Computer Systems Lab 2007-2008: Research Paper Accurate 3D-Modeling of User Inputted Molecules Using a Hill Climbing Algorithm

Ben Parr

January 24, 2008

Abstract

In order to better understand chemistry, chemists create 3D models of molecules. In a large introductory chemistry class, physical models are not viable because the supplies needed to give each student the opportunity to create even simple molecules are too costly. Also, software available online can also be costly. The goal of my research project is to create a program that will allow users to generate accurate 3D models of simple molecules (i.e. not macromolecules). Therefore, my project could help students in introductory chemistry courses better understand the geometry of different molecules.

1 Introduction

The goal of my project is to create a free program that will allow users, such as introductory chemistry students, to easily and intuitively create models of molecules. While running the program, users will be able to create atoms and bonds where they want them. Once they have everything created, with just a click of the mouse, my program will position the atoms correctly. The user will then be able to export the model so that it can be easily imported at a later time. All of the models will also be rotatable and zoomable at all times, allowing for a better understanding of the molecular geometry.

The first step of my project was creating the graphics for my program. The second step was creating an intuitive user interface that would allow users to easily create molecules. Both the first and second step of my program were done using OpenGL. The third step of my program is to create the algorithm that will accurately position the atoms. In order to do this, I will use a hill-climbing algorithm.

2 Background

A lot of research has been done on modeling molecules and the techniques to do so now have become quite advanced. Pharmaceutical companies have invested thousands of dollars in programs to predict orientations of new complex molecules. Large databases have been created, storing the orientations of thousands of molecules. Chemists have modeled everything from RNA to inorganic crystals. These programs have become increasingly more accurate over the years. However, the cost of these programs has also increased, and now very few people have access to them. For a beginning chemistry student there are not many options to play around with molecules and learn their different geometries. buying physical modeling sets for everyone in a introductory chemistry cousre is costly. Also, software on the Internet can also be costly, sometimes a couple hundred dollars. Therefore, even today, a free program to model simple molecules would be helpful.

3 Development

The first step to my project was creating the graphics for my models. The graphics for my project are a much simplified version of real molecules: spheres represent atoms and cylinders represent bonds. Nevertheless, the results are sufficient.

The second step of my project was creating an easy and intuitive user interface. Through mainly inputs from the mouse (and some keyboard input) users can create atoms and bonds, select and delete atoms and bonds, import and export models, draw single, double and triple bonds, choose which element they want to draw, and position the atoms where they want. (For atomic radius values, I used empirical values derived by J.C. Slater and published in The Journal of Chemical Physics) With these features, users will

be able to easily create the models.

The third step of my project is creating the algorithm that will correctly orient the atoms in the molecule. Since there are many variables accounting for the actual orientation of the atoms and because I only have a limited amount of time, I will need to limit the number of accounted variables. With my set number of accounted variables, I will use an A.I. algorithm (hill-climbing) to correctly orient the atoms in the molecule. The first variable that I will account for will be the polarity of the bond. Once I get that working, I will continue to add more variables, creating a better model.

Dynamic testing will not work for my project because an atom can not bond with any random atom. Therefore, I will mostly use specific structural and functional testing and path and branch testing.

4 Testing

Testing the graphic part of my program was simple; all I had to do was run it and see if my program created the intended model.

Testing the user interface part of my program was a little more complex. I had to think of everything that a user could do and account for the inputs. However, my program runs as intended.

Now that I have all of the graphics and user interface done for my project, I will now be able to start working on my algorithm to correctly orient the atoms in the molecule. I will first start by only taking one variable into account: the polarity of the bond. Once I get that working, I will then add another. Once that one is working, I will start on another. Through this process, my program will start to produce accurate representations of different molecules.

In order to see if my program is constructing accurate models, I will compare my models with the actual/accepted models. By doing this, I will also be able to do error analyses. In order to thoroughly test my program, I will test many different molecules and compare my results with the actual orientations.

5 Results

The purpose of my project is to create a free program that will allow users to easily and intuitively create models of molecules. Then, after the user finished creating the molecule, the program will correctly position the atoms. My project currently allows users to create molecules. A user can also rotate the model and zoom it in and out. These functions help users get a better picture of the molecule. However, I still need to work on the algorithm to correctly position the atoms. This will be the focus of the third quarter.

References

[1] M.A. Fox and J.K. Whitesell, "Bond Lengths in Organic Compounds", *Organische Chemie*, 1994.

[2] D. Shreiner,M. Woo, and J. Neider,

OpenGL Programming Guide: The Official Guide to Learning OpenGL 6th Edition, Addison-Wesley Professional, 2007.

[3] J.C. Slater, "Empirical Atomic Radius", *J. Chem. Phys. 39*, p. 3199, 1964.

[4] R.C. Weast, M.J. Astle, and W.H. Beyer,

CRC Handbook of Chemistry and Physics, CRC Press, 1984.

[5] "OpenGL API Documentation", Copyright 1997 - 2008, available at `http://www.opengl.org/documentation/`