

Rochester Institute of Technology

RIT Digital Institutional Repository

Theses

7-30-2021

ProMOL: Updating the Template Based Structural Alignment Plugin

Mariah Robertson
mr2893@rit.edu

Follow this and additional works at: <https://repository.rit.edu/theses>

Recommended Citation

Robertson, Mariah, "ProMOL: Updating the Template Based Structural Alignment Plugin" (2021). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

ProMOL: Updating the Template-Based Structural Alignment Plugin

Mariah Robertson

Thesis Submitted in Partial Fulfillment of the Degree Master of Science in Bioinformatics

College of Science

Thomas H. Gosnell School of Life Sciences

Rochester Institute of Technology

Rochester, New York

July 30, 2021



**Rochester Institute of Technology
Thomas H. Gosnell School of Life Sciences
Bioinformatics Program**

To:Head, Thomas H. Gosnell School of Life Sciences

The undersigned state that Mariah Robertson, a candidate for the Master of Science degree in Bioinformatics, has submitted her thesis and has satisfactorily defended it.

This completes the requirements for the Master of Science degree in Bioinformatics at Rochester Institute of Technology.

Thesis committee members:

Name	Date
_____ Paul A. Craig, Ph.D. Thesis Advisor	_____
_____ Gary R. Skuse, Ph.D.	_____
_____ Joe Geigel, Ph.D.	_____
_____	_____
_____	_____
_____	_____

Feng Cui, Ph.D. 475-4115 (voice)
Director of Bioinformatics MS Program fxcsbi@rit.edu

Abstract

In silico methods have contributed greatly to our understanding of the molecular world. PyMOL, an open-source 3D molecular visualization software, produces high quality images and videos of proteins. Here, we discuss a plugin called ProMOL that augments PyMOL's capabilities by adding a way to decipher unknown proteins using template-based structural alignment. Software is constantly updated to meet the demands of users. ProMOL fell behind in this regard and required a major overhaul in its code to work with older and newer versions of PyMOL. This included multiple API updates, Python language migration, and code optimizations. The result of the overhaul produced two versions of ProMOL. ProMOL 5.5 is a working version used with PyMOL 1.8 and ProMOL 6.0 partially works with PyMOL 2.4. Outdated software stymies scientific progress and innovation. The right tools in the right hands can lead to phenomenal discoveries.

Introduction

Purpose

With the advent of the information age, data has become an essential resource in biological databases. However, this data is nigh useless without proper tools to make sense of it all. The hallmark of understanding proteins is their relationship between structure and function. New structures are constantly being discovered and added to repositories [1], but in many cases their functions have yet to be elucidated. The Protein Data Bank (PDB) is one such repository that contains information on the sequences and structures of the peptide chains, coordinates, and distances which are in a text file in the PDB format [2, 3, 4].

Molecular visualization software, such as PyMOL, reads the PDB file to produce a 3-dimensional image of the protein in question. When the baseline toolkit of a software is not quite enough to accomplish a task, plugins are developed to accommodate missing features. Plugin's augment and enhance baseline tools by providing extra functionality that allows us to extrapolate new ideas and theories [5]. ProMOL [6] is a plugin that assists with deciphering unknown protein functions based on the structure by comparing to proteins with known functions [7].

Just as the information age is an ever-changing system, so too are the fundamental aspects that allow us to communicate the information. Both PyMOL and ProMOL are written in the programming language known as Python. The developers of Python overhauled their syntax when they upgraded from Python 2 to Python 3 [8], which prompted PyMOL to update their code in adherence to the new Python. Unfortunately, ProMOL fell behind and had not been updated to the new standards of Python. ProMOL also needed to be updated to communicate with PDB properly due to changes in the PDB's Application Programming Interface (API) [9].

What are APIs?

API stands for Application Programming Interface. An API's purpose is to help developers build a piece of software with ease [10, 11]. It allows two applications to talk to each other by one requesting information from the other then sending the necessary information back. One benefit an API can provide is security while still allowing public access for third party developers [12]. Public access promotes developer's innovation and collaboration to make products the best they can be.

APIs are written in their own dedicated query language such as GraphQL (Graph query language) and REST (Representational state transfer). Query languages focus on requesting and retrieving information from a database. When a user enters their query, a command will be executed to search and extract the data [13]. Figure 1 gives a representation of the differences between REST and GraphQL in how they handle requests and fetch data from the servers.

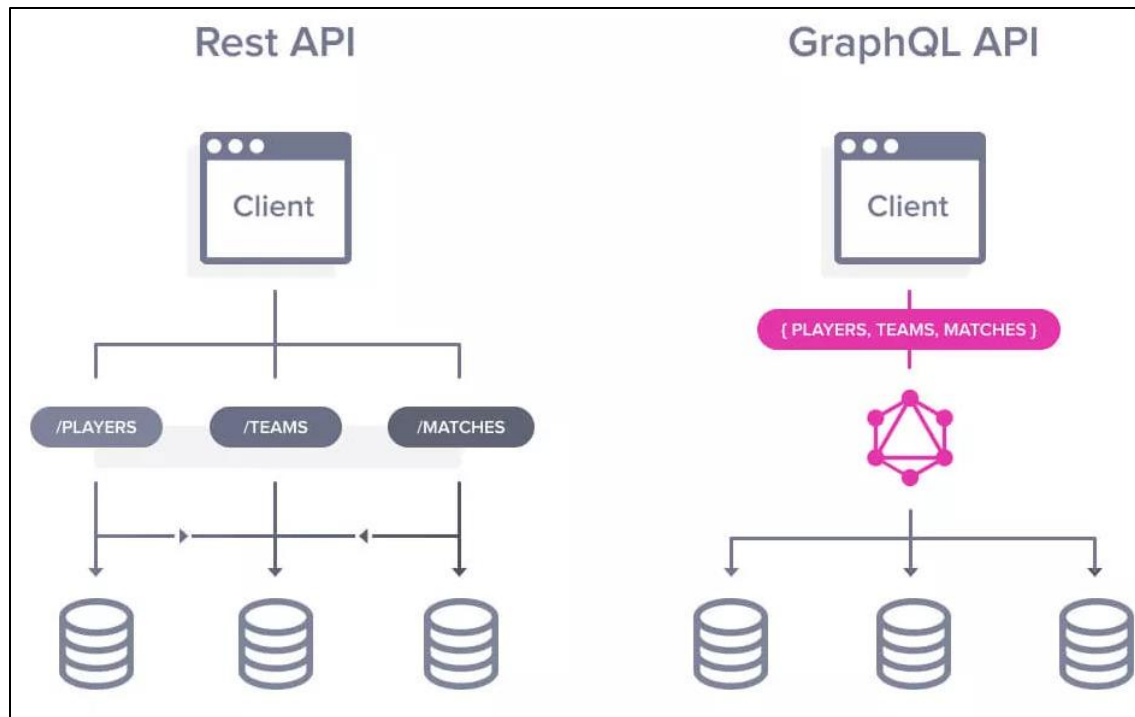


Figure 1: Depiction of how REST and GraphQL fetch and retrieve information. REST makes multiple calls and gets a fixed amount of data returned. GraphQL needs only to make a single call to get specific information. Retrieved from <https://devopedia.org/graphql>

The Research Collaboration for Structural Biology, which hosts the Protein Data Bank (PDB), has started replacing REST with GraphQL due to the way it handles requests [9]. GraphQL is faster and more efficient by only retrieving the necessary information the user wants whereas REST would also retrieve extra unneeded information. Even if the retrieval of information is changed ever so slightly, the benefits add up drastically when hundreds of thousands of requests are being made to the server. Since the PDB has updated the way it communicates with other software, the software also needs to be updated with the correct syntax. Otherwise, trying to fetch and retrieve information would always result in an error or nothing happening.

Why Python 3?

Just as dictionaries are being rewritten with new words to reflect the times, so too programming languages are rewritten. Python 2 was released in the year 2000 but is no longer supported since January 1, 2020 [14]. Python 3 was initially released in 2008 and continues to be used today in 2021. As a result of Python 2's discontinuation, any program using the old version must be updated to the new Python 3.

Python 3 offers more than Python 2 in the way of ease of use and performance. Text and binary data have been separated more clearly in Python 3 so that there isn't any confusion and less bugs will be introduced into the code [15]. Originally Python had been written to use ASCII text which only could be used by people who used the Romanized alphabet. In Python 3, the switch from ASCII to Unicode text happened, allowing the language to be more universally used around the world. The language itself also changed so that programs could run faster and more efficient.

PyMOL

PyMOL is a tool that is open-source software by Schrödinger and can be downloaded via their website (<https://pymol.org/2/>). It allows the user to visualize molecular structures in 3-dimensional space [16]. It is able to pull molecular data from the PDB which houses information on the sequences and structures of the peptide chains, coordinates, and distances within a PDB text file [2, 4]. Viewing molecular structures can be altered to show them as ribbons, cartoon, ball & stick, wireframe, and much more. If the object being visualized has multiple states with the correct coordinates associated, then a short video can be made of it changing between the states [16]. Figure 2 shows how a molecule is portrayed when its PDB file is loaded into PyMOL.



Figure 2: Structure of the complex of L-Benzylsuccinate with wheat serine carboxypeptidase II at 2.0 Angstroms Resolution shown in the PyMOL viewer. Alpha helices are shown as ribbons while the beta sheets can be seen in the background as sheets with arrows.

Specific regions of a molecule can be singled out in the PyMOL viewer, such as a motif, either by selecting them directly from the viewer or by searching from the amino acid sequence. As open-source software, PyMOL's uses are constrained only by the imagination of developers. Many third-party plugins exist that work with PyMOL to deliver extra information to the user. For example, plugins can be designed to look at specific localized features in proteins, known as motifs, which can help scientists discover protein functions.

Motifs

A motif, also known as a super-secondary structure, consists of a recurring combination of secondary structures – alpha helices or beta sheets – found in different proteins [17]. Structural motif templates [18] can be described at high levels that include large portions of secondary structure, but local structural motif templates, consisting of a collection of amino acids

in an active site or ligand binding domain, can also be described. Motifs can be one of the key components to revealing the function of an unknown protein. To reiterate, structure is vital to unlocking the function of a protein. Because small, localized motifs can act as the site where catalytic reactions occur, it is reasonable to conclude that other proteins with this same or highly similar structure also have similar function. This is one way in which proteins are classified and their functions revealed [19]. This technique is known as template-based structural alignment. These entries allow unknown structures to be clarified by examining residue identities and spatial arrangements then cross referencing with known structures that have specific catalytic functions [19]. Motifs corresponding to enzyme active sites are generally highly conserved regions between related proteins due to common ancestors and convergent evolution [7].

Active Sites

Knowing where and what the active sites are in a protein are integral in determining its function. One catalog of enzyme active sites can be retrieved from the Mechanism and Catalytic Site Atlas (M-CSA) and used for template-based alignment against unknown structures [20]. M-CSA version 1.0 originally contained only 177 hand-annotated entries and 2608 homologous entries in the year 2004 [19, 21]. As of March 2021, these numbers have jumped drastically to 964 and 15487 respectively [22].

Searching for enzymatic function in the M-CSA catalog can be done by PDB ID, UniProt ID or Enzyme Commission (EC) number [21]. The European Bioinformatics Institute (EBI) updates and maintains relevant information regarding EC numbers and any meta information. The EC number is composed of four numbers that classifies an enzyme based on its “class, subclass, sub-subclass, ... [and] a serial within the sub-subclass” [23]. It essentially conveys

what class an enzyme is, what donor group the enzyme acts on, with what type of compound as an acceptor, and then a more specific compound is denoted as a serial identifier [24]. The seven classes are oxidoreductase, transferase, hydrolase, lyase, isomerase, ligase, and translocase. For example, 3.1.21.4 is a hydrolase (EC 3) acting on an ester bond (EC 3.1), an alpha endodeoxyribonuclease producing 5'-phosphomonoesters (EC 3.1.21) and is a type II site-specific deoxyribonuclease (EC 3.1.21.4).

ProMOL

ProMOL is a plugin designed to work with PyMOL to explore proteins of unknown function. A working version of this can be downloaded at <http://www.promol.org/> [6]. ProMOL is designed to assemble motif templates that can then query an unknown structure for enzyme active sites [25]. The main features it provides are the ability to make and save templates of motifs, view the best possible alignment between a motif template and query structure, calculate root-mean square deviation (RMSD) and Levenshtein distance between the template and query, request alignments using subsets of the motifs via a template source and/or the EC number, and lastly describe the nature of a structure with an unknown function using *in silico* methods [6].

One capability of ProMOL, in more depth, includes the Motif Maker whereby the user can create a new motif template by using the PDB ID for a given protein, the Enzyme Commission number, and a list of active site residues obtained from the M-CSA. The ProMOL algorithm starts building the motif and begins by evaluating it against itself. ProMOL will then display a group of residues that adhere to distance constraints [25]. Users can design new motif and then test them against homologs of the protein (looking for true positives and false negatives), then against random PDB entries (looking for true negatives and false positives).

ProMOL can then use the Motif Finder to screen against query structures where the method of the search is stringent upon the relative distances of catalytic site residues as shown in Figure 3.

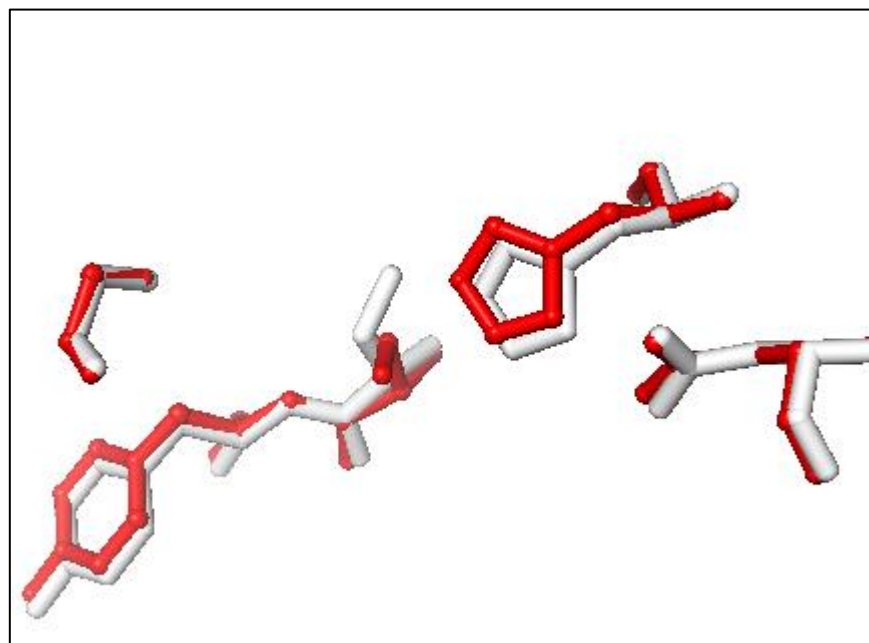


Figure 3: Using ProMOL's Motif Finder capabilities on the protein PDB ID 1wht against the known protein PDB ID 1whs. Five residues were selected using the Motif Maker to search for in 1wht. The white portion shows 1whs and its similarity to the red portion of 1wht. This image is from using PyMOL 1.7 and ProMol 5.4.

Goals

Programs are constantly improving in their functionality, ease of use, and optimization [26]. As such, ProMOL is no longer compatible with the most recent version of PyMOL which is currently PyMOL 2.4. PyMOL 2.4 has had its code migrated from Python 2.7 to Python 3.7 whereas ProMOL is still running on Python 2.7.

The main goal of this project is to go through each file in ProMOL and migrate from Python 2.7 to Python 3.x, so that it is compatible with the newer versions of PyMOL. Any code

that is part of the Legacy API needs to be updated to the new GraphQL PDB API. Careful and well-documented changes in the code will be placed on GitHub. A working version of ProMol with any version of PyMOL is the first hurdle to overcome.

There are a number of additional goals for the project as well. The first is essential because not only has Python been updated, but the PDB has also updated their Application Programming Interface (API) [9]. This has affected the way in which ProMol fetches and searches for PDB files and no longer works. PDB's API has recently discontinued their 'Legacy' API in favor of the new and improved version [27]. The only way to use ProMol is to use older versions of PyMOL which is not convenient for end users as older versions are more difficult to obtain and download. Some additional goals for improving ProMol involve the user interface and user manual. Some portions of the user interface in ProMol are not intuitive and require a bit of finesse to figure out how to work. Text within the interface will be added to clarify the usage of the different options. While the user manual is helpful in installing and using ProMol, it will also be updated to be more user friendly and easier to follow. One of the main reason's users give up on software is the frustration with using it, whether for the challenging installation process, confusing interface, or incompatibility. By creating a more user-friendly experience, ProMol has the potential to help in the discovery of new, unknown proteins.

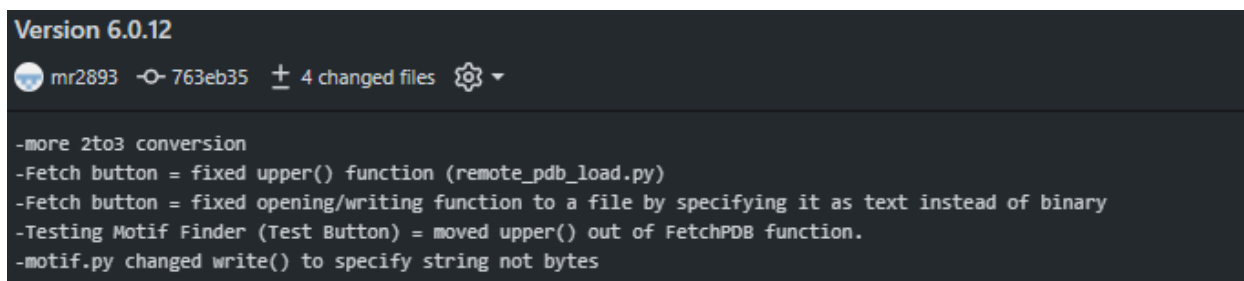
Materials and Methods

Documentation

When doing any coding updates, it is essential to have good documentation of the changes being done in case the newly introduced code breaks the program. A safety net to revert to the most recent working version will save time and increase efficiency. GitHub

(<https://github.com/>) is a web-based platform that provides the tools necessary to keep track of alterations in the code. A separate application called GitHub Desktop works in conjunction with GitHub to ease uploading changes to ProMOL's code.

ProMOL is open to the public and its source code can be downloaded (<https://github.com/mr2893/promol-5.5>). For this project, ProMOL was downloaded onto the desktop and then within the GitHub Desktop, the ProMOL folder was selected to be the initial commit to store the original files on GitHub. A new folder was automatically created inside the ProMOL folder named ".git" to store all future changes. A clear depiction of the line(s) being removed or added to each file can be displayed using the GitHub Desktop interface. The user can make additional notes to new versions being pushed to GitHub as seen in Figure 4.

The image shows a screenshot of a GitHub commit message. At the top, it says "Version 6.0.12". Below that, there is a header with a GitHub logo, the username "mr2893", a commit hash "763eb35", a plus sign, and "4 changed files" with a gear icon. The main body of the commit message contains the following text:

```
-more 2to3 conversion  
-Fetch button = fixed upper() function (remote_pdb_load.py)  
-Fetch button = fixed opening/writing function to a file by specifying it as text instead of binary  
-Testing Motif Finder (Test Button) = moved upper() out of FetchPDB function.  
-motif.py changed write() to specify string not bytes
```

Figure 4: When new versions of ProMOL were pushed to GitHub, a short, required summary of changes was provided along with the version number.

Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is a code editor used to help develop programs by giving visual cues that allows users to know their location in the coding file in a glance. In this project VS Code was used to quickly visualize errors produced by Python's interpreter. Three extensions were installed to help facilitate changes which were called Python, Pylance, and Jupyter. The last two extensions came bundled with Python. One of the useful features of the Python extension is intellisense which provides code completion, parameter info, and quick info via a drop-down box. Some great features include importing an entire folder of content and searching for every

occurrence of specific syntax, replacing all instances of a variable with a new variable name in quick manner, and showing where code blocks begin and end to easily know the scope of the variables and/or functions.

Importing a folder into VS Code is a simple process. Use the following steps to open the drop-down menu: Click on File → Open Folder → Locate and Click on desired folder. All files and folders show up in the explorer tab upon a successful import. Using the magnifying glass on the left side allows the user to search for a word or phrase in any of the files. Results will show up in the explorer tab and can be clicked on to automatically open that file and takes the user to the line with the word or phrase. Figure 5 shows what the layout should look like when the folder has successfully been imported into VS Code.

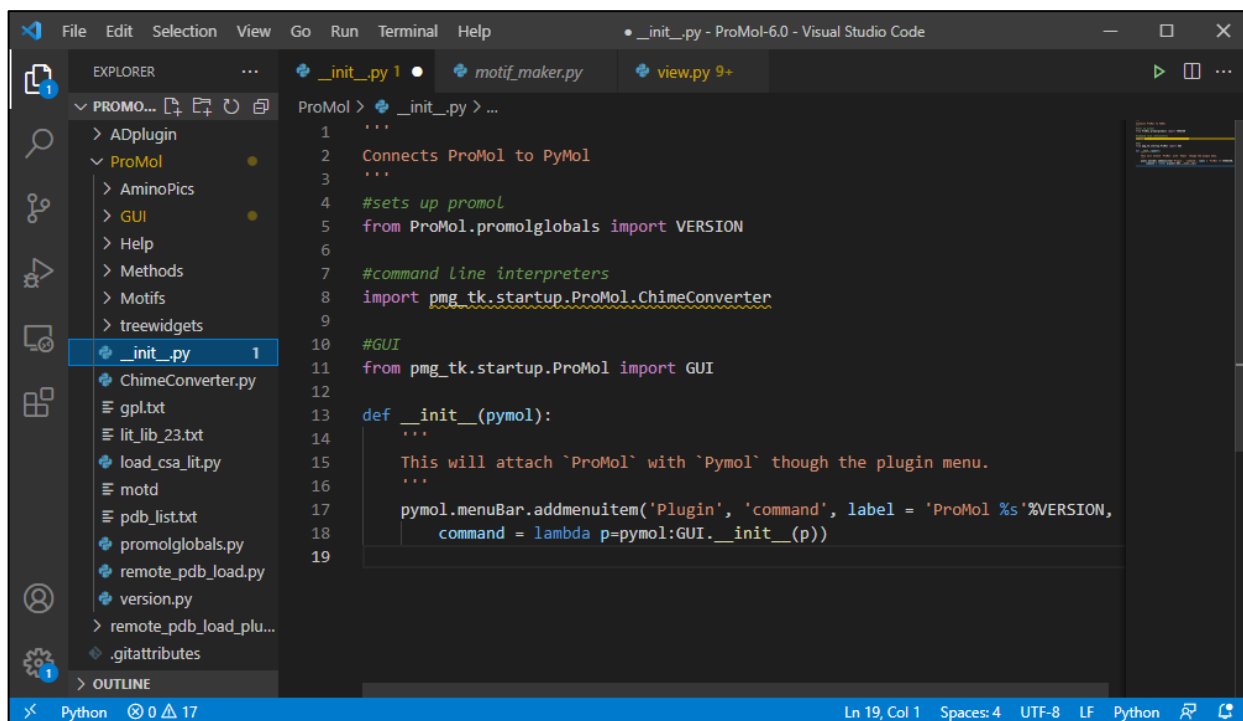


Figure 5: Visual Studio Code window showing ProMOL-6.0 folder being imported on the left side. The magnifying glass in the left column can then be used to search for any word or phrase with the results populating in the Explorer tab.

Setting up PyMOL and ProMOL (Old Versions)

The first step to successfully updating ProMOL required using PyMOL version 1.8 which was written in Python 2. This changed the least number of variables to pinpoint the initial problem of fetching PDB IDs. Installation files of PyMOL 1.8 can be found at sourceforge as a tar.bz2 (<https://sourceforge.net/projects/pymol/files/pymol/1.8/>). ProMOL 5.4 can be downloaded from <https://sourceforge.net/projects/sbevsl/files/ProMOL/ProMOL-5/>. After both have been downloaded and decompressed, it was necessary to move all of ProMOL contents to the directory where PyMOL had been installed, specifically in the directory “PyMOL\modules\pmg_tk\startup\.” Next, the “remote_pdb_load.py” inside the “remote_pdb_load_plugin” folder also had to be copied and pasted into the “PyMOL\modules\pmg_tk\startup” to replace the old remote_pdb_load.py. If ProMOL is properly installed, the Plugin drop down menu should show ProMOL when PyMOL.exe is opened. Clicking on it will bring up the GUI interface as shown in Figure 6 with an error being produced.

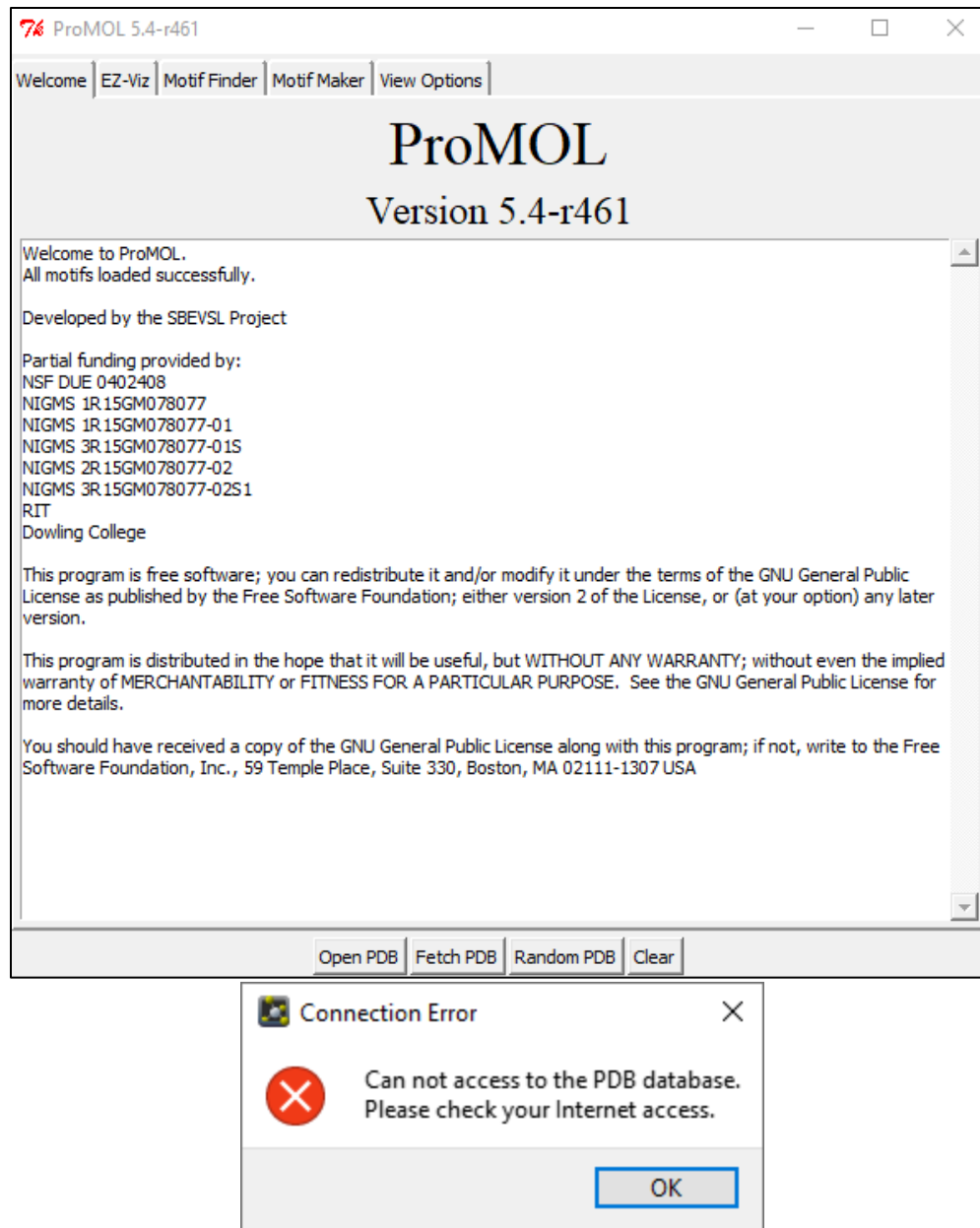


Figure 6: ProMOL's GUI interface. The Fetch PDB button does not work with PDB's API update and produces an error. The incorrect API connects to the wrong database because it no longer exists, regardless of internet connection.

Setting up PyMOL and ProMOL (New Versions)

The most up to date version of PyMOL (2.4) can be found at <https://pymol.org/2/> and can be installed by following the instructions upon downloading it. For this project, the original

version of ProMOL was downloaded to the desktop and as it was being updated, it would become the backup once it was being tested in PyMOL 2.4. Connecting the two requires less effort by the user to install compared to the old versions. In PyMOL, the Plugin drop-down menu has a Plugin Manager option. When it is open, there is an Install New Plugin tab where the user must choose the file location of ProMOL and select the correct python file which is C:\Users\name\Desktop\ProMol-6.0\ProMol__init__.py. Now that the two are connected, updating the code and testing may begin.

Updating PDB API

Twenty-four files with hundreds of lines of code made it difficult to pinpoint exactly where the fetch error was occurring. PDB services had changed the URL for downloading files. A list of where the new files can be downloaded can be found on their website at <https://www.rcsb.org/docs/programmatic-access/file-download-services> under PDB entry file as shown in Figure 7. Once the first instance of the error was located, any instances of the incorrect URL were replaced with the correct one. In this case, 'http://files.rcsb.org/download/' replaced any old or no longer working URLs. This process was expedited by importing the entire ProMOL folder into VS Code and using the search bar to locate any files that contained the incorrect fetch URL. The line number would also be shown to easily see where in the file the error occurred.

PDB entry files

PDB entry files are available in several file formats (PDB, PDBx/mmCIF, XML, BinaryCIF), compressed or uncompressed, and with an option to download a file containing only "header" information (summary data, no coordinates).

File Format	Action	Storage Compression	Example URL
Biological Assembly File in PDB	Download	Uncompressed	https://files.rcsb.org/download/1hh3.pdb1
Biological Assembly File in PDB	Download	Compressed	https://files.rcsb.org/download/1hh3.pdb1.gz

Figure 7: A screen capture from the PDB website of the locations to download different file types. Only the first two types are shown here. The first is an uncompressed file of PDB ID 1hh3 which requires more memory and computational power to download than the second one listed. The second is a GNU zip compression file that is far faster to download than the first. Retrieved from the PDB website.

Updating Python 2 to Python 3

One major change in moving from Python2 to Python 3 was in the naming and calling of many of Python's libraries. Luckily, there is a tool named 2to3 which automates the process. All that is required is setting up Python 3.x from <https://www.python.org/downloads/>. By using the Command Prompt (for Windows users), the working directory must be where the desired python file is located. The user must know where Python was installed to use the 2to3 tool. This can easily be determined by taking the following steps on a Windows operating system as shown in

Figure 8:

1. Press the Windows Key
2. Type in "Command Prompt" and open it.
3. Change to the root directory
4. Search for the file and wait.

```

Command Prompt
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jkell>cd \

C:\>dir "\2to3.py" /s
Volume in drive C is OS
Volume Serial Number is 5291-1210

Directory of C:\Users\jkell\AppData\Local\Programs\Python\Python39\Tools\scripts

05/03/2021  05:34 PM                101 2to3.py
                1 File(s)                101 bytes

```

Figure 8: An example showing how to locate a file anywhere on a Windows computer. “cd \” tells the computer to change to the root directory – the highest folder in the hierarchy. The second line “dir “\2to3.py” /s” will search for all file locations containing 2to3.py. The result shows that 2to3.py exists in Directory of C:\Users\jkell\AppData\Local\Programs\Python\Python39\Tools\scripts

Once the location is known, the 2to3 tool can be utilized by executing the line of code found in Figure 9. The file in question (code.py in this case) was overwritten with the correct Python 3 code and a backup file was also automatically created within the same folder of the original Python 2 syntax.

```

C:\Users\name\Desktop\ProMol-6.0\ProMol>python.exe
C:\Users\name\AppData\Local\Programs\Python\Python39\
Tools\scripts\2to3.py -w code.py

```

Figure 9: An example of how to use the 2to3 Python tool. The first line shows the working directory to run python.exe. The remaining lines tell the directory where 2to3 is located, so that it can be run on the file called code.py.

Anytime a file was changed using 2to3, it was tested with PyMOL / ProMOL and the traceback calls dictated the path to follow. The translation tool did not catch all the lines that

needed to be changed for ProMOL to work. Due to Python's desire to delineate between strings and binary more clearly, many of the functions produced incompatibility errors. Based on the surrounding code, Python 2 would decide what data type (i.e. string or binary) a variable would be. In Python 3, the programmer now must explicitly tell the program what the data type is. In the long run, less bugs would be introduced at a slight cost to convenience to the programmer.

The initial errors dealt with syntax and incorrect library names that could easily be fixed with 2to3. All print statements became a function that can take in arguments / parameters. Many of the changes for this project deal with Python 2's Tkinter package and importing of renamed libraries with changed functions. The Tkinter package from Python 2 was renamed to tkinter in Python 3 and the correct syntax just needed to be implemented for Python 3 compatibility in ProMOL. This mostly involved changing the import statements near the top of python files and calling of different functions throughout the file.

Obtaining a User-Friendly Experience

The original installation process was not intuitive and required some obscure files to be moved to another obscure location in order for ProMOL to communicate properly with PyMOL and load correctly. To streamline the process, folders have been moved around while ensuring all files could still access each other using the right imports. The user only needs to tell PyMOL select a specific file (`__init__.py`) and the installation process is complete.

The View Options tabs presented some unique challenges. It contains many sliders to adjust features in the different views (cartoons, spheres, sticks, surface, ambient light). It is unclear how many of the sliders displayed in Figure 10 interact with the molecule in PyMOL's viewer window. On default settings, the right-hand sliders do not alter anything about how the

molecule is portrayed which may leave the user puzzled. The user must select the EZ-Viz tab to change the surface type of the molecule as seen in Figure 11. The corresponding surface type can be changed via the sliders back in Figure 10. Any place there are unclear interactions between options was solved by placing instructions on how to use it within the ProMOL window, right where the user can see it by the options. A more comprehensive overview of the View Options will be discussed in the results section.

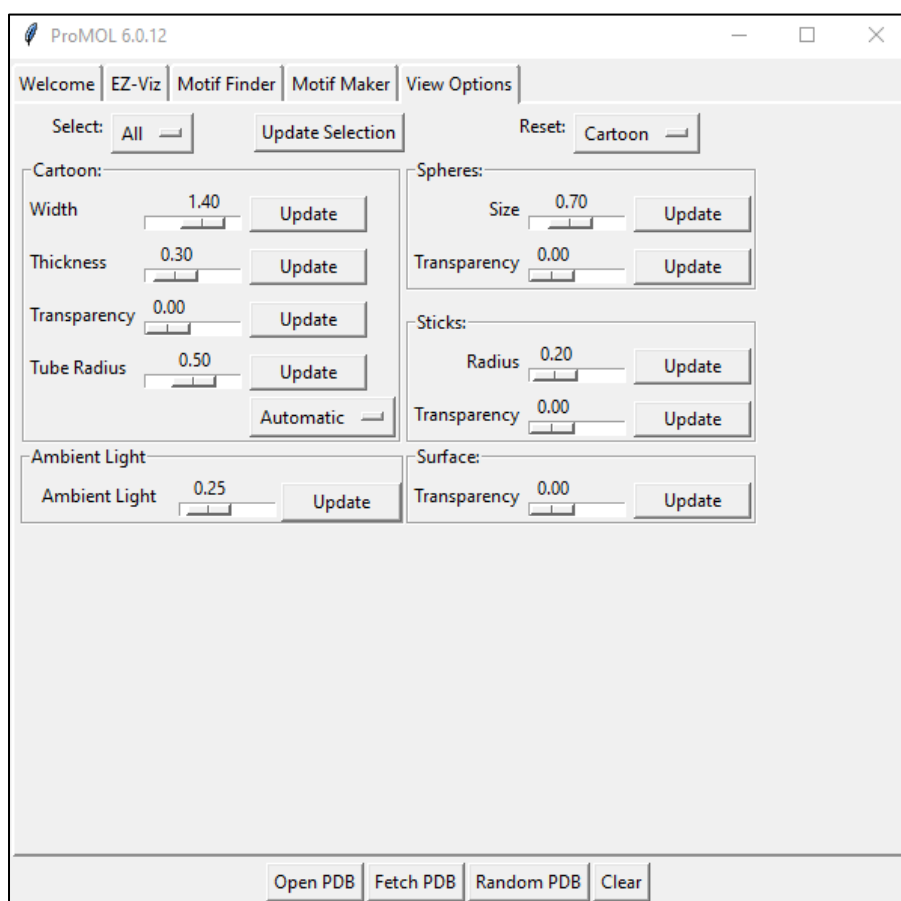


Figure 10: The View Options tab contains options to change how large sphere size and stick radius should be. These options seem to have no effect on the molecule visually without changing other options.

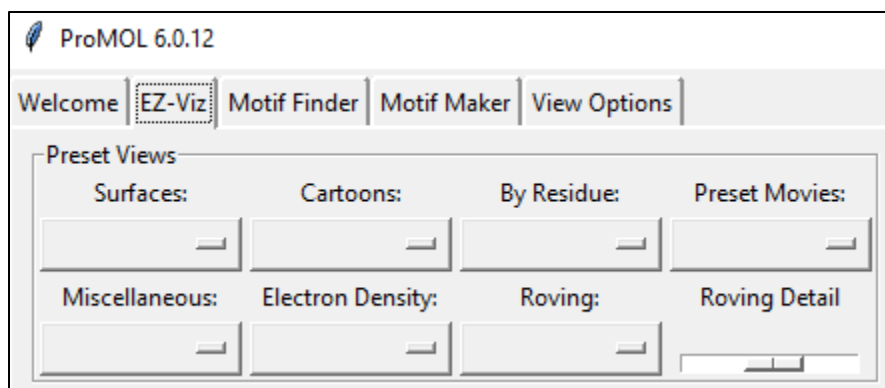


Figure 11: EZ-Viz option tab. The surfaces drop down menu interacts with some of the View Options sliders such as sphere and sticks.

Testing ProMOL

PyMOL's command line shows traceback calls when an error occurs. Troubleshooting the problem requires looking at the last step in the traceback call. The key information to take away from the traceback call are the name of the file where the error is located in, the line number, and syntax/function/statement that describes the error which can be seen in Figure 12.

```

CmdLoad: "" loaded as "lwhs".
ExecutiveAlign: mobile selection must derive from one object only.
CmdException Exception in Tk callback
Function: <bound method TreeNode._onDoubleClick of <pmgTk.startup.ProMol.treewidgets.node.TreeNode object at 0x000001C2F602DA88>> (type: <class 'method'>)
Args: (<ButtonPress event state=Mod1 num=1 x=133 y=118>,)
Event type: ButtonPress (type num: 4)
Traceback (innermost last):
  File "C:\Users\jkell\AppData\Local\Schrodinger\PyMOL2\lib\site-packages\pmw.py", line 1823, in __call__
    return self.func(*args)
  File "C:\Users\jkell\AppData\Local\Schrodinger\PyMOL2\lib\site-packages\pmgTk\startup\ProMol\treewidgets\node.py", line 165, in _onDoubleClick
    self.treewidget.toggleNode(self)
  File "C:\Users\jkell\AppData\Local\Schrodinger\PyMOL2\lib\site-packages\pmgTk\startup\ProMol\treewidgets\widget.py", line 424, in toggleNode
    self.showContentFunc(node)#changed 4/21
  File "C:\Users\jkell\AppData\Local\Schrodinger\PyMOL2\lib\site-packages\pmgTk\startup\ProMol\Methods\motif.py", line 375, in showContent
    data1 = cmd.align(motifSubsetName, querySubsetName)#edited 2/19
  File "C:\Users\jkell\AppData\Local\Schrodinger\PyMOL2\lib\site-packages\pymol\fitting.py", line 395, in align
    if _self._raising(r,_self): raise pymol.CmdException
CmdException: Error:

```

Figure 12: An example of a traceback call when an error occurs. ProMOL is trying to show an alignment between two molecules but is encountering a problem.

Each time a change to the code occurred in ProMOL, another test had to be done to check if the problem was fixed. This involved using the Plugin Manager to uninstall ProMOL and close down PyMOL, reopen PyMOL and install ProMOL from the Plugin Manager again. This process

was repeated until no errors were produced from the command line. Upon installation, the errors could be found right away due to the incompatibility between the Python 2 code in ProMOL and the Python 3 code in PyMOL.

Once each file had been successfully translated from Python 2 to Python 3, each button from the Graphical User Interface (GUI) was tested. PyMOL has its own API and command line syntax, both of which are very similar in structure. PyMOL also changed some of their API and commands that no longer work with the old ProMOL 5.4 version and thus required more diligence in deciphering the problem. It was not always clear if the error occurred in PyMOL's API/commands, PDB's API, or Python's new packages/library system. In order to figure out which of the three possibilities it was, one would have to search all three for any documentation. As an example, PyMOL's command reference (<https://pymol.org/pymol-command-ref.html>) assisted in revealing the changes they made. The command to 'fetch' downloads produced an error. According to the command reference in Figure 14, the 'async' argument now has a default value of 0, meaning it no longer has to be specified. As shown in Figure 13, all that needed to be done was remove 'async=0' and the problem was resolved since no errors were produced upon testing ProMOL.

```
- cmd.fetch(random.choice(database), async=0, path=FETCH_PATH)
+ cmd.fetch(random.choice(database), path=FETCH_PATH)
```

Figure 13: A simple fix of removing one small part to an error that required reading many different documents.


```

fetch

DESCRIPTION

    "fetch" downloads a file from the internet (if possible)

USAGE

    fetch code [, name [, state [, finish [, discrete [, multiplex
        [, zoom [, type [, async [, path ]]]]]]]]]]]

ARGUMENTS

    code = a single PDB identifier or a list of identifiers. Supports
    5-letter codes for fetching single chains (like 1a00A).

    name = the object name into which the file should be loaded.

    state = the state number into which the file should be loaded.

    type = str: cif, pdb, pdb1, 2fofc, fofc, emd, cid, sid {default: cif
    (default was "pdb" up to 1.7.6)}

    async_ = 0/1: download in the background and do not block the PyMOL
    command line {default: 0 -- changed in PyMOL 2.3}

PYMOL API

    cmd.fetch(string code, string name, int state, int finish,
        int discrete, int multiplex, int zoom, string type,
        int async, string path, string file, int quiet)

NOTES

    When running in interactive mode, the fetch command loads
    structures asynchronously by default, meaning that the next command
    may get executed before the structures have been loaded. If you
    need synchronous behavior in order to insure that all structures
    are loaded before the next command is executed, please provide the
    optional argument "async=0".

    Fetch requires a direct connection to the internet and thus may
    not work behind certain types of network firewalls.

```

Figure 14: PyMOL's command reference guide to 'fetch.' In order, it shows what fetch does, how to use it, the arguments/parameters it takes, how to use it with cmd, and additional notes.

Miscellaneous Updates

The user manual will be updated to include helpful tips. Major revisions will be done to the installation process section and using the View Options tab within ProMOL. The direction for the installation steps needed to be more concise and straightforward with more emphasis on finding where things such as ProMOL or PyMOL are installed on the user's computer. Knowing the locations for these files is extremely important to a successful installation.

Inside ProMOL folders is a readme.txt describing the general changes made and a promol-changes.txt with more specific changes made. Anything GitHub has recorded will also be saved to these files.

Results

ProMOL Version 5.5 (Updated PDB API)

As shown in Figure 15, ProMOL 5.5 was able to fetch PDB ID 1wht and tell PyMOL to show it in PyMOL viewer when coupled to PyMOL version 1.8. A simple replacement of an outdated URL as shown in Figure 16 solved the Connection Error when using Fetch PDB. All of the outdated URLs were removed and updated throughout the ProMOL directory, and a working version could finally be used.

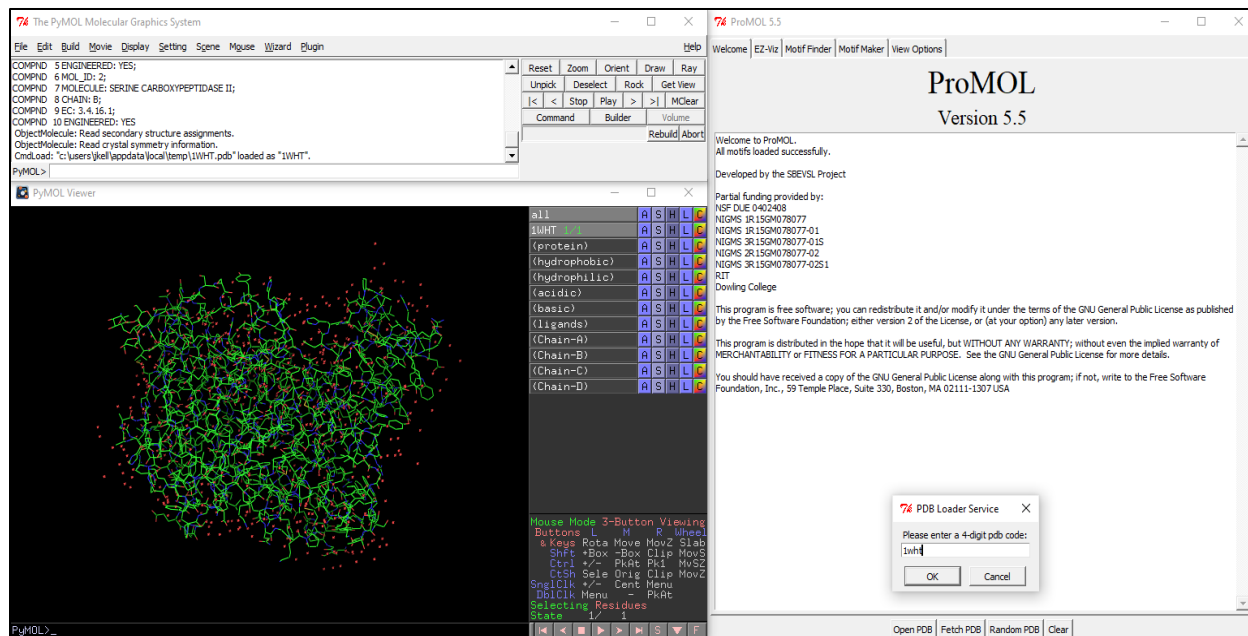
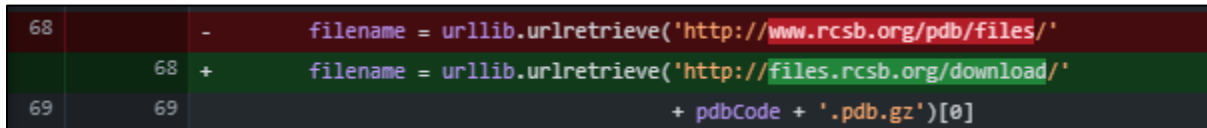


Figure 15: ProMOL and PyMOL were successfully communicating with each other after the API was updated. The molecule being displayed is PDB ID 1wht, a structure of the complex of L-Benzylsuccinate with wheat serine carboxypeptidase II.



```
68 - filename = urllib.urlretrieve('http://www.rcsb.org/pdb/files/'
69 + filename = urllib.urlretrieve('http://files.rcsb.org/download/'
+ pdbCode + '.pdb.gz')[0]
```

Figure 16: GitHub Desktop showing the changes being made to one line of code. The red highlighted line shows what is being removed. The green highlighted line shows what has been added. Anything else remained untouched.

Motif Maker was tested, as shown in Figure 17, using PDB ID 1WHS, another structure of the complex of L-Benzylsuccinate with wheat serine carboxypeptidase II with an EC # 3.4.16.6. This can be broken down as a class of hydrolases (3) acting on peptide bonds (3.4) and is a serine-type carboxypeptidase (3.4.16) D (3.4.16.6). The next few lines ask the user for residues to test for. For example, the first residue is Ser (Serine) on Chain A of the protein at position number 146 and the backbone residues are not involved in the active site. Pressing the Save button will save to C:\Users\jkell\AppData\Roaming\SBEVSL\ProMol\UserMotifs so that the user won't have to input this information every time. Figure 17 tests it against itself before testing against other proteins.

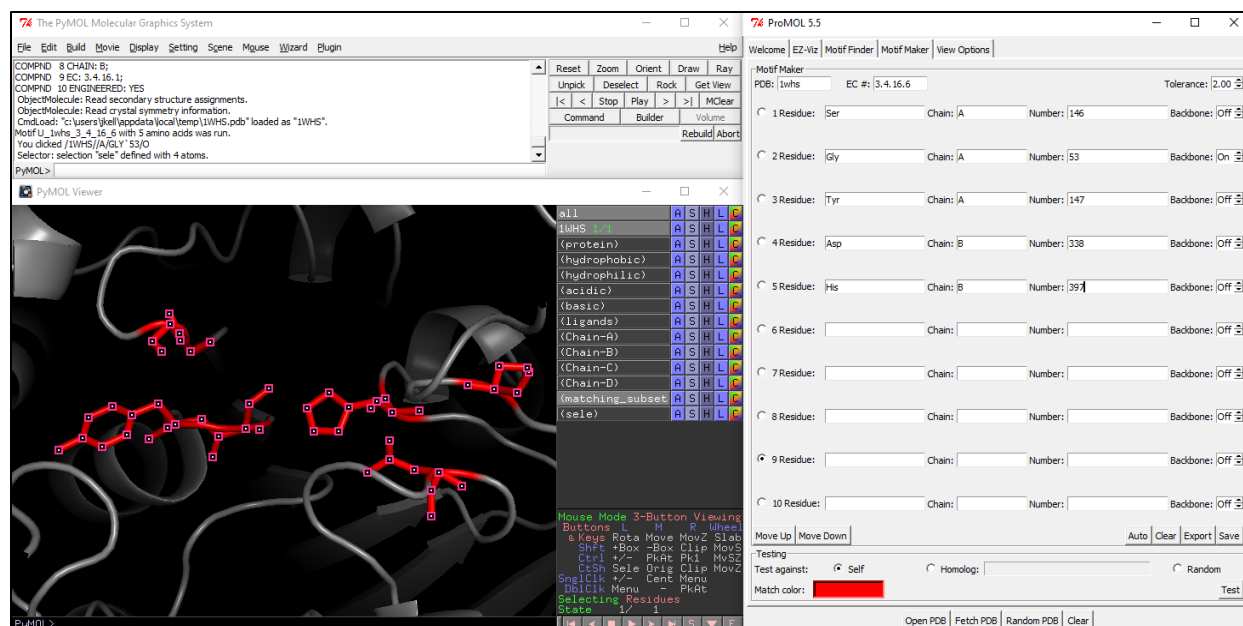


Figure 17: Five residues (right side) were filled in by the user and tested against itself. The result can be seen in the PyMOL viewer (bottom left side). The five selected red areas correspond to the five residues from the Motif Maker. The command line shows the behind-the-scenes work (upper left side).

Figure 18 shows the Motif Finder working as intended. Saved motifs show up under the results section categorized first by the PDB IDs to be searched then by the first EC number, which in this case is 3. The user can choose whether or not to show alignment of the selected motif and/or to show Root Mean Square Deviation (RMSD) values. RMSD values are useful in comparing multiple alignments and determining which fits best, or most similar. The lower the RMSD value, the more similar the two alignments would be [28]. PyMOL's viewer shows only the residues that produced a significantly close alignment from the saved motifs and queried PDB IDs. Specific RMSD values can be seen in the command line.

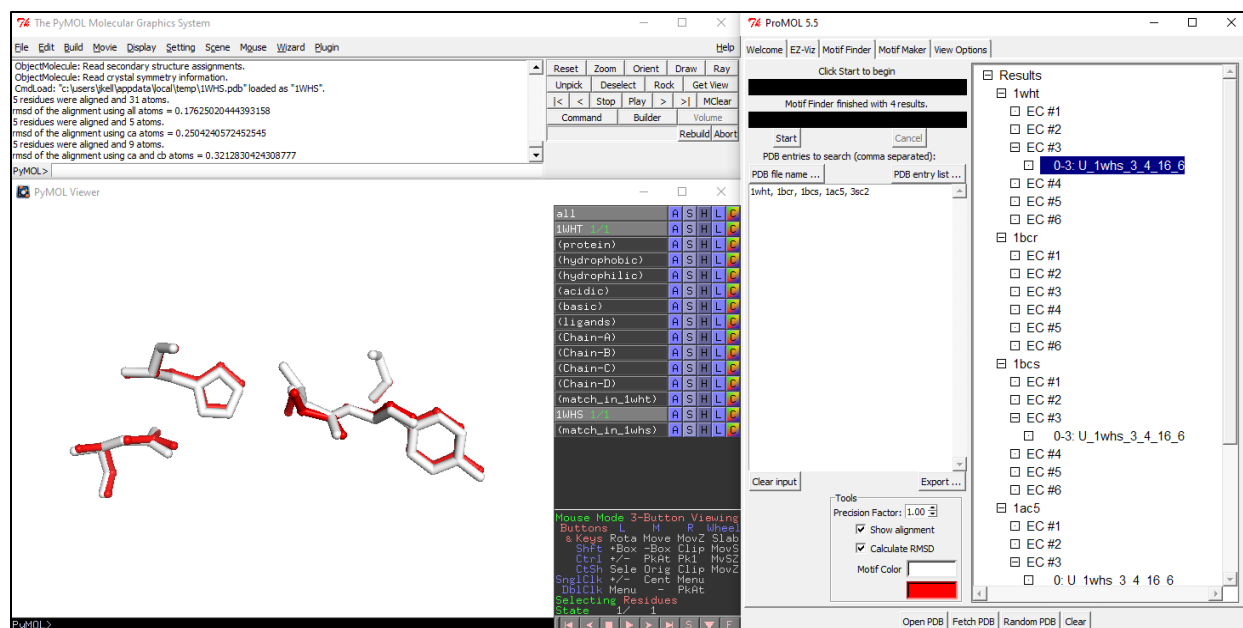


Figure 18: PDB entries are listed in the right window that are produced in response to the query. Here, 0-3: U_1whs_3_4_16_6 had been double clicked to tell PyMOL viewer to show alignment between the motif saved earlier and the one highlighted in blue. On the left side, the red color shows the motif saved by the user while the white color shows the PDB entry to be searched which is 0-3: U_1whs_3_4_16_6. RMSD values can be seen in the command line window (upper left side).

These were the features being tested in ProMOL 5.5 since they are the primary purpose of this plugin. Options from the other tabs had been tested with success as they deal with the appearance of the molecule in the PyMOL viewer. Moreover, examples of the behavior of these options will be discussed in further detail with ProMOL version 6.0.x.

ProMOL Version 6.0.X

Prior to migrating from Python 2 to Python 3, ProMOL would not even connect with PyMOL version 2.4. Successful translation would show a pop-up window telling the user that it had been installed without error. Currently, ProMOL works with older APBS plugin which can be found in PyMOL's dropdown menu under Plugin → Legacy Plugins → APBS Tools2.1 (placeholder). APBS Tools2.1 (placeholder) must be clicked before ProMOL is able to load. The Fetch PDB button still correctly shows the desired PDB protein in the PyMOL viewer.

Testing out the Motif Maker with the same parameters as ProMOL 5.5 shows similar results using PDB ID 1whs and EC# 3.4.16.6 with the same residues tested against itself.

PyMOL 2.4 now shows more detail than PyMOL 1.8 in the viewer area as can be seen in Figure 19. The residues show double bonds instead of every bond looking as if it were single. This makes it easier to verify the identity of a residue instead of having to select each one individually.

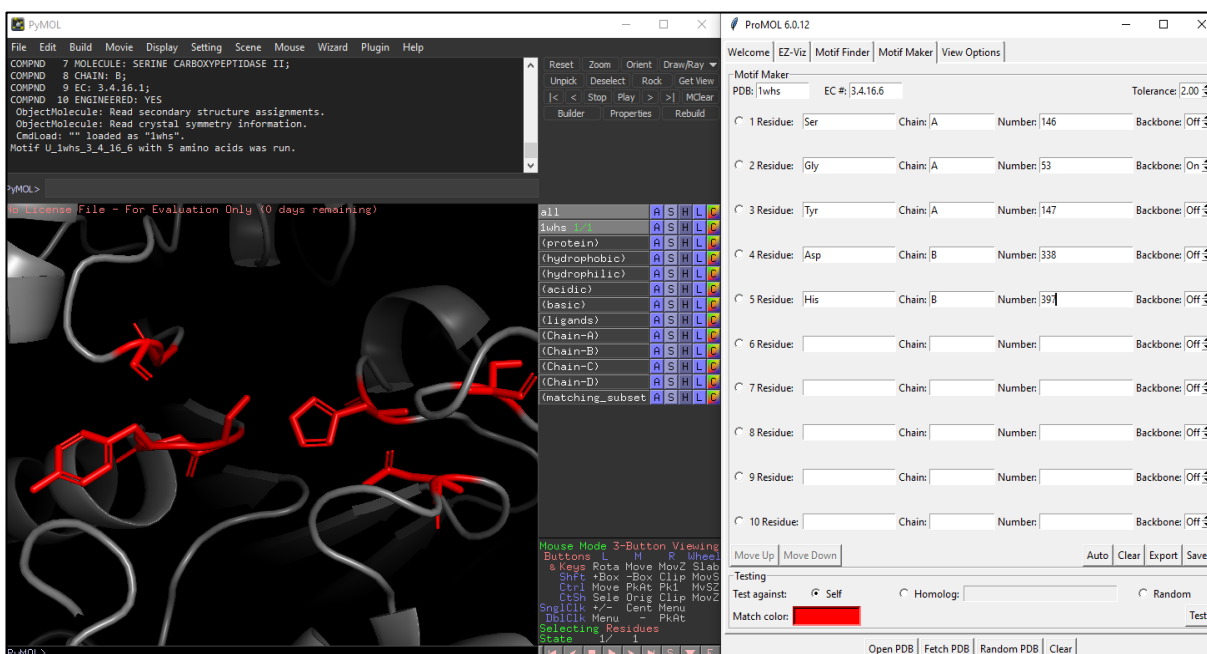


Figure 19: ProMOL 6.0.12 testing 1whs EC# 3.4.16.6 against itself. Residues show up in red on the window with new double bonds in PyMOL 2.4.

The next step tests the Motif Finder. Half of the program works before an unexpected result occurs. As can be seen in Figure 20, the window on the right was populated with the same information as ProMOL 5.5 in Figure 18. Unfortunately, a window pops up that states: “Error in background function.” Even though “Show alignment” was selected, only a part of one molecule (1wht) was displayed in the viewer. Molecule 1whs was not shown overlapping 1wht. This also

means that RMSD values could not be calculated. This gap will need to be addressed in future development of ProMOL.

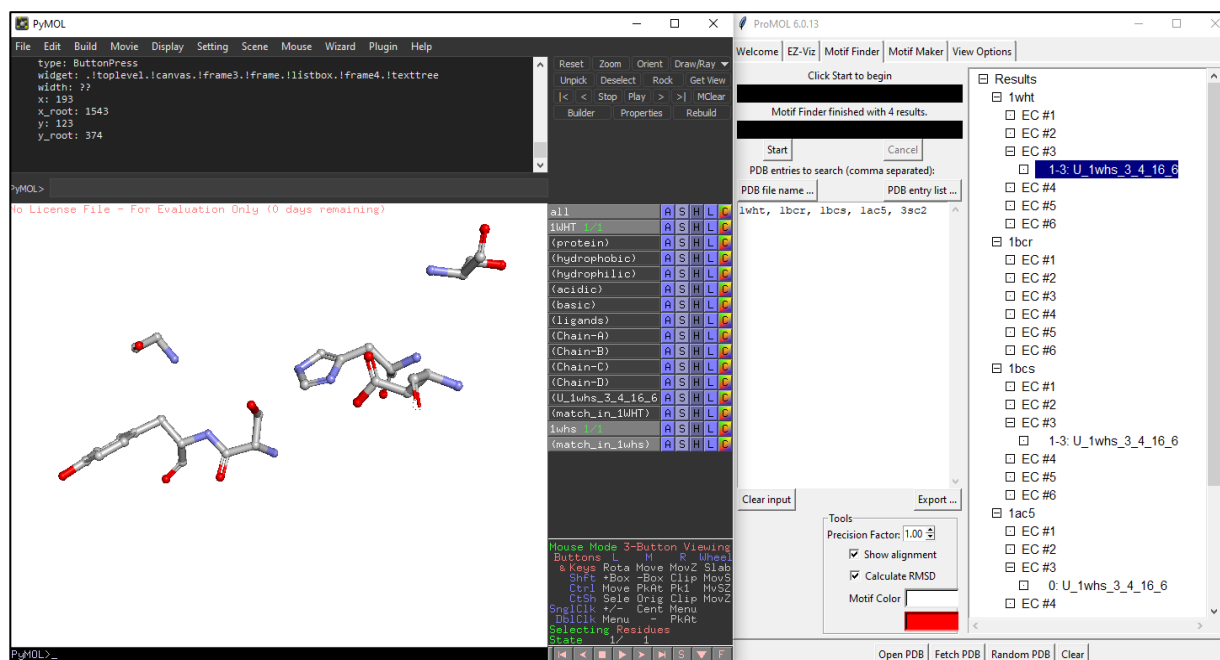


Figure 20: ProMOL 6.0.12 Is not correctly communicating to PyMOL to show alignment of the two molecules. Part of the error message can be seen in the command line in the upper left.

The primary features have been discussed above and the secondary features are next. While not the main focus of the program, ProMOL has made viewing molecules in different ways accessible via the EZ-Viz and View Options tab. As stated earlier, the different sub-categories named Cartoon, Spheres, Sticks, Surface, and Ambient Light aren't easily understood when the user adjusts the sliders because an expected change to the displayed molecule doesn't happen. To help alleviate this issue, a small explanation was added below the sub-categories on the View Options tab as seen in Figure 21. Adding an additional text message to the bottom wasn't as simple as adding a new line of code because it ended up shifting the location of all the other options. Certain configurations displayed the right column of sub-categories completely off-screen unless the user expanded the window or options would overlap each other.

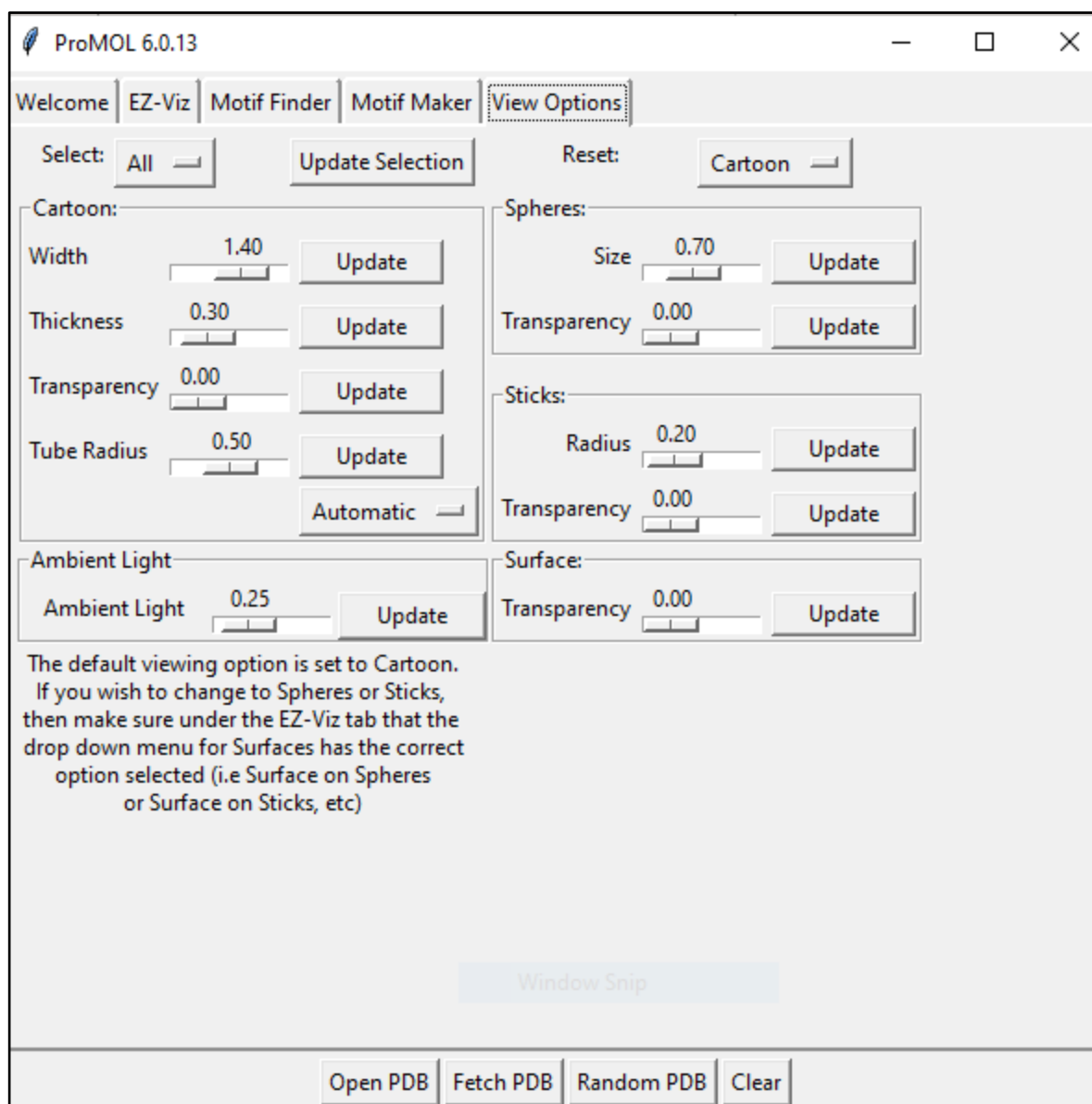


Figure 21: A new explanation at the bottom of the View Options tab about how to use the sliders.

Miscellaneous Updates

Additions to the user manual involved the installation process and opening of ProMOL in PyMOL. It is important to note that only one operating system, Windows, has been updated for installation due to unavailability of a Linux/Unix or Macintosh operating system. The user manual can be found within the ProMOL folder labeled “ProMOL_User_Guide.pdf” and the installation process for ProMOL 6.0.x can be found on page 27 as seen in Figure 22. While

GitHub serves as a source of version control and documentation of ProMOL, the readme.txt file has had more revision descriptions added to the list under the Change Log section. For this project, ‘Revision 368’ through ‘Revision 372’ has been appended in the change log as shown in Figure 23.

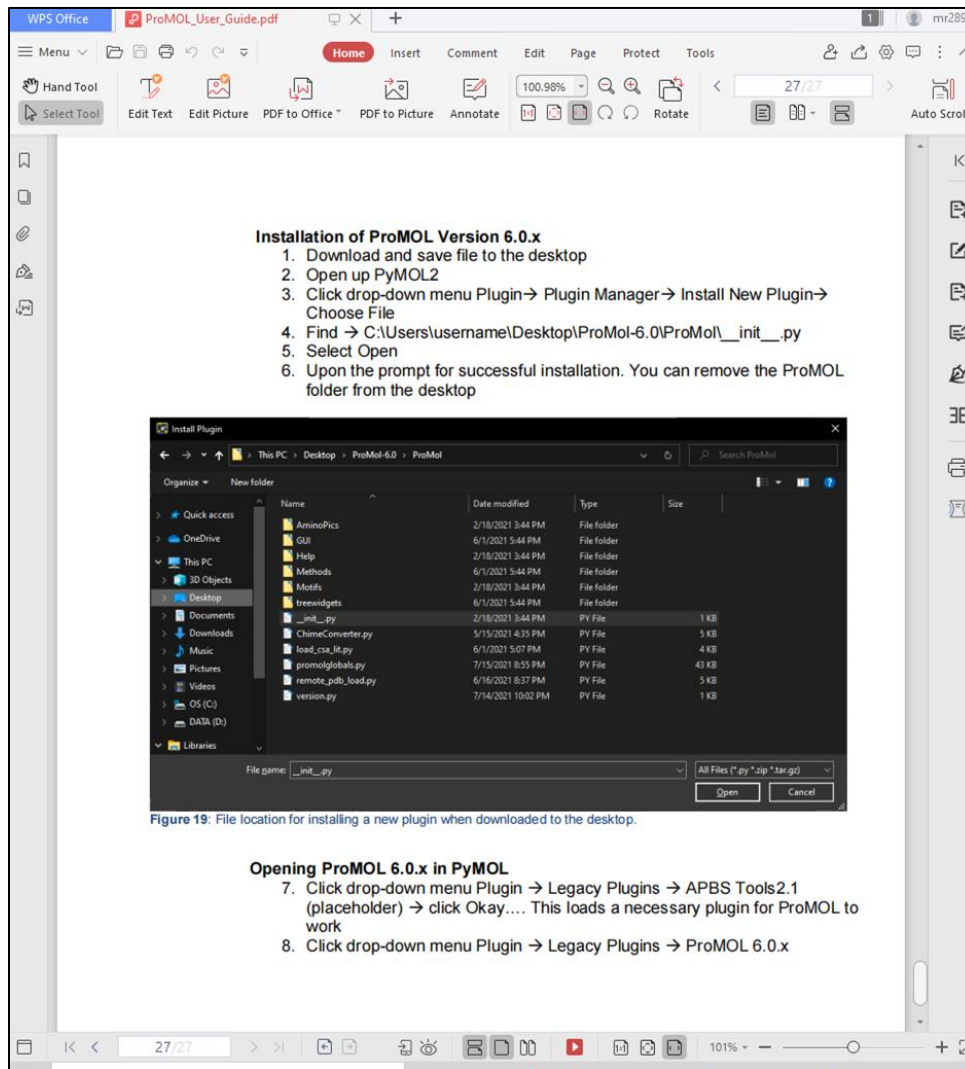


Figure 22: The ProMOL_User_Guide.pdf file showing the addition of the installation process for ProMOL Version 6.0.x and how to open it in PyMOL 2.

```

Change log:
Revision 372
Modified Wednesday, July 14, 2021 by Mariah Robertson
-Added an explanation to the View Options tab at the bottom (view.py)

Revision 371
Modified Wednesday, June 16, 2021 by Mariah Robertson
-more 2to3 conversion
-Fetch button = fixed upper() function (remote_pdb_load.py)
-Fetch button = fixed opening/writing function to a file by specifying it as text instead of binary
-Testing Motif Finder (Test Button) = moved upper() out of FetchPDB function.
-motif.py changed write() to specify string not byte

Revision 370
Modified Saturday, June 1, 2021 by Mariah Robertson
-moved treewidget folder inside ProMol
-updated import path files
-updated files to python 3
-removed async=0 parameter in multiple files

Revision 369
Modified Saturday, May 15, 2021 by Mariah Robertson
All file changes:
-changed Tkinter to tkinter
promolglobals.py changes:
-2to3 Python
-removed async=0 (pymol made it the default value)
-removed del x (optimization - list comprehension is enough)
-Moved remote_pdb_load to into ProMol folder for easier loading into PyMol

Revision 368
Modified Thursday, May 13, 2021 by Mariah Robertson
-Updated the api fetch url

Revision 367
Modified Thursday, July 31, 2014 by Herbert J. Bernstein

Make state of EC number entry boxes consistent with their content.
If there is a number is it enabled. If there is a blank or zero
and we are below the top level it is disabled. This fixes a bug
report by Talia McKay.

```

Figure 23: This readme.txt file contains a section for the changes made with dates, author(s) specified, and a short description of the modifications. Each time a major revision is made to ProMOL's code, it is documented in this text file as another form of documentation.

Discussion

Much needed updates to ProMOL have allowed it to work in conjunction with newer versions of PyMOL. There are obvious rooms for improvement as not all modules are quite working yet, but a large stride in the right direction has been taken. Before any updates had been done, ProMOL was unable to function at all because of the new API changes to PDB. Now, ProMOL 5.5 (<https://github.com/mr2893/promol-5.5>) works with PyMOL 1.8 and it is open to

the public. While only a few lines of code needed to be changed, it is vital knowing *where* the problem was happening and *what* needed to be changed. Altering code in the wrong place can break the program even further and in the worst case, kill the project. Having excellent documentation plus backups of the program can save time and money. GitHub has more than sufficed for this project. Their GitHub Desktop application has especially made documenting any changes a breeze. Their interface is easy to navigate, and users can quickly look at previous changes made. In this project, changes were reverted to the original code when it clearly was not executing properly or had unintended consequences. It was easy as a couple of clicks of a button, thus saving a lot of time. ProMOL 5.5 was a success in terms of accomplishing a working version with the older version – PyMOL 1.8.

With ProMOL 6.0.x (<https://github.com/mr2893/ProMol-6.0>), I was able to make some major strides in working with the most up to date software and Python language. First, all Python files were translated using the 2to3 tool allowing ProMOL and PyMOL to be compatible with one another. While they could now communicate with each other, that did not mean they could work together as intended. Not only was Python and PDB updating their own projects, PyMOL updated some of their commands and API, too. Luckily, all three of them had some sort of documentation that could be referenced.

Updating to ProMOL 6.0.x was much more difficult to fix since the errors were more numerous and far more varied compared to updating to ProMOL 5.5. Unfortunately, not everything was in working order, but about 3/4 tabs were able to function without errors. EZ-Viz, Motif Maker, and View Options had no issues interacting with PyMOL 2.4. The Motif Finder tab partially works when searching PDB entries for predetermined motifs. It will successfully populate the expected list in the right-hand column, but double clicking on the predetermined

motifs with “Show alignment” selected does not produce the correct results in the PyMOL viewer. Rather than showing both molecular sections, only the queried PDB entry is shown. Lastly, 3/4 buttons located along the bottom of ProMOL work flawlessly. “Open PDB” opened a PDB file on the computer correctly and show it in the PyMOL viewer. “Fetch PDB” pulled the desired PDB ID from the website and the PyMOL viewer displayed it. “Clear” just removed any existing molecule in the PyMOL viewer. The “Random PDB” button produces an error that could be traced using the command line, but the command line states: ‘Error-fetch: unable to load X.’ It is currently unclear if the URL associated with clicking “Random PDB” no longer works or there is still an underlying problem with the code. Regardless, ProMOL as a whole has ascended to new heights where its legacy can be continued by another.

Conclusion

Open-source software is important to advancing society and pushing the boundaries of scientific knowledge. Allowing anyone to use these types of tools can assist in the discovery of new and amazing molecules that may have never been characterized. Characterizing new molecules has the potential to illuminate new ways to combat diseases/pathogens or develop different approaches to solving man-made problems. The possibilities are endless when intellect is not restricted by expensive software. Other developers are even working on homology-based motif generation. A collaboration between developers may accelerate the process of creating an unparalleled algorithm.

Software that is widely used by researchers or anyone must have a user interface that is smooth and seamless. Atrocious designs leave a bad impression that will cause people to immediately give up trying to learn the software. If navigating the software is not intuitive, end-

users will likely seek out other applications that do the same thing only better. ProMOL is far from perfect, but the essential necessities are laid out in a reasonable fashion.

Future Opportunities

In future iterations, ProMOL could be simplified in its design by combining a couple of the tabs that actually interact with each other. As it currently stands, some confusion arises when certain menus appear to not work. This may require updating the aesthetic to make the important options stand out more. Tooltips are always helpful when learning a program. Cluttering the window with superfluous tips should be avoided. However, hovering over an option to reveal a tooltip with an extra explanation can be very useful for the user. While the user manual may serve this purpose, convenience goes a long way to bring in new users.

Only the Windows installation process has been introduced in the user manual for ProMOL 6.0.x. Linux/Unix and Macintosh users are currently not included and the manual needs to be updated for them. Leaving out part of the user-base is not good practice nor respectful, especially when there exist instructions for them with older versions.

The APBS Tools2.1 plugin has been updated to APBS Electrostatics. Currently APBS Tools2.1 appears to be used in conjunction with ProMOL and further updates to ProMOL may be required to work with the Electrostatics version. In order to open up ProMOL 6.0.x, the user has to click APBS Tools2.1 first, then click on ProMOL for it to work. It is unclear when or if the original plugin will be discontinued sometime in the near future. Another possibility that may fix this problem involves changing from Tkinter to PyQt5 Python library for creating a GUI framework. According to PyMOL's Plugin Architecture page, legacy plugins are still supported that use the Tkinter package. This update may remove ProMOL from the legacy plugins menu in

PyMOL and remove the need to play around with the outdated APBS Tools2.1 plugin. As a final note, Table 1 shows a list to quickly glance at for all the buttons in ProMOL that needs troubleshooting.

Interactable Options	Functional	Not Functional	Notes/Thoughts
Open PDB	x		
Fetch PDB	x		
Random PDB		x	Binary vs Text - seems to be only storing one letter/number in the database list instead of four (for the PDB ID)
Clear	x		
Surfaces	x		
Cartoons	x		
By Residue	x		
Preset Movies		x	The only drop-down option that doesn't work is Highlight Chains. Everything else works
Miscellaneous	x		
Electron Density	x		
Roving	x		
Roving Detail	x		Move the slider then select the same Roving Option to see differences
Select	x		
Show	x		
Color	x		
Update Selection	x		
Hide	x		
Stereo	x		
Background Color	x		
Color Space	x		
Internal GUI	x		
Start	x		
Cancel	x		
PDB file name...	x		
PDB entry list ...	x		
Clear input	x		
Export...	x		
Show alignment		x	The checkbox is fine, but double clicking the results is what produces the error
Calculate RMSD		x	This is probably tied to Show alignment and may fix itself once the other is fixed

PDB field	x		
EC #: field	x		
Radial Button	x		Used with the move up/move down button
Residue	x		
Chain	x		
Number	x		
Backbone	x		
Tolerance	x		
Move Up	x		
Move Down	x		
Auto		x	Unclear of the purpose of this button (not explained in user manual)
Clear	x		
Export	x		It does produce a traceback call if nothing is in the fields (maybe make that a pop-up window with explanation?)
Save	x		
Self	x		
Homolog	x		
Random		x	Same issue as Random PDB button
Select	x		
Update Selection	x		
Reset	x		
Width	x		
Thickness	x		
Transparency	x		
Tube Radius	x		
Ambient Light		x	Might be the ambient command? https://pymolwiki.org/index.php/Ambient
Size (Spheres)	x		
Transparency (Spheres)	x		
Radius (Sticks)	x		
Transparency (Sticks)	x		
Transparency (Surface)	x		

Table 1: The left column shows all interactable options in ProMOL. Each colored category corresponds to a tab in the GUI interface. Yellow indicates the buttons seen on all tabs, blue corresponds to EZ-Viz, green corresponds to Motif Finder, purple corresponds to Motif Maker, and red corresponds to View Options. The next two columns indicate if the interactable options are working as intended or not. The last column contains information that may be helpful to the next programmer for successfully troubleshooting the issue.

Acknowledgments

Dr. Paul Craig for the opportunity of this project, reviewing, and editing my thesis.

Dr. Gary Skuse for finding this project for me and providing helpful suggestions that would enhance the project.

Dr. Joe Geigel for agreeing to be on my committee and providing helpful feedback during my proposal defense.

References

1. Young, J. Y., Westbrook, J. D., Feng, Z., Sala, R., Peisach, E., Oldfield, T. J., Sen, S., Gutmanas, A., Armstrong, D. R., Berrisford, J. M. *et al.* (2017) OneDep: unified wwPDB system for deposition, biocuration, and validation of macromolecular structures in the PDB archive. *Structure*, **25**, 536–54
2. Bank, R. P. D. (n.d.). *RCSB PDB*. Retrieved February 17, 2021, from <https://www.rcsb.org/docs/file-downloads/http-and-https-services>
3. Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., & Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, **28**(1), 235–242. <https://doi.org/10.1093/nar/28.1.235>
4. Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Di Costanzo, L., Christie, C., Dalenberg, K., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranović, V., Guzenko, D., Hudson, B. P., Kalro, T., Liang, Y., ... Zardecki, C. (2019). RCSB Protein Data Bank: Biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Research*, **47**(D1), D464–D474. <https://doi.org/10.1093/nar/gky1004>
5. *Plugins Tutorial—PyMOLWiki*. (n.d.). Retrieved March 4, 2021, from https://pymolwiki.org/index.php/Plugins_Tutorial
6. *PromOL*. (n.d.). Retrieved February 7, 2021, from <http://www.promol.org/home>
7. Hanson, B., Westin, C., Rosa, M. *et al.* Estimation of protein function using template-based alignment of enzyme active sites. *BMC Bioinformatics* **15**, 87 (2014). <https://doi.org/10.1186/1471-2105-15-87>
8. *Porting Python 2 Code to Python 3—Python 3.9.2 documentation*. (n.d.). Retrieved March 4, 2021, from <https://docs.python.org/3/howto/pyporting.html>
9. Yana Rose, Jose M. Duarte, Robert Lowe, Joan Segura, Chunxiao Bi, Charmi Bhikadiya, Li Chen, Alexander S. Rose, Sebastian Bittrich, Stephen K. Burley, John D. Westbrook. RCSB Protein Data Bank: Architectural Advances Towards Integrated Searching and Efficient Access to Macromolecular Structure Data from the PDB Archive, *Journal of Molecular Biology*, 2020. DOI: [10.1016/j.jmb.2020.11.003](https://doi.org/10.1016/j.jmb.2020.11.003)
10. Mulesoft. (2018, February 14). *What is an API? (Application Programming Interface)*. MuleSoft. <https://www.mulesoft.com/resources/api/what-is-an-api>

11. Shi, L., Zhong, H., Xie, T., & Li, M. (1970). *An Empirical Study on Evolution of API Documentation*. 416–431. https://doi.org/10.1007/978-3-642-19811-3_29
12. *What is an API?* (2019). Redhat.com. <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>
13. *What is Query Language? - Definition from Techopedia*. (n.d.). Techopedia.com. <https://www.techopedia.com/definition/3948/query-language>
14. *Sunsetting Python 2*. (n.d.). Retrieved from <https://www.python.org/doc/sunset-python-2/>
15. Cannon, B. (2015, December 16). *Why Python 3 exists*. Tall, Snarky Canadian. <https://snarky.ca/why-python-3-exists/>
16. Delano WL. The PyMOL molecular graphics system. Schrodinger, LLC., San Carlos, CA, USA
17. Pelley, J. W. (2012). *Protein Structure and Function. Elsevier's Integrated Review Biochemistry*, 19–28. doi:10.1016/b978-0-323-07446-9.00003-9
18. Jones, S., Barker, J. A., Nobeli, I., & Thornton, J. M. (2003). Using structural motif templates to identify proteins with DNA binding function. *Nucleic Acids Research*, 31(11), 2811–2823. <https://doi.org/10.1093/nar/gkg386>
19. Porter CT: The catalytic site atlas: a resource of catalytic sites and residues identified in enzymes using structural data. *Nucleic Acids Res*. 2004, 32: D129-D133. [10.1093/nar/gkh028](https://doi.org/10.1093/nar/gkh028)
20. McKay, T., Hart, K., Horn, A., Kessler, H., Dodge, G., Bardhi, K., Bardhi, K., Mills, J. L., Bernstein, H. J., & Craig, P. A. (2015). Annotation of proteins of unknown function: Initial enzyme results. *Journal of Structural and Functional Genomics*, 16(1), 43–54. <https://doi.org/10.1007/s10969-015-9194-5>
21. Furnham, N., Holliday, G. L., de Beer, T. A. P., Jacobsen, J. O. B., Pearson, W. R., & Thornton, J. M. (2014). The Catalytic Site Atlas 2.0: Cataloging catalytic sites and residues identified in enzymes. *Nucleic Acids Research*, 42(D1), D485–D489. <https://doi.org/10.1093/nar/gkt1243>
22. Ribeiro, A. J. M., Holliday, G. L., Furnham, N., Tyzack, J. D., Ferris, K., & Thornton, J. M. (2018). Mechanism and Catalytic Site Atlas (M-CSA): A database of enzyme reaction mechanisms and active sites. *Nucleic Acids Research*, 46(D1), D618–D623. <https://doi.org/10.1093/nar/gkx1012>

23. McDonald, A. G., Boyce, S., & Tipton, K. F. (2009). ExplorEnz: The primary source of the IUBMB enzyme list. *Nucleic Acids Research*, 37(suppl_1), D593–D597. <https://doi.org/10.1093/nar/gkn582>
24. McDonald, A. G., & Tipton, K. F. (2014). Fifty-five years of enzyme classification: Advances and difficulties. *The FEBS Journal*, 281(2), 583–592. <https://doi.org/10.1111/febs.12530>
25. Osipovitch, M., Lambrecht, M., Baker, C., Madha, S., Mills, J. L., Craig, P. A., & Bernstein, H. J. (2015). Automated protein motif generation in the structure-based protein function prediction tool ProMOL. *Journal of Structural and Functional Genomics*, 16(3), 101–111. <https://doi.org/10.1007/s10969-015-9199-0>
26. Regebro, Lennart. (2015) Supporting Python 3: An In-depth Guide. <http://python3porting.com/toc.html>
27. Bank, R. P. D. (n.d.). *Announcement: Legacy RCSB PDB APIs Will Be Discontinued November 2020*. Wwww.rcsb.org. Retrieved July 16, 2021, from <https://www.rcsb.org/news?year=2020&article=5eb18ccfd62245129947212a&feature=true>
28. Reva, B., Finkelstein, A., & Skolnick, J. (1998). What is the probability of a chance prediction of a protein structure with an rmsd of 6 Å? *Folding and Design*, 3(2), 141–147. [https://doi.org/10.1016/S1359-0278\(98\)00019-4](https://doi.org/10.1016/S1359-0278(98)00019-4)