FURY: Ribbon and Molecular Surface Representations for Proteins

GSoC 2021 Proposal

Organization: Python Software Foundation

Sub - organization: FURY

Contents

About Me

Personal Info

- Name: Sajag Swami
- Location: India
- Timezone: India (UTC+5.30, EST+9.30)
- Github: SunTzunami
- LinkedIn

University Info

- University: Indian Institute of Technology, Roorkee
- Major: Production and Industrial Engineering
- Current year: 2nd year (2023 expected graduation)
- Degree: Bachelor of Technology (B.Tech)

Brief Introduction

I am a second year student enrolled in Production and Industrial Engineering (4 year course) at IIT Roorkee. I developed interest in programming especially in computer graphics in my high school days while working on my high school computer science project in which I utilised the graphics library of C++ 3.0 to create visuals. Since then, I have worked on multiple personal and group projects which mostly involved python and ranged across diverse fields like statistical analysis, 2D animation, and quantitative finance. All these projects can be found on my Github profile.
In addition to Python, I possess a good knowledge of R and some basic knowledge of machine learning and biology which I gained during my research internship. I also possess basic knowledge of biomolecules which I acquired while preparing for various national level exams.
I came to know about FURY back in September 2020 and have since then spent quite some time reading the documentation of FURY and also read a bit of the documentation of VTK. I've experimented with the

tutorials/demos made using FURY and made some tutorials/demos of my own and added some of them to the FURY repo. I find FURY fascinating because of its ease of use and versatility as a scientific visualization library and enjoy working with it.

Programming Skills

Programming Languages and Frameworks:

Fluent in Python, C/C++
Moderately experienced in R, SQL
Sound knowledge of Object Oriented Programming

Development Environment

Ubuntu 20.04.2 (mostly used) or Windows 10
Fully customised Visual Studio Code as primary editor
Good grasp on concepts of Git

Code Contributions

I have installed FURY locally and created PRs to address some issues. In addition to these PRs, I've also created PRs to add new tutorials and features in the FURY github repo. Following are the PRs that I have worked on -

#351 - Opacity bug fix for point and sphere actors (**merged**)

It was a **Bug Fix PR** in which I ensured that the opacity argument passed to point and sphere actors manipulated opacity of the actors (this was earlier not happening). This PR closed the issue #335.

#362 - An example for a time-varying 2D wave function (**opened**)

I tried to use a surface actor to render a time varying 2D wave function. It aimed to resolve #324.

#376 - Added animations for some electromagnetic phenomena (**merged**)

I had an idea of creating visuals for some basic electromagnetic phenomena and created PR for the same. It was a **Documentation PR** in which I added two tutorials -

- Propagation of an an electromagnetic wave
- Helical motion of a charged particle under the influence of a combined magnetic and electric field.

#383 - Minor documentation fix (**merged**)

Documentation PR

#385 - Fixed the example for superquadric function (**merged**)

Some variables were not initialised and some were initialized incorrectly, as a result, the example was not working. That was fixed. It was a **Documentation** and **Enhancement PR.**

#388 - Added simulation for brownian motion (**merged**)

I had an idea of creating a simulation of brownian motion via FURY. Made a tutorial for the same. It was a **Documentation PR.**

#393 - Added primitive and actor for triangular prism, added tests too (**merged**)

I made a primitive for a triangular prism and created an actor for the same. Added unit tests for the actor and primitive respectively. It was a **New Feature PR.**

#404 - Parametric functions- actor, primitives **(opened)**

I made an actor which generates parametric surfaces like Möbius strip, Klein bottle etc. and made primitives for the same. Added tests for the newly created actor and primitives.

Some issues that I've raised -

#335 - _opacity argument for point doesn't seem to work:

_opacity argument for point and sphere actors didn't work as the value stored in the argument was not being used to manipulate the opacity.

#363 - Minor error in documentation of create_colormap function

This was with respect to the create_colormap function.
According to the comment describing the auto variable, if auto is True then `v` is interpolated to [0, 10] from `v.min()` to `v.max()`. In the actual code however, `v` is being interpolated to [0, 1] from `v.min()` to `v.max()`.

Project Information

Abstract

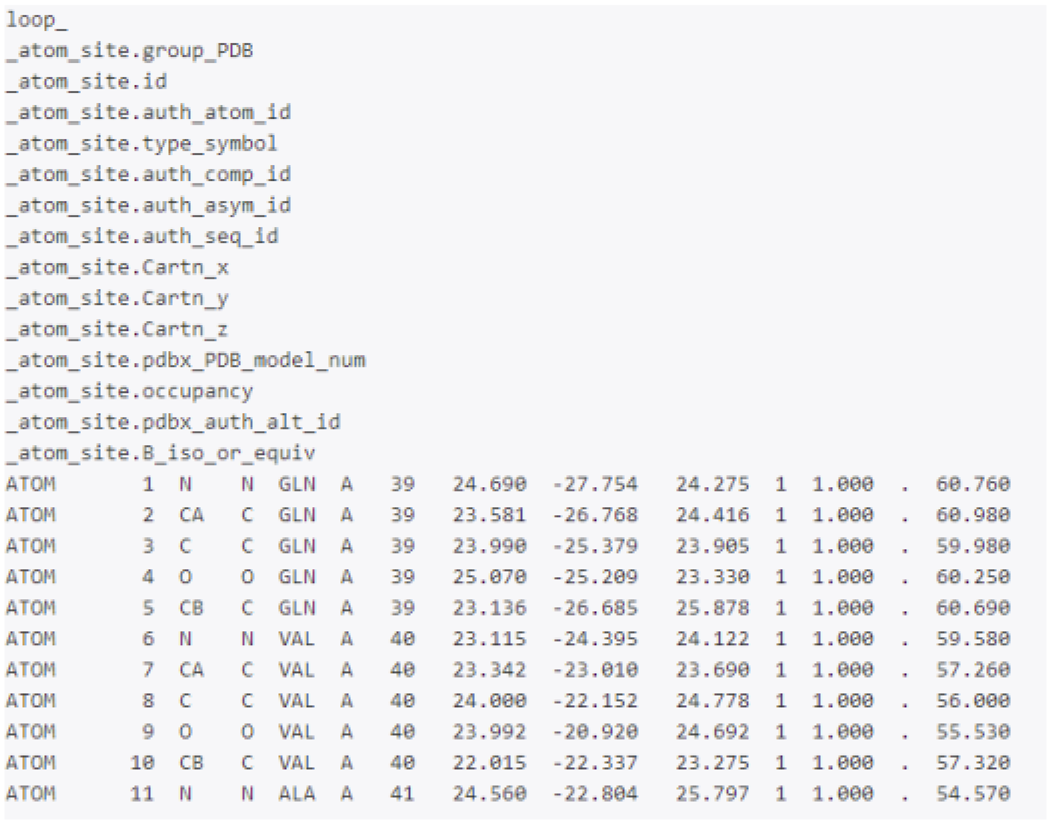
I will be working on adding a new functionality to FURY which shall enable users to visualize various types of proteins via different representations like Richardson aka Ribbon diagrams and molecular surface diagrams. As a part of my stretch goals, I'd like to expand protein representations via other representations including -

- Stick
- Ball and stick
- Wire
- Pipes and Planks
- Sphere

Detailed description of the project

Atomic Level Data

Details about the protein to be visualized such as its atomic level data (coordinates of the atoms) will be essential for generating the visuals. Molecular structural data of a protein is stored in a PDBx file. PDBx/mmCIF format is the latest standard for files containing atomic coordinates of proteins (it replaced the PDB format and became the standard PDB archive format in 2014).The format is based on a context-free grammar. PDBx/mmCIF has a simple grammar. Data is presented in either key-value or tabular form. It is much easier to parse than the record-oriented PDB format. Parsing PDBx/mmCIF files to obtain atomic coordinates will be the first step and can be done by using Biopython which has a module (Bio.PDB.MMCIFParser) that focuses on working with crystal structures of biological macromolecules and has a MMCIFParser object which will help in extracting atomic level data (coordinates of the atoms). A snapshot of PDBx/mmCIF file -



```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.auth_atom_id
_atom_site.type_symbol
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_seq_id
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.pdbx_PDB_model_num
_atom_site.occupancy
_atom_site.pdbx_auth_alt_id
_atom_site.B_iso_or_equiv
ATOM      1  N    N  GLN  A   39   24.690  -27.754   24.275  1  1.000  .  60.760
ATOM      2  CA   C  GLN  A   39   23.581  -26.768   24.416  1  1.000  .  60.980
ATOM      3  C    C  GLN  A   39   23.990  -25.379   23.905  1  1.000  .  59.980
ATOM      4  O    O  GLN  A   39   25.070  -25.209   23.330  1  1.000  .  60.250
ATOM      5  CB   C  GLN  A   39   23.136  -26.685   25.878  1  1.000  .  60.690
ATOM      6  N    N  VAL  A   40   23.115  -24.395   24.122  1  1.000  .  59.580
ATOM      7  CA   C  VAL  A   40   23.342  -23.010   23.690  1  1.000  .  57.260
ATOM      8  C    C  VAL  A   40   24.000  -22.152   24.778  1  1.000  .  56.000
ATOM      9  O    O  VAL  A   40   23.992  -20.920   24.692  1  1.000  .  55.530
ATOM     10  CB   C  VAL  A   40   22.015  -22.337   23.275  1  1.000  .  57.320
ATOM     11  N    N  ALA  A   41   24.560  -22.804   25.797  1  1.000  .  54.570
```

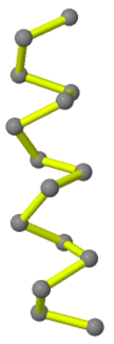
Richardson aka Ribbon diagrams

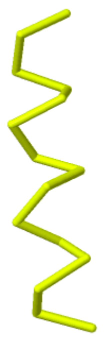
Brief info about the structure of Ribbon diagrams:

Ribbon diagrams are generated by interpolating a smooth curve through the polypeptide backbone. α-helices are shown as coiled ribbons or thick tubes, β-strands as arrows, and lines or thin tubes for random coils. The direction of the polypeptide chain may be indicated by a colour ramp along the length of the ribbon.

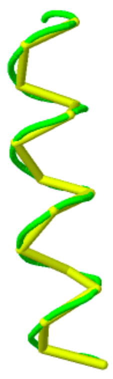
Steps involved in constructing Ribbon diagrams:

1. Collect the coordinates of alpha carbons from the PDBx file of the protein being rendered.
2. Generate backbone trace of the protein by generating a curve through alpha carbons.


Backbone trace with αC atoms (grey spheres)


Backbone trace alone

3. Apply smoothing algorithms to the generated backbone trace to convert it into smoothed backbone trace. (Potential smoothing algorithms that might be used: cubic B-spline or Hermite spline)


Smoothed backbone trace

Smoothed backbone trace alone

4. The ribbon is a smoothed backbone trace expanded in width. We can generate ribbon curves by translating the curve parallel to the helix axis.

Ribbon

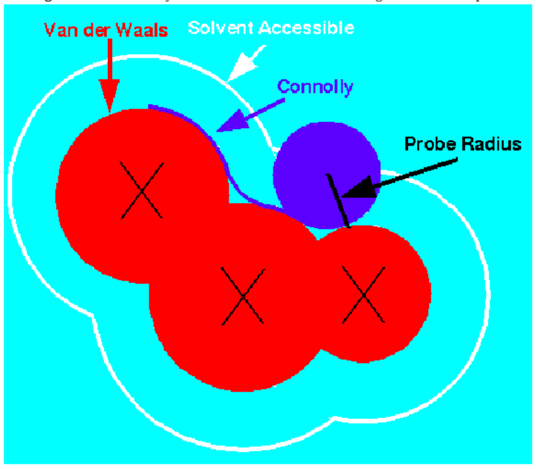
Ribbon alone

5. Add the other features like beta strands and loops, shading and colors etc.

<u>Molecular Surface Diagrams</u>

A common representation is due to Connolly. It virtually rolls a 'water' probe ball (1.4-1.8 Å diameter) over the Van der Waals surface, smoothing the surface and bridging narrow crevices, which are inaccessible to the solvent. This partitions the surface into convex, concave and saddle patches according to the number of contact points between the surface atoms and the probe ball. As Output, the representation consists of points + normals to the surface. These are sampled according to some required sampling density (e.g. 10 pts/Å^2).

Figure 13.10: Connolly's Solvent Accessible Surface. Image taken from http://www.chem.leeds.ac.uk/ICAMS/eccc/cangaroo.html.



There have been many advances in molecular surface visualization since Conolly's proposal and newer approaches by Lin et al and Ryu et al might prove more efficient. As such, implementing molecular surface diagrams will be done by comparing rendering quality, speed taken to render by the various methods discussed above (and/or other parameters as per the mentor's suggestions). Mentor's insights will prove beneficial when selecting which method to use for the generation of protein surfaces.

Creating demos

After writing the code for generating ribbons and molecular surface diagrams, I'll add demos to show users how they can use the newly added features to render protein structures.

References

Bio.PDB.MMCIFParser module
PDBx/mmCIF General FAQ
Backbone representations
Molecular Surface Representation

I've collected and stored some research papers that could help in constructing ribbon diagrams and molecular surfaces here.

Stretch Goals

After the implementation of ribbon diagrams and molecular surface, I aim to expand Protein representation via other models like-

- Stick
- Ball and stick
- Wire
- Pipes and Planks
- Sphere

I'll explain each model with a tutorial/demo for the users. I also hope to add a feature of custom colormaps for the different structures of proteins which will let the users use custom colormaps to color the structures.
If I still possess time after implementing the models mentioned above, I plan on creating new physics simulations (using FURY and pybullet).

Project Timeline

Period	**Goal/Milestone**
	Community Bonding Period May 17, 2021 - June 7, 2021
Week 1 May 17- May 24	● Get acquainted with the mentors and the admins of the project and discuss the vision of the project and gain insight about implementing the features. ● Decide which of the two main structures should be worked upon first and finalize implementation details for the structures.
Weeks 2, 3 May 25 - June 7	● Identify and merge any pending pull requests that could aid the project and otherwise. ● Discuss about the stretch goals and decide the order of their implementation.
	Commencement of the coding period
Week 4 June 7 - June 14	● Learn how to interpret and parse PDBx files. ● Write code to extract atomic coordinates of alpha carbons and use them to construct backbone traces.
Week 5 June 15 - June 21	● Select the optimal smoothing algorithm for smoothening the backbone traces from B spline and Hermite spline and implement it. ● Generate ribbon curves by translating the curve parallel to the helix axis.

Week 6 June 22 - June 29	● Add the other features like beta strands and loops, colors etc.
Week 7 June 30 - July 7	● Create Unit Tests for the newly added Ribbon representations ● Create demos or examples for the same. ● Perform bug fixes.
Week 8 July 8 - July 15	● Start working on Molecular Surface structures of proteins by reading the research papers. ● Select the appropriate method for Molecular Surface Representation which best suits the needs (optimum speed and rendering ability) as instructed by the mentor. ● Write the code to implement CPK coloring for different atoms which will comprise the protein.
Evaluations July 12 - 16	
Week 9 July 16 - July 23	● Create the Molecular Surface structures by implementing the methods discussed in the research papers which were selected in the previous week i.e. week 8.
Week 10 July 24 - July 31	● Create Unit Tests for the newly added Molecular Surface representations. ● Create a demo or example for the same. ● Perform bug fixes. ● Work on stretch goals if time permits by implementing stick, ball and stick and wire representations (create tests and demos for the same)
Week 11 Aug 1 - Aug 8	● Work on stretch goals by implementing Pipes and Planks and Sphere representations of protein structures. ● Add unit tests and demos for the same. ● Try making some physics simulations if time permits.
Week 12	● Debug the code for all the developed protein

Aug 9 - Aug 16	structures. ● Prepare demonstration videos/tutorials for the newly implemented features. ● Prepare the final demo for all new features.
Final Evaluations Aug 16 - 23	

Commitments and Availability

1. College End Term Examinations from 19 May to 29 May.
2. I'm doing a research internship on the side but it won't affect my time investment for GSoC as I'm doing the research internship now too while managing online classes, assignments and quizzes (which add up to at least 40 - 45 hrs a week).
3. I have no exams in the coding period (from June 6 to Aug 23). I don't have classes either for the most part of the coding period as I have my summer vacations from May 30 to Aug 2, therefore I'll be able to invest at least 40- 45 hrs a week in the GSoC project as I won't have any online classes nor will I have to study for any exam/assignment.
4. I have no plans of any personal vacations nor of any travel otherwise.
5. Typical working hours (in EST): 11:30 pm to 2:30 am, 4:30 am to 7:30 am, 11:30 am to 2:30 pm.
6. I'm willing to reschedule and re-plan the timings as per mentor's/core team's availability/requirements.
7. I'm only applying to FURY for GSoC 2021.