Alexander Chang & Yogindra Raghav

BIOSC 1640

# ProteinVR/Yogalex & Chemoinformatics Machine Learning

Faculty Advisor: Dr. Jacob Durrant

## **Part 1: ProteinVR/Yogalex**

The current status of ProteinVR/Yogalex is currently on hold, however, it is ready to be seen by students and used in a classroom setting. As an overall experience as a group we faced a few challenges on the way. First was the resources that we have available to us. Our software for developing 3D models, Blender, requires tons of graphic intensive processing power because of the details of the models. However, the lab only had one available desktop that would be able to handle the processing power of all the different vertices that are models have. This caused working on the project to be relatively slow because both of us had to share the computer and work around each other's time schedules in order to work on the project. Although we had issues with working around the one computer, we were still able to create a presentable product. The next issue that we faced was the learning curve of Blender Python and Blender itself. Blender Python is not exactly the same a the standard Python language that many have experience using. Blender Python has specific functions that are used frequently for model positioning and animation. Therefore learning the new syntax and functions took some time before making headway into creation of certain plugins that we both created. Blender is an open-source software, this being the reason why we chose it. However, the user-experience of the software was not friendly. Many of the key bindings were not as expected, so learning to overcome habitual muscle memory for certain keys was not as easy, making the experience sometimes frustrating.  The biggest issue that we faced was render time.  Rendering models took hours, alas there was downtime waiting for certain models to be rendered before we could move on to the next one.

The ambiguity that we dealt with during the project was that we had more independent work.  Dr. Durrant gave us more freedom than was expected, so the procedure of creating models was completely done on our own.  This meant for us that we had to do more independent study on learning Blender and Blender Python and learning the tips and tricks that would help save time.  As a resource we found YouTube to be one of the most valuable places for learning Blender as a whole because the YouTube Blender community is strong.  In terms of Blender Python, we were mostly on our own.  We had available a Blender Python API, Blender Stack Exchange and YouTube, yet resources were extremely limited.  Blender Python is a language not used as often so forum posts about it were rare.  When we had a question about certain features of Blender Python, they were not so readily available as the construction of the 3D models.

## Alex's Experience

When making the cell models, I was hoping to find some template of the 3D models on the internet and then modify them for the project.  However, that was not the case.  I continuously browsed around the internet for a couple of days before finally giving up and deciding that I would have to make the models from scratch.

It was hard to decide where to proceed in terms of creating the models because this was the first time I worked with Blender.  After tinkering with Blender I found a methodology that would be extremely useful to me.  There is an option when using Blender to put a background image while you are manipulating objects.  I used this feature to superimpose an image that represented a cell or part of a cell. Then by using a brush tool I was able to stretch 3D spheres to the shape that overlays the background image. After some trial and error, I found that there was ways while molding the shape of the object that I could do it on both sides by adding a symmetry modifier. Once I found this method I stuck with it for the rest of the creation of the models. Before, I tried creating the object free hand, and it was extremely frustrating and time consuming. There were some other methods that were shown on YouTube that showed how to create complex objects. However, it had a steep learning curve to it that would take a ton of time to learn. I decided to opt out of those other methods.

Next I had to help create the ProteinVR plugin that is supposed to help users decide what objects need to be rendered in higher quality. This plugin was suppose to help give student-driven customization in terms of what they want to learn. Dr. Durrant wanted a 3 option customization where there was one category that help high quality rendered objects, another that held lower quality objects, and then the background image by itself.

There was already code that was created for the creation of this plugin, however it did not fit the paradigm that we wanted to have. Essentially, the plugin need a complete overhaul in order for it to be useful for us. Changing the framework of the plugin was difficult because the code was not documented well. Therefore, going through the Blender Python code took time because I was unfamiliar with the syntax and framework of the code. It was frustrating to be sitting in front of the computer and stare at code that did not make sense to me. This part of the project ended up in me doing a ton of research on the side in order to understand the code.

As an overall experience, it was really enjoyable to see some of the molecules that I created come to life in Virtual Reality. Even though there were many frustrations while doing the project, I would love to see where the rest of this project ends up in the future.

**Yogi’s Experience**

Alas, even making realistic looking models took a very long time and required learning many new skills. To create the human anatomical models, we employed a long but effective plan of action. Creating correct shapes of the objects and modeling every organ system in the body would have been extremely tedious. Instead, we were able to obtain free .obj files from online. Obj files are basically 3D model files that include many data points (vertices and edges) that give a definite shape to a “basic” material. The important thing to remember from all of this is that these files have no color, come with improper sizing, and are practically impossible to work with.

The ridiculous number of data points each file contains means that within the 3D editing engine of Blender, it is very hard to navigate and work with the objects. Heavy lags of multiple seconds were found when trying to refocus objects, move around/view the object from different angles, and even zooming in on an object. To be able to work with these objects, we decided we needed to decimate all of them. Decimation is a process similar to lossy compression which aims to keep the overall general outline of a file (in our case object) while getting rid of as much extraneous data as possible (in our case, many extra vertices and edges). These extra edges and vertices give extremely sharp and definite boundaries to our object. Yet even after decimation, for the most part, there are few detectable differences from our original file and the “compressed” version.

To do the decimation, we wrote a small script using Blender Python to iteratively go through every object and compress them into half the original space they took up. This process had to be done on multiple layers of Blender. You may be wondering what I mean by layers?

Here’s a quick explanation. Within the Blender 3D editing engine you can separate your 3D files into different “scenes” or layers so that not all objects loaded in are displayed at once. We decided to use each layer to represent a different organ system (eg. nervous, arteries, veins, bones, etc.). This decimation process must be repeated on every single layer. Blender Python is not that intuitive and very buggy causing this simple script to take weeks to get correct. The unbelievable amount of online forums I scoured for help during this process caused me to nearly choose another approach to the problem. Thankfully, we fixed the script and it did it’s job. We could move on to the next part: realistic rendering, lighting and cameras.

Once the files were usable within the editing view of Blender, then there was more work to do. The next step was to create equirectangular cameras (allows Virtual Reality view) and position them in the right places on the scene. After watching many YouTube tutorials and trial/error, this section of the project was finished. Time to move onto lighting.

At this point, I had to learn less about Blender and biology but instead more about photography. I learned about three-point photography, a modern and effective lighting technique so that scenes are illuminated from all angles. I learned how to add and use lights in Blender scenes while implementing three-point lighting. Once this was done, it was time to move onto materials.

We aimed to make each of the organ systems as realistic as possible. This meant that I had to look at multiple live-surgery images/videos to determine the color and how the body parts interact with light. We care about how it interacts with light so we can model it properly under our well-lit conditions. To realistically model objects, we need to use the node editor. Before I explain what the node editor, let me motivate it.

A glass surface is one that absorbs and reflects light at a very high ratio whereas an extremely matte object absorbs light without reflecting much light back to the source. Most things in our world are made of materials that fall somewhere between those two above example objects, in terms of reflectivity. The actual organ systems, under well-lit conditions, are made up of materials that are only slightly reflective. This is because they are made up of different molecules (let’s not get into the organic chemistry of it) that exhibit reflectivity and others that don’t. So how can we reflect this mixture in a virtual object? This is the purpose of the node editor.

The node editor allows a user to mix many different types of materials. I tried multiple colors and mixed a glass material with a matte material to get a semi-reflective and accurate representation. Once this was done on a chosen object, I copied these settings to all objects in a scene. This had to be done for every single scene. An important note is that “scene” and “layer” are the same and can be used interchangeably. Now time to go to backgrounds.

The final issue was backgrounds. I created a huge plane underneath the objects represented on each scene and kept messing with background colors that would accentuate the materials of the objects on the current scene. Dark blue ended up being a good background for all the scenes and so I made a plane which I copied onto each of the scenes.

In between all this there were a few more scripts, lots of frustration, many YouTube videos, and forum searching to the max.

## <u>**Part 2: Machine Learning/ Chemoinformatics**</u>

When we transferred to this project we had to learn a new branch of bioinformatics! It definitely took us a couple days to fully understand the subject of machine learning. We needed this to understand the open-source Chemoinformatics software NNScore 1.0 and 2.0. The purpose of this software is to use a neural network to better predict whether or not a given small molecule would bind a receptor protein.

However, once we did some basic learning, we could finally read the papers that accompanied NNScore 1.0 and 2.0. Our job was to extract machine learning predictors that describe the interactions between a ligand and a receptor within NNScore 1.0 and 2.0 and integrate the features into a new machine learning algorithm based on a multi-variate regression model. We read the papers which explained how machine learning works to categorize ligand and receptor interactions. The biggest problem was when we started looking at the code to find the sections that describe features we were looking for.

Once we downloaded the code, we found one glaring issue. The code was not documented. If there was any documentation it was sparse and not useful for us to understand what is happening within the code. There were thousands of lines of code, and we had no idea where to start looking for these features. The only option left was to go line by line and connect all the functions that called each other. This part of the project was specifically time consuming because trying to read someone else's code without documentation takes a long time to understand. From this experience, we learned a lesson for the future. Always document your code so other people can understand what is happening!

To make sure that we targeted the right pieces of code for extraction, we had code reviews with Dr. Durrant's lab manager, Patrick Ropp. Having another person to discuss what is happening in the code was helpful for us. This process helped us understand NNScore's organization. There were Python dictionaries hard-coded in NNScore where there were symbols that represented larger molecules. Once we were able to identify the features that we are suppose to extract, our next step was to develop pseudocode for optimization and separation into another function.  Once we developed the pseudocode and ran it by Dr. Durrant's lab manager, we were finally able to start coding the new function.  This specific experience was a good lesson for us.  We realized that it is always helpful to have another person look at what you are doing to double check that your logic or code makes sense.

When we developed the pseudocode for the extraction of all the features, we expected the coding to be easier.  However, the coding was a little more difficult than anticipated.  We wanted the code to be able to be compatible with the AutoDock and Protein Data Bank (PDB) files.  So we had to learn the structure of PDB files so that we can differentiate between a ligand atom versus a receptor atom that is listed in the files.

The current state of the project is still going through the programming process.  We believe that we could have made some more progress if we did not have to take so long to understand the code.  All in all, it has been an enriching and productive semester here in the Durrant lab.