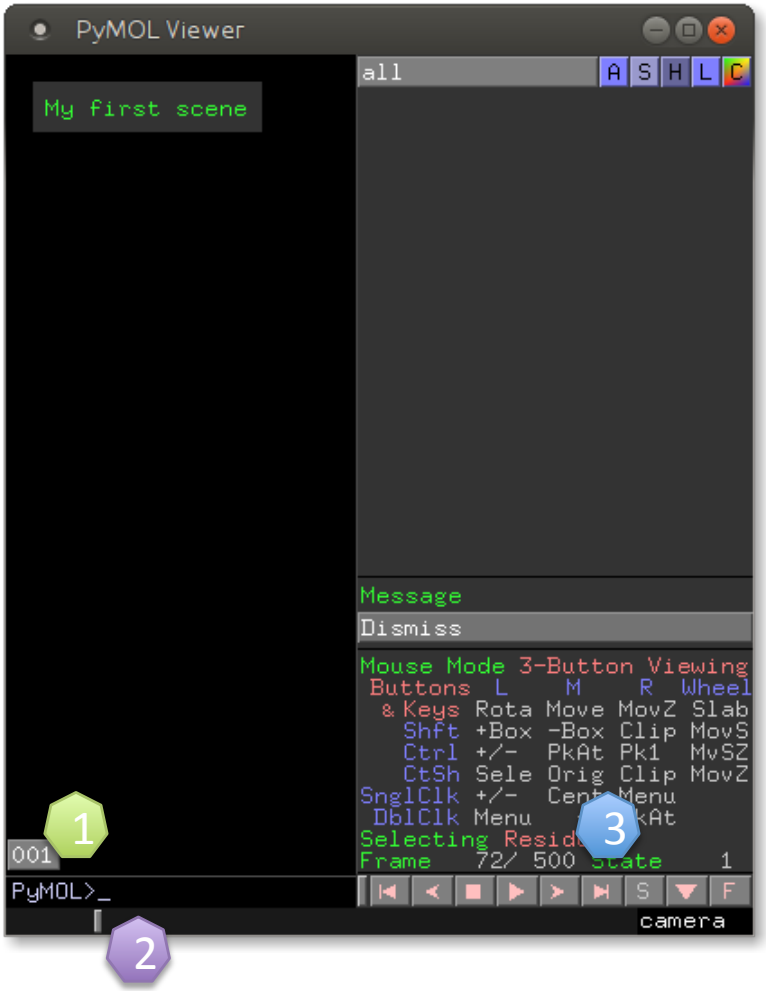


Movie School I: Setup

Import modules and start PyMOL session

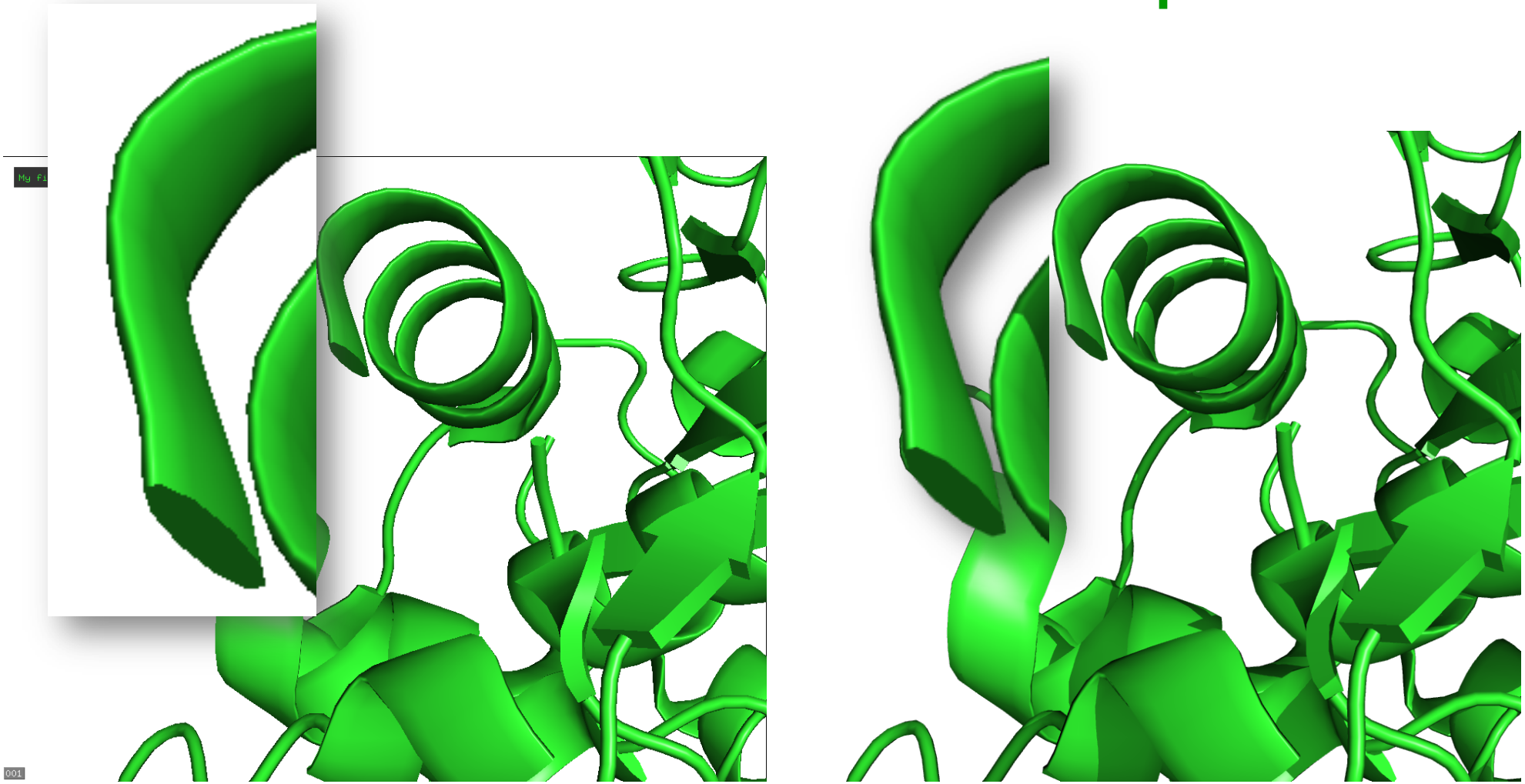
```
import pymol
from pymol import cmd
pymol.finish_launching()

cmd.set('scene_buttons', 1)
cmd.set('matrix_mode', 1)
cmd.set('movie_panel', 1)

cmd.mset("1 x500”)  # Number of frames*
cmd.set('ray_trace_frames', 1)
cmd.viewport(800, 800)  # Resolution (px)
```

* 500 frames are enough for a 20 seconds video with 25 frames/second



Launching this script with Python should invoke PyMOL and show a black screen with everything set up for a movie.

Movie School I: Setup



Standard

Raytracing+Antialiasing

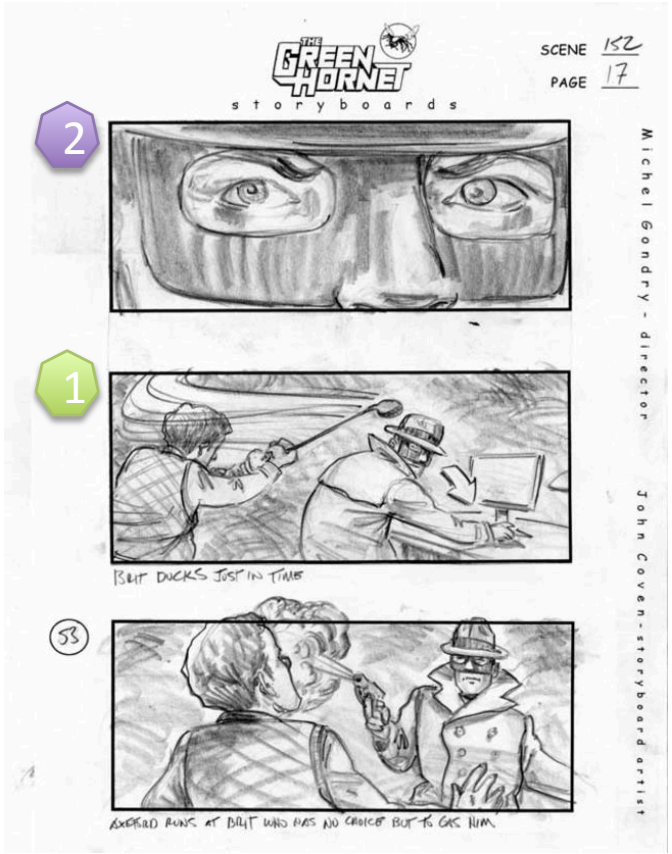
Using `cmd.set('ray_trace_frames', 1)` **or** `cmd.ray()`
will raytrace and antialiase your image,
but takes much longer then real-time rendering

Movie School II: Scenes

Set everything up

```
setup_pymol() # see step I

cmd.load('Cathepsin.pdb')  # Load the PDB file

cmd.hide('everything', 'all')  # Hide everything
cmd.show('cartoon', 'all')  # Protein in cartoon
cmd.select('ligand', 'resn NFT')  # Select ligand
cmd.deselect()  # Deselect everything
cmd.show("sticks", "ligand")

cmd.zoom('Cathepsin', 10) # Overview
cmd.scene('001', 'store', message='scene 1')
cmd.zoom('ligand', 5) # Close-up of ligand
cmd.scene('002', 'store', message='scene 2')
```

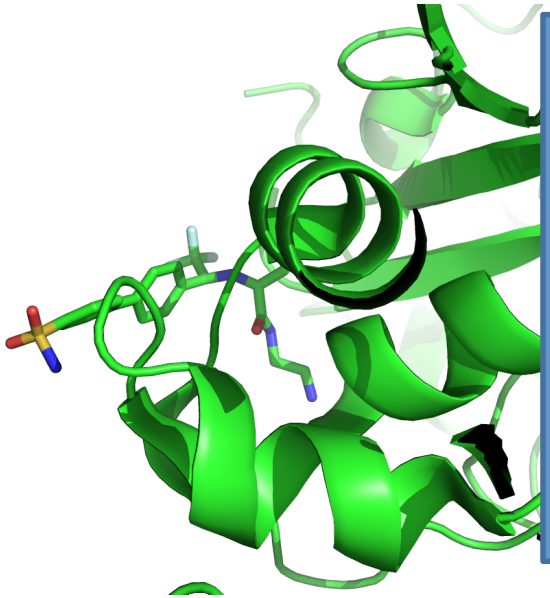
Scene messages are optional parameters



Set up each scene (actors, camera position) like in a storyboard.
Later, we will let PyMOL take care of the transitions.

http://screencrush.com/files/2013/01/movie_storyboards_the_green_hornet_01.jpg

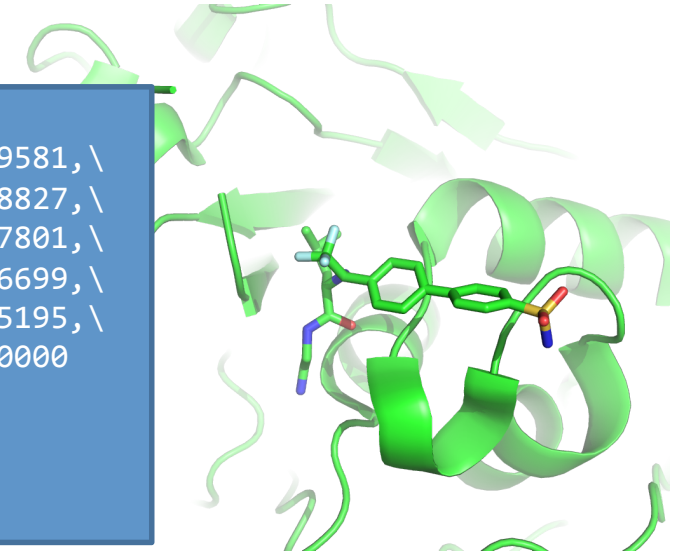
Movie School III: Manual Viewpoints



```
setup_pymol() # see step I
initial_representations() # see step II

cmd.zoom('Cathepsin', 10) # Overview
cmd.scene('001', 'store', message='scene 1')

cmd.zoom('ligand', 5) # Close-up of ligand
cmd.scene('002', 'store', message='scene 2')
```

Viewpoint for scene 2

```
closeup = '''
    -0.775189936,    0.267432511,   -0.572329581,\
     0.387867898,    0.916590214,   -0.097048827,\
     0.498639554,   -0.297219634,   -0.814257801,\
     0.000021780,   -0.000062047,  -62.138366699,\
    -3.786274910,   25.372997284,    6.908325195,\
    45.995002747,   78.286071777,  -20.000000000
'''

cmd.set_view(closeup)
cmd.scene('002', 'store', message='scene 2')
```



Choose manual viewpoints for better views into binding sites.
Get the coordinates and angles with `get_view()`.

Movie School IV: Animation

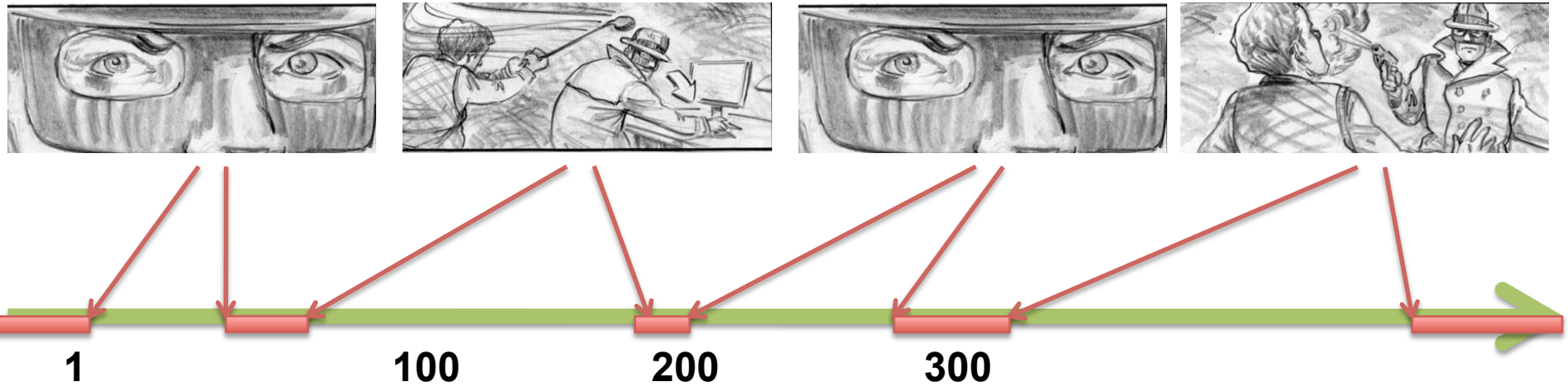
Movie 4

```
setup_pymol() # Step I
initial_representations() # Step II
set_up_scenes() # Step III

cmd.scene('001', animate=0) # Choose scene
cmd.mview('store', 1)  # Assign to frame
cmd.mview('store', 150)

cmd.scene('002', animate=0)
cmd.mview('store', 250)
cmd.mview('store', 400)
```

First, choose an existing scene

Then, assign it to a frame

In our example, scene 001 is shown from frame 1 to 150. Then, we will have a cameraflight of 100 frames to scene 2, which is shown until frame 400. Then, another cameraflight leads again to scene 1 (loop!)



PyMOL takes care of camera flights between scenes by interpolating smoothly between views in scenes.

Movie School V: Animation

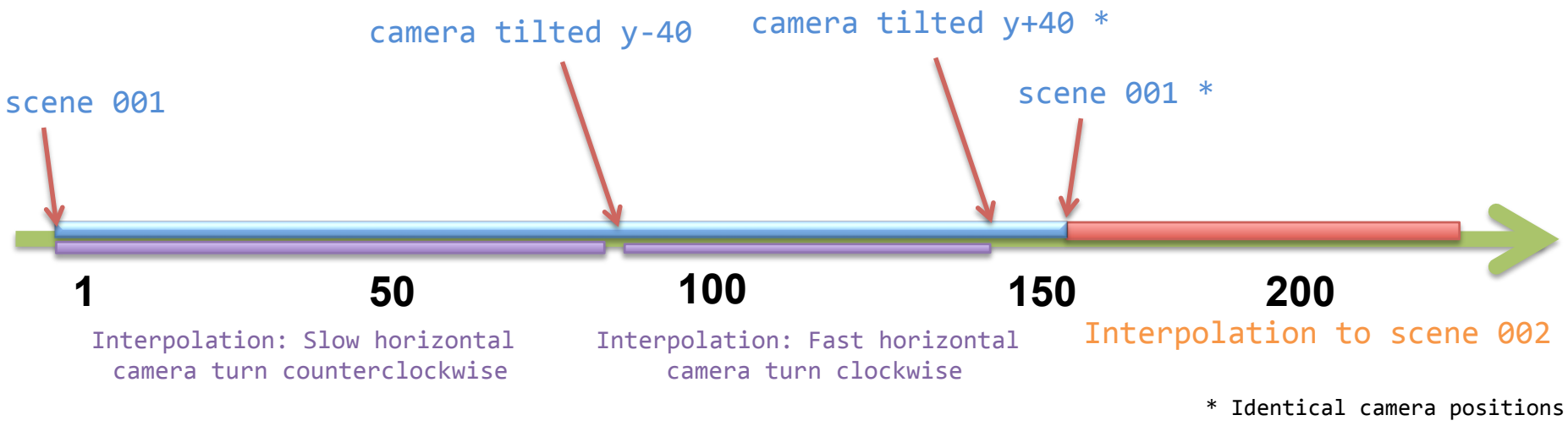
Movie 5

```
setup_pymol() # Step I
initial_representations() # Step II
set_up_scenes() # Step III
scenes_to_frames() # Step IV

cmd.scene('001', animate=0)
cmd.turn('y', -40)
cmd.mview('store', 80)
cmd.turn('y', 40)
cmd.mview('store', 140)

cmd.scene('002', animate=0)
cmd.move('x', 5)
cmd.mview('store', 320)
```

First, choose an existing scene (starting point)

Then, modify the camera

Save the new viewpoint to a different frame



* Identical camera positions

Additional camera movement are interpolated by PyMOL as well.

Movie School VI: Finishing up

Movie 6

```
setup_pymol() # Step I
initial_representations() # Step II

cmd.color('palegreen', 'Cathepsin')
cmd.color('tv_orange', 'ligand')
cmd.util.cnc('ligand') # Atom coloring
cmd.set('bg_rgb', 'white')

set_up_scenes() # Step III
scenes_to_frames() # Step IV
additional_camera() # Step V
```

Nice color contrast for ligand and binding site

White background is best for presentations

Some improvement options

```
cmd.rewind()
cmd.save('/tmp/movie_session.pse')
cmd.set('ray_trace_frames', 1)
cmd.mpng('/tmp/movie')
```

Rewind movie to frame 1

It's always a good idea to save the session

Turn on ray-tracing and antialiasing

Save frames as /tmp/movie0001.png, /tmp/movie0002.png, etc.

Render and save movie frames

Go to the folder with the rendered frames and use

`avconv -f image2 -i movie%04d.png -r 25 movie.mp4`

to generate a movie file.