ENGR 103 – Spring 2011

Freshman Engineering Design Lab

"BioGUI: Protein Circle Plot Visualization"

Final Report

Section 19, Group 275, Project 126	**Date: June 1, 2011**

Submitted to:	*Fred Allen*
	David Jamison
	Ahmet Sacan
Group Members:	*Brian Tighe*
	Ethan Sena
	Henry Kuns
	Gavin Rapp

Abstract

This goal of this project was to create a circle plot based on different protein data files. In order to accomplish this, two primary tasks were established. The first task was to read in the data from a file and the second was to draw the plot using this data. Once the plot had been created, the program was then modified to incorporate a graphical user interface. This interface allows the user to select the file to be plotted.

1. Overview

In the field of bioinformatics, there is no single unifying graphical user interface system. This project is aimed towards developing small program sections that can be used to display plots based on information from data files. Other goals of this project include gaining group programming experience and developing programs to solve a realistic task.

2. What is a PDB?

A Protein Data Bank, or PDB, is a tabulated text file that features structural data of biological molecules, such as proteins. The PDB files can contain up to thousands of lines of data that describe quite precisely the properties of a particular protein.

3. Roadmap

3.1 Initial Planning and Design

The first task that this project entailed was deciding on a programing language to develop the program in. After debate, the group selected C++ over Java because of the robust nature of the language, in addition to its ability to seamlessly work with various libraries. Following that choice, the task of researching graphics libraries was tackled. The group decided that the Cairo Graphics Library would be the best choice, as it was the most mature and documented library of the ones available. Because only simple, two dimensional shapes needed to be shown in PNG files, many of the additional features of a more advanced graphics library would have been unnecessary. The design and functionality of the GUI was planned as a rough target for the group to meet (Figure 1). In order to share updates to code and project files with one another, each group member installed TortoiseSVN on their machine, which saved files in a central repository that was tracked via Subversion (SVN).

In order to make the 10 week deadline, the group divided the work between two groups: the parse PDB group and the drawcircle group, each with two members.

3.2 Overview of Technologies Used

3.2.1. Cairo Graphics

Cairo is a 2D graphics library that works with a number of programming languages, C++ included. While researching how to develop a GUI, the group discovered that GTK+ relied on Cairo for a variety of things and was included in the release.

3.2.2. GTK+

GTK+ is a widget toolkit that eases the process for developers when creating graphical user interfaces. Since it features support for C++ and Windows, it was the ideal choice to develop a GUI in.

3.2.3. TortoiseSVN

In order to share changes made to the project source code, documentation, and files, the group used a server running Subversion. In order to access and synchronize the files between the client and repository, the group members installed TortoiseSVN, a program that allowed users to perform changes and "commit" them to the repository for all to see.

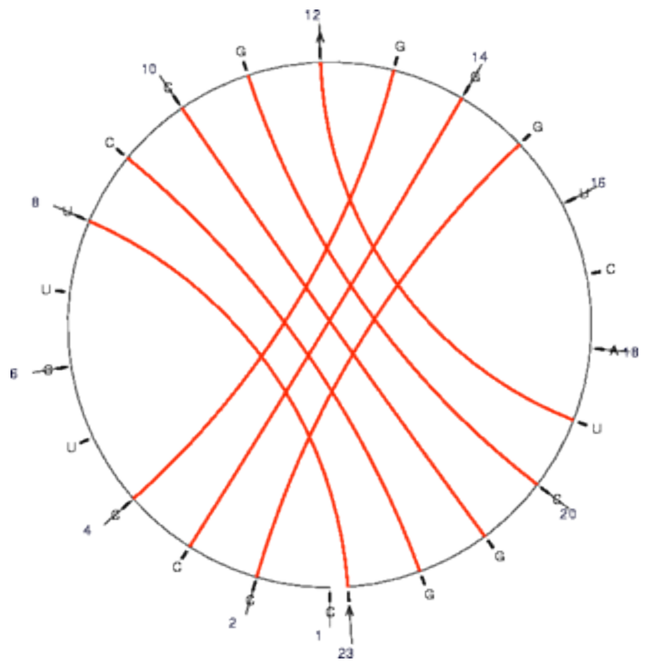


Figure 1, A mockup for drawcircle

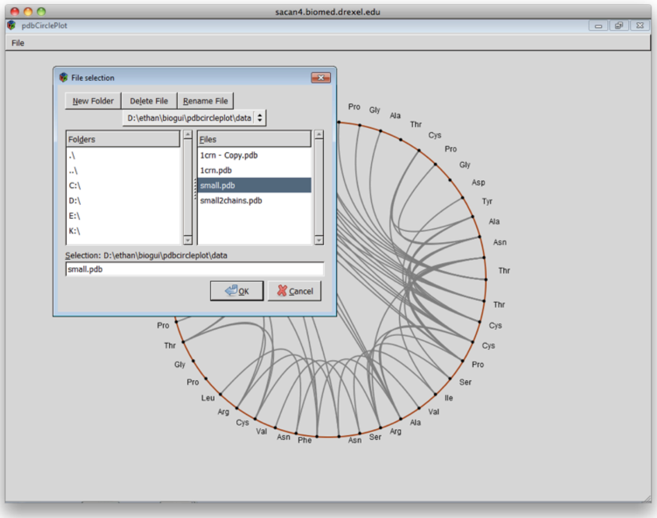
3.3 Parsing PDB Files

One group had the task of extracting the pertinent information from the database files. Protein Data Bank files (PDB) contain tabulated information about the atoms within a protein structure. The essential data were the coordinates of the different atoms and the type of atom. These were necessary in order to plot the connections between the alpha carbon atoms. A connection was drawn between two alpha carbon atoms if the distance between them was within a certain tolerance, so the distances between atoms also needed to be found. In addition to coordinates, alpha carbons have residue names, which are used in the circle plot.

Along with parsing the PDB files to be used with the circle plot, the parsing group also created a function to obtain coil values. These values would be used in other parts of the BioGUI unrelated to the circle plot. The algorithm for finding these coils is based on the positions of CA atoms and its neighbors. The algorithm used was T M-Align from Zhang and Skolnik. [4]

3.4 GUI Design and Implementation

The GUI was designed first as a mock up in PowerPoint. The specifications were then taken and implemented into GTK+. Originally the program Glade was going to be used for the GUI design, but it was not correctly working in the Windows environment. Instead, pure code was used to create a functioning GUI. This was not as simple but allowed for more precise control and a learning experience. This GTK+ format allows for ease of use from the user and the ability to easily add more features such as settings for threshold (a feature that is already implemented outside the GUI) and different ways to draw the circle plot.



D:\ethan\biogui\pdbcircleplot\data

Folders	Files
.\	1crn - Copy.pdb
..\	1crn.pdb
C:\	small.pdb
D:\	small2chains.pdb
E:\	
K:\	

Selection: D:\ethan\biogui\pdbcircleplot\data

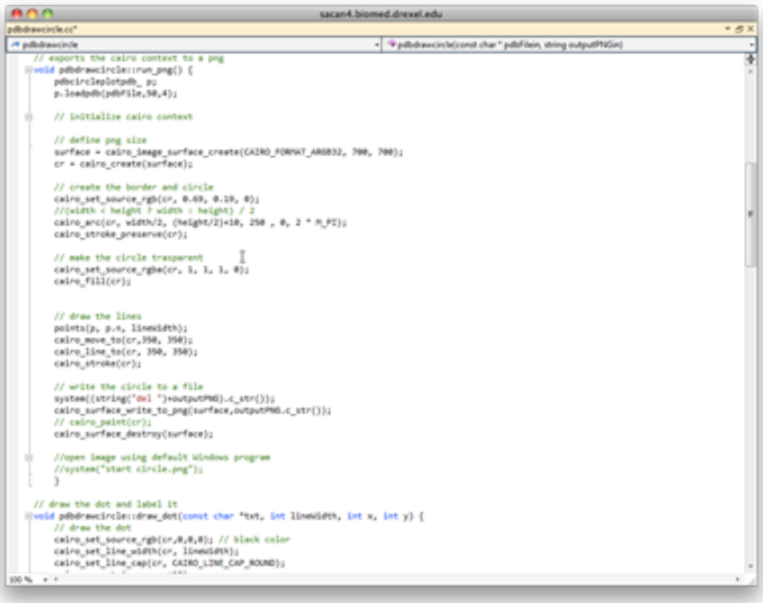
small.pdb

Figure 2, BioGUI on Load PDB screen.

The current GUI design includes a large window for image viewing and a single file menu. The file menu allows the user to choose to open a PDB file for viewing, while saving the image as a PNG file to the default directory. The user chooses a PDB file to load from a browse menu (Figure 2). Currently the file has to be manually copied and saved elsewhere for each new PDB, due to the restriction on time and attempts to keep the code simple.

3.5 Circle Graphic Programming

Perhaps the most important part of this project was to display the information retrieved from the PDB files in an easily interpreted visual format. In this case, the objective was to display the protein and the different atoms along the protein in the form of a circle. Each atom would be labeled and represented by a point along the edge of the circle (Figure 1). Then, given a minimum distance, the program would find the atoms that were farther away than the threshold, and draw a line between them.



```
 // exports the cairo context to a png
void pdbdrawcircle::run_png() {
    pdbcircleplotpdb_ p;
    p.loadpdb(pdbFile,50,4);

    // initialize cairo context

    // define png size
    surface = cairo_image_surface_create(CAIRO_FORMAT_ARGB32, 700, 700);
    cr = cairo_create(surface);

    // create the border and circle
    cairo_set_source_rgb(cr, 0.69, 0.19, 0);
    //(width < height ? width : height) / 2
    cairo_arc(cr, width/2, (height/2)+10, 250 , 0, 2 * M_PI);
    cairo_stroke_preserve(cr);

    // make the circle trasparent
    cairo_set_source_rgba(cr, 1, 1, 1, 0);
    cairo_fill(cr);


    // draw the lines
    points(p, p.n, linewidth);
    cairo_move_to(cr,350, 350);
    cairo_line_to(cr, 350, 350);
    cairo_stroke(cr);

    // write the circle to a file
    system((string("del ")+outputPNG).c_str());
    cairo_surface_write_to_png(surface,outputPNG.c_str());
    // cairo_paint(cr);
    cairo_surface_destroy(surface);

    //open image using default Windows program
    //system("start circle.png");
    }

 // draw the dot and label it
void pdbdrawcircle::draw_dot(const char *txt, int linewidth, int x, int y) {
    // draw the dot
    cairo_set_source_rgb(cr,0,0,0); // black color
    cairo_set_line_width(cr, linewidth);
    cairo_set_line_cap(cr, CAIRO_LINE_CAP_ROUND);
```

Figure 3, The pdbdrawcircle class.

In order to do this, it was necessary to split the job into a series of predefined steps. First, the circle would be drawn using the Cairo libraries and the C++ programming language (Figure 3). After that, given a number of points, the points would be placed evenly spaced around the edges of the circle. The next step was to retrieve the number of atoms, and instead print the circle with that many points around the edge. Using the functions created for parsing the data file, each point was then labeled with the appropriate atom name. Then connections were drawn between the atoms that were appropriately far apart. At first these lines were linear, but once the connections between them were being made correctly, these were changed to be curved, and therefore more aesthetically pleasing and easier to see.

4. Difficulties Faced

Throughout the 10 weeks, the group faced a number of difficulties with the project. The first problem faced was including Cairo and getting it to work on every group member's machine. The main problem was that the group failed to properly include the libraries in the Microsoft Visual Studio project, thereby causing Cairo to produce linker errors. The next problem was with GTK+, as the team assigned to GUI development attempted to use Glade, a GTK+ "WYSIWYG GUI maker" of sorts, in order to generate the code for the GUI. Unfortunately, despite countless attempts, Glade was found to not be compatible with Windows, and therefore not helpful for the project. Finally, the last major problem was combining the work of the drawcircle and gtk classes. As the members developed the C++ classes separately, they needed to take time to adapt their code to properly work with one another.

5. Results

After 10 weeks, the group delivered a finalized program that met all the requirements and goals set forth in the first week. The program brings up a program that features a GTK+ GUI allowing the user to select a PDB file. Once selected, the program will parse the PDB, generate a PNG featuring the bonds between the atoms, and display it in the window (Figure 4). Additionally, the program works with a web interface that allows the user to type the name of a PDB file, which calls on the program to generate an image for that data.

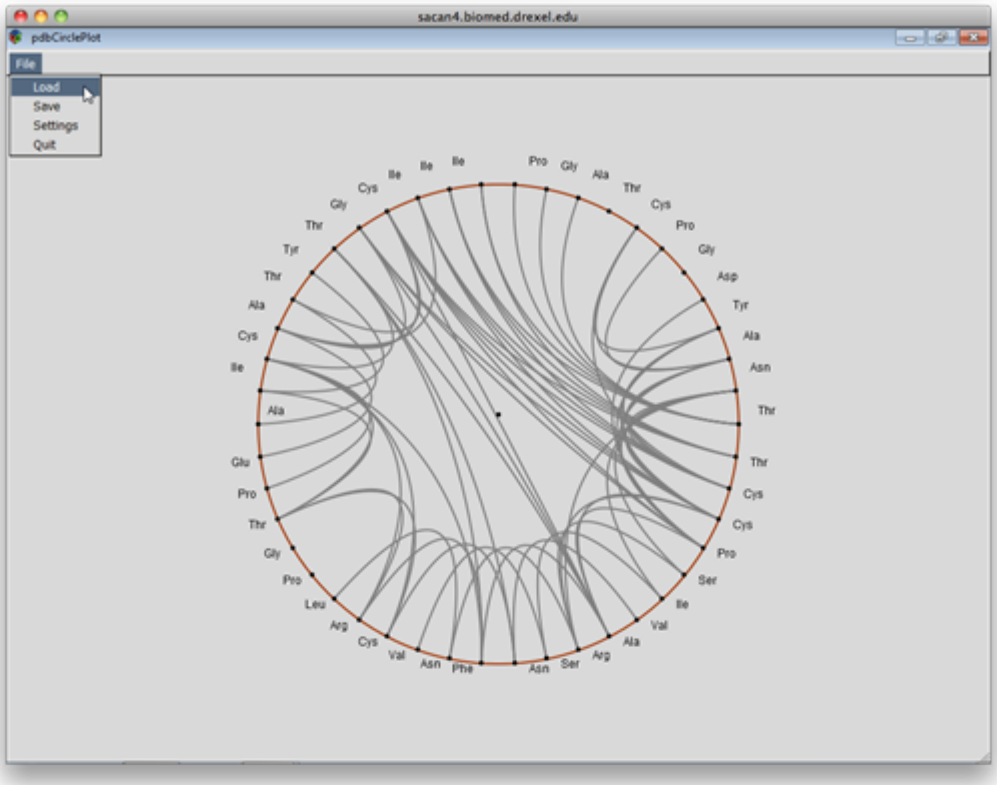


Figure 4, BioGUI displaying a parsed PDB file.

6. Other Options for Protein Visualization

In addition to the group's solution, there are a number of other programs that exist to accomplish a similar task. Examples of such programs are Geneious [1], a general bioinformatics analysis tool, pymol/jmol [2][3], 3D protein visualization tools, and other web-based solutions.

These solutions are not as ideal as the one produced by the group because the visualization is not as practical due to the excessive visuals featured in the programs. BioGUI offers a simple yet informative image showing protein bonds of various atoms. This allows the user to quickly see important information regarding these bonds without having to guess with a messy 3D image.

7. Timeline

The timeline, as seen in table 1, for the project has remained for the most part unchanged. Weeks seven and eight have been modified slightly to include for the GUI design.

Table 1: Freshman Design Project Timeline

	Week									
Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Circle Plot Program Design	x	x								
Parsing PDB File		x	x							
Connecting with Splines			x	x						
Secondary Structure Visualization				x	x					
GUI Design and Implementation						x	x	x	x	
Final Report Preparations										x

8. References

[1] Drummond AJ, Ashton B, Buxton S, Cheung M, Cooper A, Duran C, Field M, Heled J, Kearse M, Markowitz S, Moir R, Stones-Havas S, Sturrock S, Thierer T, Wilson A (2010) Geneious v5.3, http://www.geneious.com

[2] The PyMOL Molecular Graphics System, Version 1.3, Schrödinger, LLC.

[3] The JyMOL Molecular Graphics System, Version 1.0, Schrödinger, LLC.

[4] Yang Zhang and Jeffrey Skolnick, "TM-align: a protein structure alignment algorithm based on the TM-score" Center of Excellence in Bioinformatics, University at Buffalo. Apr. 22, 2005.