Avogadro: A Framework and Cross Platform GUI for Building Molecular Structures and the Analysis of Output

Marcus D Hanwell*[1,2] , Donald E Curtis[3], David Lonie[4], Tim Vandermeersch[5], Eva Zurek[4], Geoffrey R Hutchison[1]

[1]Department of Chemistry, University of Pittsburgh, 219 Parkman Avenue, Pittsburgh, PA, 15260, USA
[2]Kitware, Inc., 28 Corporate Drive, Clifton Park, NY, 12065, USA
[3]Department of Computer Science, University of Iowa
[4]Department of Chemistry, State University of New York at Buffalo
[5]Tim's Affiliation, Europe

Email: Marcus D Hanwell*- marcus.hanwell@kitware.com; Geoffrey R Hutchison - geoffh@pitt.edu;

*Corresponding author

Abstract

Avogadro is a free, cross-platform, open source, OpenGL based graphical user interface and library for building molecular structures, formatting input files and analyzing output files from computational chemistry codes. The work presented here details the Avogadro library, which provides a framework, application programming interface, and three-dimensional visualization capabilities that have direct applications in research and education in the fields of chemistry, physics, materials science and biology. The Avogadro application provides a rich graphical interface using dynamically loaded plugins. The application can be extended by implementing a plug-in module in C++ or Python in order to explore different visualization techniques, build/manipulate molecular structures, and interact with other programs. We describe some example extensions, one which uses a genetic algorithm to find stable crystal structures, and one which interfaces with the PackMol program to create packed, solvated structures for molecular dynamics simulations.

1 Introduction

Many areas such as chemistry, materials science, physics and biology need efficient computer programs to both build and visualize molecular structures. The field of molecular graphics is dominated by viewers with little or no editing capabilities, such as RasMol [1], JMol [2], PyMOL [3], VMD [4], QuteMol [5], and

BALLView [6] among many others. The aforementioned viewers are all freely available, and most of them are available under open source licenses and work on the most common operating systems (GNU/Linux, Apple Mac OS X, BSD and Microsoft Windows).

The choice of software capable of building chemical structures is far smaller. There are existing commercial packages, such as Spartan, CAChe, GaussView, Materials Studio [7] and CrystalMaker [8], that are well polished and capable of constructing many different types of molecular structures. They are, however, not available for all operating systems (most of them only have programs for Microsoft Windows), are not easily extensible or released under an open source license. Licensing costs can be prohibitive. If the company were to change its direction or focus this can lead to a loss of significant research investment in a commercial product.

The selection of free, open source, cross platform molecular builders was quite limited when the Avogadro project was founded in late 2006. Ghemical [9] was one of the only projects satisfying these needs at the time. Two of the authors (Hutchison and Curtis) contributed to Ghemical previously, but had found that it was not easily extensible. This led them to found a new project in order to address the issues they had observed in Ghemical and other packages. The Molden [10] application was also available, able to build up small molecules and analyze output from several quantum codes. It suffers from a restrictive license and it uses an antiquated graphical toolkit.

Broad goals for the design of a molecular editor were identified following a case study of the available applications. One of the main issues with both commercial and open source applications is a lack of extensibility, many of the applications also only work on one or two operating systems. An open and extensible framework is needed in order to perform innovative research. The creation of a framework that implements many of the necessary foundations for a molecular builder and visualizer would facilitate more effective research in this area.

At the time of writing it is apparent that other researchers have perceived similar needs. Several new applications are available today that focus on both building and visualizing molecular structure. These include CCP1GUI [11], Gabedit [12] along with some highly specific editors such as MacMolPlt [13]. Whilst offering many interesting and useful features, these projects suffer from the same kind of issues centering around effective reuse of existing code, well commented and documented code and easy extension to specialized areas.

The Avogadro project has endeavored to make a free, open source framework for both building and visualizing molecular structures. Much of the initial focus has been placed on preparing input and

analyzing output from quantum calculations. Other applications such as preparing input for MD simulations and visualizing periodic structures will also be presented, demonstrating the flexibility of the Avogadro framework.

Avogadro has close ties to several other free, cross platform, open source projects in order to reuse as much code as is practical. These projects include Nokia Qt to provide a free, cross platform graphical toolkit, Open Babel [14] for chemical file input/output, geometry optimization and other chemical perception, Eigen [15] for matrix and vector mathematics, OpenGL/GLSL for real time three dimensional rendering and POV-Ray for ray-traced rendering.

Avogadro also uses quantum codes such as GAMESS-US, Q-Chem, Gaussian and MOPAC to perform ab-initio and semi-empirical calculations. Support for more calculation backends is planned and we are working with Q-Chem on an advanced input file generator. As with most open source projects, the directions the project has taken were influenced largely by the research areas and interests of the principal contributors.

2 The Graphical User Interface

The first thing most people will see is the main Avogadro application window. Binary installers are provided under the GPL license for Apple Mac OS X and Microsoft Windows, along with packages for all of the major Linux distributions. This means that Avogadro can be installed quite easily on most operating systems. Easy to follow instructions on how to compile the latest source code are also provided on the main Avogadro web site for the more adventurous, or those using an operating system that is not yet supported. The Qt toolkit from Nokia gives Avogadro a native look and feel on the three supported operating systems—Linux, Mac OS X and Windows. The basic functionality expected in a molecular builder and viewer has been implemented, along with several less common features. The vast majority of this functionality has been written using the API made available to plug-in writers, and is loaded at runtime. It is very easy for new users to install Avogadro and build their first molecules within minutes. Thanks to the Open Babel library Avogadro supports a large portion of the chemical file formats that are in common use. The first thing most people will see is the main Avogadro application window. The basic functionality expected in a molecular builder and viewer has been implemented, along with several less common features. The vast majority of this functionality has been written using the API made available to plugin writers, and is loaded at runtime. It is very easy for new users to install Avogadro and build their first molecules within minutes. Thanks to the Open Babel library Avogadro supports a large portion of the

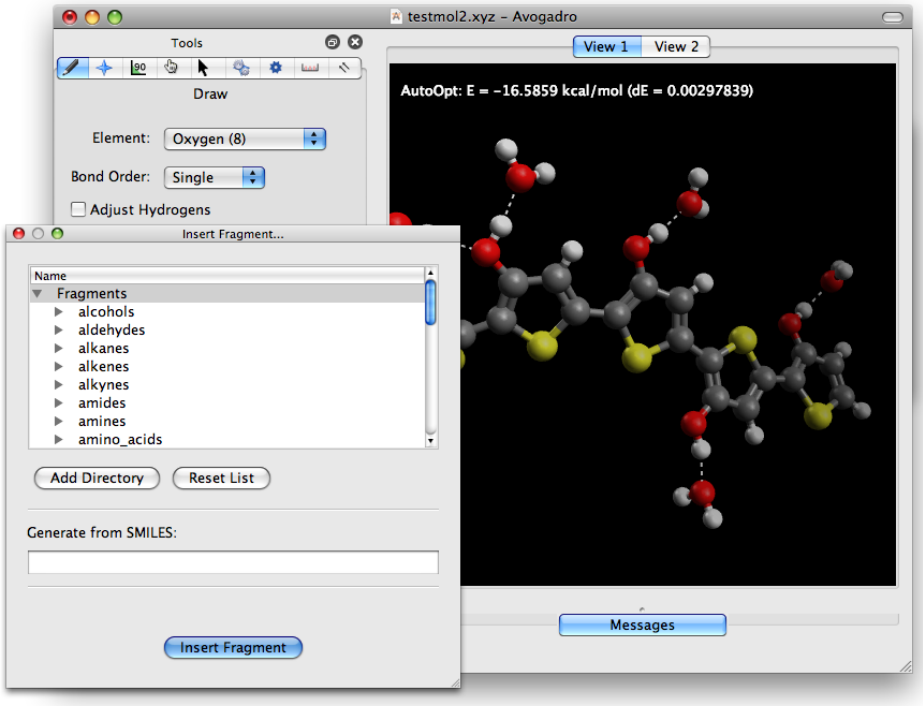

Figure 1: The Avogadro graphical user interface, showing the editing interface for a molecule.

chemical file formats that are in common use.

2.1 Building a Molecule: Atom by Atom

After opening Avogadro a window such as that shown in figure 1 is presented. Simply left clicking on the black part of the display allows the user to draw a carbon atom. If the user pushes the left mouse button down and drags a bonded carbon atom would be drawn between the start point and the final position where the mouse was released.

A large amount of effort has been expended to create an intuitive tool for drawing small molecules. Common elements can be selected from a drop down list, or a periodic table can be displayed to select less common elements. Clicking on an existing atom changes its type, dragging reverts the original atom and draws the new atom bonded to the original. If the bonds are left-clicked then the bond order cycles between single, double and triple.

Right clicking on atoms or bonds deletes them. If the "Adjust Hydrogens" box is checked, the number of hydrogens bonded to each atom is automatically adjusted to satisfy valency. Alternatively, this can also be

done at the end of an editing session by using the add hydrogens extension in the build menu. Keyboard shortcuts are also available to change the active element (by typing the one or two letter symbol), or bond order (by entering the numeric bond order).

In addition to the draw tool there are two tools for adjusting the position of atoms in existing molecules. The atom centric manipulate tool can be used to move an atom or a group of selected atoms. The bond centric manipulate tool can be used to select a bond and then adjust all atoms positions relative to the selected bond in various ways. These three tools allow for a great deal of flexibility in building small molecules interactively on screen.

Once the molecular structure is complete the forcefield extension can be used to perform a geometry optimization. By clicking on "Extensions" and "Optimize Geometry" a fast geometry optimization is performed on the molecule. The forcefield and calculation parameters can be adjusted, but the defaults are adequate for most molecules. This workflow is typical when building up a small molecular structures for use as input to quantum calculations, or publication quality figures.

An alternative is to combine the "Auto Optimization" tool with the drawing tool. This presents a unique way of sculpting the molecule while the geometry is constantly minimized in the background. The geometry optimization is animated, and the effect of changing bond orders, adding new groups or removing groups can be observed interactively.

Several dialogs are provided to provide information on molecule properties, and to precisely change parameters, such as the cartesian coordinates of the atoms in the molecule.

2.2 Building a Molecule: From Fragments

2.3 Preparing Input for Quantum Codes

Several extensions were developed for Avogadro that assist the user in preparing input files for popular quantum codes such as GAMESS-US, Gaussian, Q-Chem and MOPAC. The graphical dialogs present the features required to run basic quantum calculations, some examples are shown in figure 2.

The preview of the input file at the bottom of each dialog is updated as options are changed. This approach helps new users of quantum codes to learn the syntax of input files for different codes, and to quickly generate useful input files as they learn. The input can be edited in the dialog before the file is saved and submitted to the quantum code. The MOPAC extension can also run the MOPAC program directly, if it is available on the user's computer, Avogadro can also load the output file once the calculation is complete.

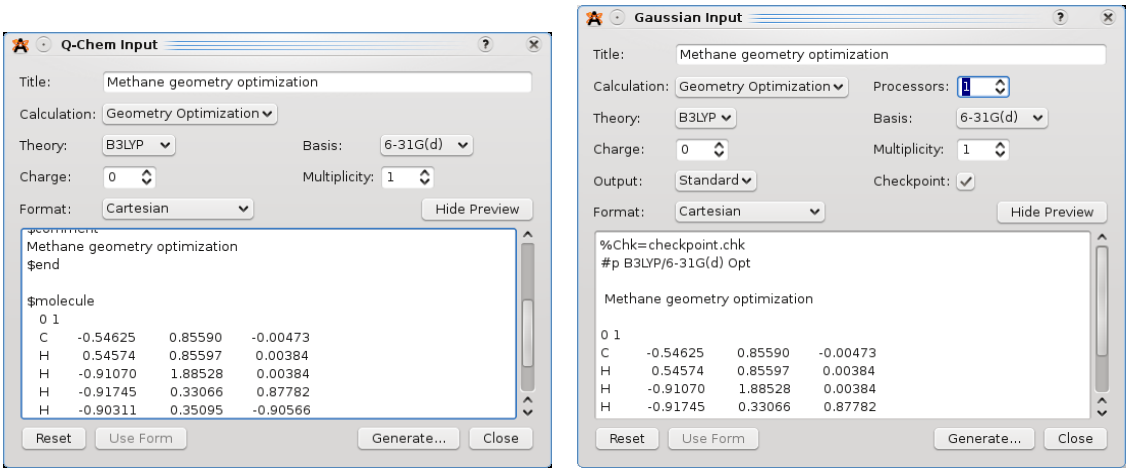

Figure 2: Dialog for generating input for Q-Chem (left) and Gaussian (right). Note that both dialogs are similar in interface, allowing users to use multiple computational chemistry packages.

2.4 Alignment and Measurements

One of the specialized tools included in the standard Avogadro distribution is the alignment tool. This mouse tool facilitates the alignment of a molecular structure with the coordinate origin if one atom is selected, and along the specified axis if two atoms are selected. The alignment tool can be combined with the measure, select and manipulate tools in order to create inputs for quantum codes where the position and orientation of the molecule is important. One example of this is calculations where an external electric field is applied to the molecule. In these types of calculations the alignment of the molecule can have a large effect.

More complex alignment tools for specific tasks couuld be created. The alignment tool was created in just a few hours for a specific research project performing calculation on the piezoelectric effect in single molecules. This is a prime example where extensibility was very important in order to be able to perform research using a graphical computational chemistry tool. It would not have been worth the investment to create a new application just to align molecular structures to an axis, but creating a plugin for an extensible project was not unreasonable.

3 Visualization

The Avogadro application uses OpenGL to render molecular representations to the screen interactively. OpenGL offers a good cross platform API for rendering three dimensional images using hardware accelerated graphics. OpenGL 1.1 and below is used in most of the rendering code, and so Avogadro can

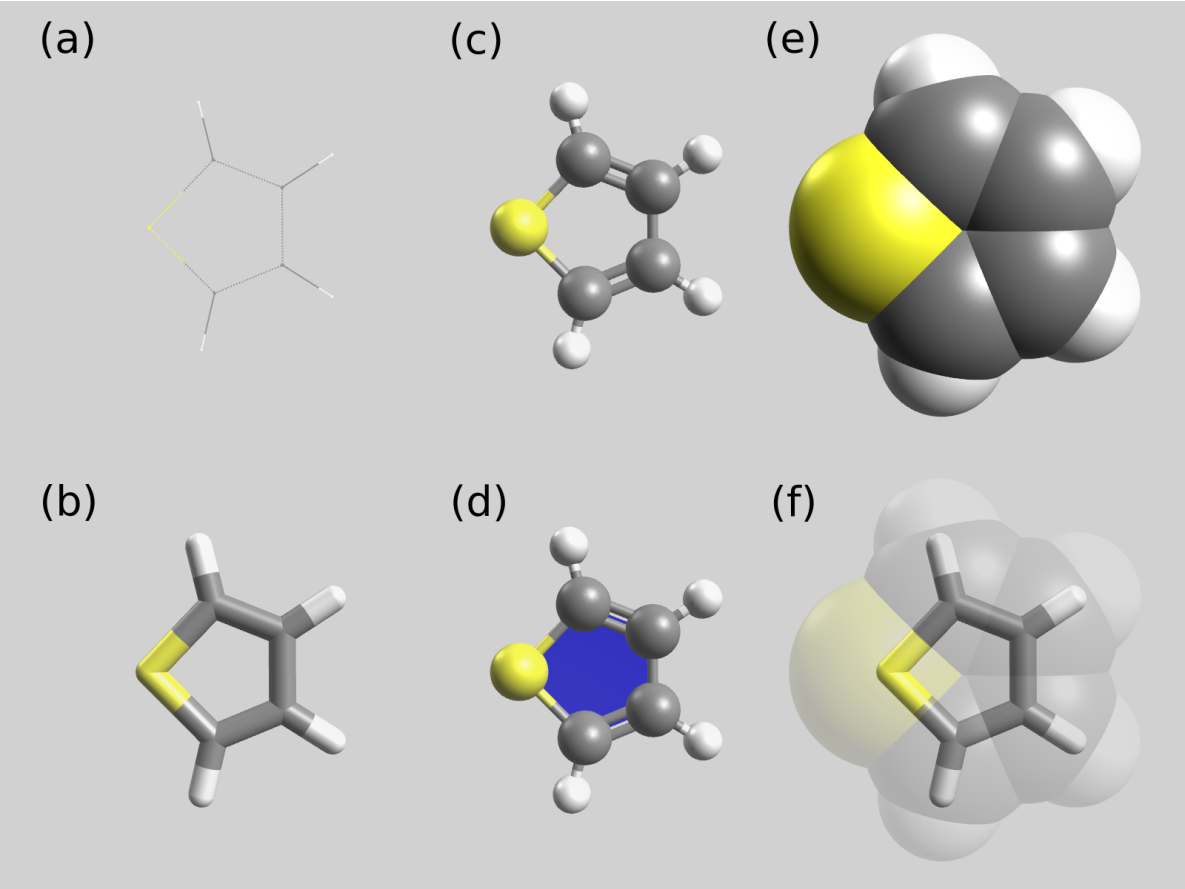

Figure 3: Several molecular representations of thiophene, (a) wireframe, (b) stick/liquorice, (c) ball and stick, (d) ball and stick with ring, (e) Van der Waal's/CPK and (f) transparent Van der Waal's with stick.

be used even on very modest/old computer systems. It is capable of taking advantage of some of the newer features available in OpenGL 2.0, but this has been kept as an optional extra feature when working on novel visualizations of molecular structure.

3.1 Standard Representations

In chemistry there are several standard representations of molecular structre, originally based upon those possible with physical models. The Avogadro application implements each of these representations shown in figure 3 as a plugin. These range from the simple wireframe representation, stick/liquorice, ball and stick and Van der Waal's.

It is also possible to combine several representations, such as ball and stick with ring rendering, figure 3 (d), and a semi-transparent Van der Waal's space-filling representation with a stick representation to

eludicate molecular backbone, figure 3 (f).

3.2 Electronic Structure

The support for electronic structure visualization evolved in Avogadro. Initially support for reading in Gaussian cube files was added to Open Babel, and a marching cubes implementation was added to Avogadro to visualize isosurfaces. Once this was integrated it became clear that many codes could not easily produce cube files, and that direct support for calculating cubes in the Avogadro application would be advantageoous.

Initially support for Gaussian type orbitals was added, as used in a large range of quantum codes. Later support for Slater type orbitals was added in order to visulize the output from the MOPAC code. This code was written using the Qt Concurrent framework, reading in the basis set, eigenvectors and density matrix in order to directly calculate molecular orbitals and electron densities. The cube calculation is highly multithreaded, parallelising the calculation of each point in the regular grid making up the volume of interest. The original marching cubes implemeentation is then used to calculate isosurfaces in much the same way as for Gaussian cube files.

Both features were developed within plugins, and matured there. More recently the code responsible for reading quantum output files and calculating the cubes has been separated out into a separate library called "OpenQube". This allows other applications to reuse the code more easily, as well as facilitating its use on clusters where graphical rendering may not be available. A range of output files are supported including Gaussian formatted checkpoint files, Q-Chem formatted checkpoints, GAMESS, Molpro and MOPAC AUX files. There are several related projects to add semantic meaning to this type of output, including the JUMBO converter project and Quixote.

3.3 Secondary Biological Structure

Avogadro uses the PDB reader from Open Babel, and is able to to visualize the secondary structure annotated in those files. Initially a simple plugin was developed that drew a simple tube between the biomolecule backbone, then a second more advanced visualization plugin was developed to calculate meshes for the alpha helices and beta sheets. The simple plugin is much faster, but with more optimization it is clear that the superior rendering produces the results expected in that field.

3.4 GLSL, Novel Visualization

GLSL, or OpenGL Shader Language, is a C-like syntax that can be used to develop code that will run on graphics cards. It has been used to great effect by the games industry, as well as in many areas of data visualization. Several recent papers highlight the potential in chemistry, such as QuteMol [5] in adding support for features such as ambient occlusion to add depth to images.

Avogadro has support for vertex and fragment shader programs, and several examples are bundled with Avogadro. If the user's graphics card is capable, these programs can be loaded at runtime and used to great effect to visualize structure. Some of these include summarization techniques such as isosurface rendering where only the edges orthogonal to the view plane are visible giving a much better rendering of both the molecular and electronic structure.

3.5 Ray Tracing

Avogadro uses a painter abstraction that makes it much easier for developers to add new display types. It also abstracts away the renderer, making it possible to add support for alternative backends. Currently only OpenGL and POV-Ray are supported. Due to the abstraction we are able to use the implicit surfaces available in ray tracers to render molecular structure at very high levels of clarity with none of the triangle artifacts present in standard OpenGL rendered images. Much higher quality transparency and reflection also allow for the images to be used in poster and oral presentations as well as research articles.

This feature is implemented in an extension, with an additional painter class deriving from the base class and a dialog allowing the user to edit the basic rendering controls. The POV-Ray input file can also be retained and edited to produce more complex images, or to allow for much finer control of the rendering process if desired.

4 Software Architecture

One area that seems to suffer in many code bases in chemistry is software architecture. This can lead to less maintainable code, poor code reuse and a much higher barrier to entry. Problems were identified in other projects with a view to minimizing their impact when developing Avogadro. Modern software design processes were used in the initial planning stages of Avogadro, along with the choice of modern programming languages and libraries.

Based on the previous experience of the authors, and a review of available programs at the time, several fundamental choices were made. The C++ programming language was chosen, with Qt as the cross

platform graphical toolkit, OpenGL for 3D rendering, CMake as the build system and Open Babel as the chemical library. Using this combination of languages and libraries allowed for the project to be licensed under the GNU GPLv2 license and made (and kept) openly available to all.

The choice of license is an interesting one, and is hotly debated in many industries. The choice of the GPLv2 was necessitated because of the use of the Qt and Open Babel libraries. Qt has since been released under the much more permissive LGPL license. Packages such as PyMol use the more permissive BSD license, which allows the sale of commercial versions in addition to the open source version (some capabilities are present only in the commercial version).

The core of Avogadro is written in portable C++ code with platform specific differences abstracted away by Qt, OpenGL and Open Babel. The CMake build system makes the build process relatively simple on all supported platforms. Avogadro has been successfully built and tested on x86 and x86_64 versions of Linux, PPC and x86 version of Apple Mac OS X and x86 Microsoft Windows.

Avogadro has a well defined set of graphical and programming interfaces. Almost all functionality is implemented in self contained plugins that are loaded at runtime. The majority of these plugins are written in C++, but the Avogadro API has also been exposed to the Python scripting language. This allows for a great deal of choice in how plugins are implemented. In both cases each plugin is a self contained class that implements a set of functions that are part of the Avogadro API, allowing for a wide variety of features to be implemented in a very modular way.

The Avogadro framework uses the model, view, controller paradigm. The model being the core data classes such as Molecule, Atom and Bond, the view being the engine plugins and the controllers being the tools (interactive mouse) and extensions (non-interactive, form based/menu based). Each plugin has full access to the core data model.

4.1 Plug-in Interface

The Avogadro library was developed as an extensible library using C++ plugins that are loaded at runtime for most functionality. The Avogadro plugins are divided into four separate types, all having a common base class. The Plugin class is the common base, defining a minimal set of interfaces for an Avogadro plugin inside the Avogadro C++ namespace.

There are four classes that derive from this common base class specializing their interface for specific activities. The Avogadro::Color base class defines the virtual interface for applying colors to atoms, bonds and other properties. Avogadro::Engine defines the common interface for all display types in Avogadro,

from the simple ball and stick or Van der Waals visualizations through to surfaces and force visualizations. The Avogadro::Tool base class provides the interface for all interactive tools, focusing principally on mouse and keyboard interaction with Avogadro. Examples of tool plugins include the draw tool used to draw molecules atom by atom, and the navigation tool used to pan, rotate and scale the view of the molecule. There are also several specialized tools such as the alignment tool.

Finally there is the Avogadro::Extension class, which defines the interface for dialog based plugins. These extensions can interact with the molecule, and are used for a variety of purposes from molecule properties dialogs to input file generation dialogs for many quantum codes including NWChem, Gaussian, GAMESS and others. This class of plugin is also applied to file import, and network aware extensions querying web databases for structures given their common name for example.

When the application starts up it searches several directories for plugins, which it then attempts to load. The Qt plugin framework is used to check that the plugins have a recent enough version to be loaded, and the plugin type can be deduced once it has been loaded. The user interface is then populated with appropriate entries. The tools being added to the main toolbar using their embedded icons, display types are added to the display type list, and menu entries are added for all loaded extensions.

The tool and display type plugins can both (optionally) return a dialog that is used to configure the plugin. These are specific to each plugin, and provided in the user interface.

4.2 Display Types

Display types are one type of plugin, referred to as engines internally. Their primary focus rendering graphics to the screen. As is the case with most molecular graphics, a large portion of the geometric primitives are spheres and cylinders, typically used to represent atoms and bonds. There are many other properties that can be rendered using the display type plugins.

Some of the engines also convey some information about the underlying data the geometric primitives represent, in order to allow for the molecule to be edited. Engines are performance critical as the render functions are called each time a frame is requested for display. It is very important for the engines to efficiently render all requested data, and multiple display types can be combined to form a composite display, for example ball and stick display overlaid with a transparent Van der Waals space filling diplay and ring rendering to highlight all rings in the structure. Figure 3 (d) and (f) show two such combinations of multiple display types.

4.3 Tools

The tools are responsible for virtually all mouse and keyboard interaction with the molecule. The navigation tool provides basic scene navigation, implementing rotation, panning, tilting and zooming support. This tool provides optional visual cues to show what type of navigation is taking place, and all scene navigation will take place about the center of the molecule if an empty space is clicked, or about the center of the clicked atom. The navigation tool is also used as the default tool if the currently active tool does not handle the mouse event passed to it.

One of the other central tools is the draw tool, this implements a free hand molecule drawing input method, supporting keyboard shortcuts, combo boxes or a periodic table view to select elements. The user can use the left mouse button to add new atoms or bonds, or click on the bonds to change their order. The right mouse button can be used to delete atoms or bonds and the directional keys can be used in combination with the mouse to quickly rotate/pan the molecule.

There are also two tools for adjustment of structures (atom or bond centric), a selection tool supporting standard selection interactions and an auto-rotate tool that allows users to set the speed and angles about which to rotate the molecule. The interactive auto-optimization tool provides a sculpting interaction, where the user can begin a continuous geometry optimization, and switch back to the draw or adjustment tools and change the shape and structure of the molecule while observing the new structure being optimized. This can also be combined with the measurement tool to interactively observe bond lengths and angles evolve as the structure is updated and the geometry minimized. If the optimization tool is turned off the measurement tool also allows the user to precisely adjust bond lengths and/or angles using the adjustment tools.

4.4 Extensions

The extensions represent quite a diverse range of plugins. These range from the input generation dialogs for various quantum chemistry codes such as GAMESS, MOLPRO, NWChem, etc, through to animation of the molecule and visualization of molecular orbitals and electron density. Network aware extensions allow the user to click on File-¿Import-¿Fetch by chemical name and search for “tnt” or “propanol” and have structures returned by the NIH CACTUS Chemical Structure Resolver service.

Other extensions translate the entire scene to POV-Ray input, and call POV-Ray to render the molecule using ray tracing techniques to provide higher quality renderings for publication. Various molecular property dialogs are also implemented as plugins, drawing largely on Open Babel functionality to provide

an overview of the molecule. Cartesian editors, addition and removal of hydrogens, fragment, SMILES and peptide insertion are all implemented as extensions showing up in the Avogadro menus. More recently a crystallography extension was added giving access to a much wider range to functionality useful to practitioners in that area.

4.5 Colors

The color plugins primarily take either double precision numbers or integer values and return an RGB value. The plugins range from the standard color plugin that takes atomic number and returns the standard RGB value for that element through to mapping things like partial change and index to more easily view various aspects of the molecule's structure.

5 Python Interface

Description of the Python bindings, PyQt, plugins as Python classes.

6 Quantum Calculations

Description of the quantum calculation code already implemented. Slater and Gaussian types, output file parsing, parallel calculation code and details of data layout.

7 Avogadro Library in Use

The Avogadro library's first user was the Avogadro application, closely followed by the Kalzium periodic table program that is part of the KDE software collection. This initial work was funded as part of the Google Summer of Code program in 2007, and also resulted in the addition of several other features in the Avogadro library to support Kalzium and general visualization and editing of molecular structure.

7.1 Packmol

7.2 XtalOpt

The XtalOpt [16, 17] software package is implemented as a third-party C++ extension to Avogadro and makes heavy use of the libavogadro API. The extension implements an evolutionary algorithm tailored for crystal structure prediction. The XtalOpt development team chose Avogadro as a platform because of its open source license, well-designed API, powerful visualization tools, and intuitive user-interface. XtalOpt exists as a dialog window (Figure 5) and uses the main Avogadro window for visualizing candidate structures as they evolve. The API is well suited for XtalOpt's needs, providing a simple mechanism to

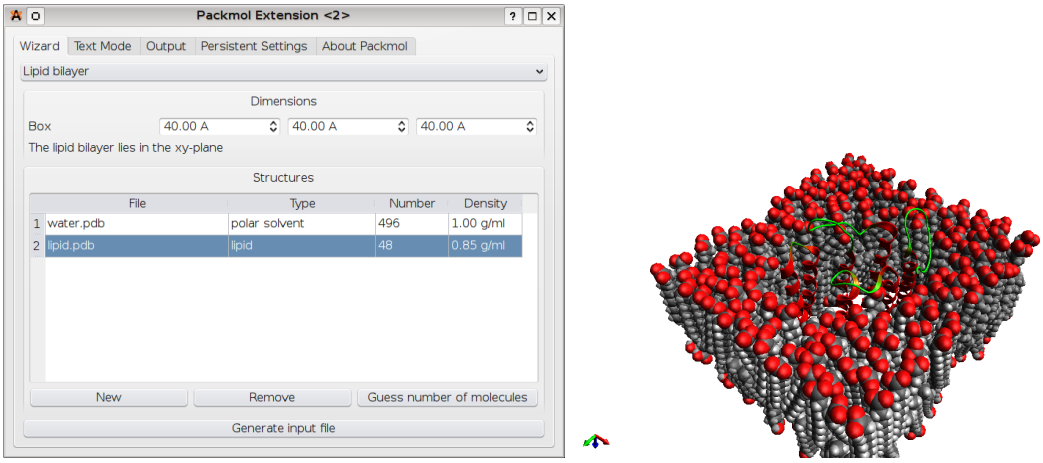

Figure 4: The PackMol extension for Avogadro (left) and a lipid layer (right) as produced by the PackMol program.

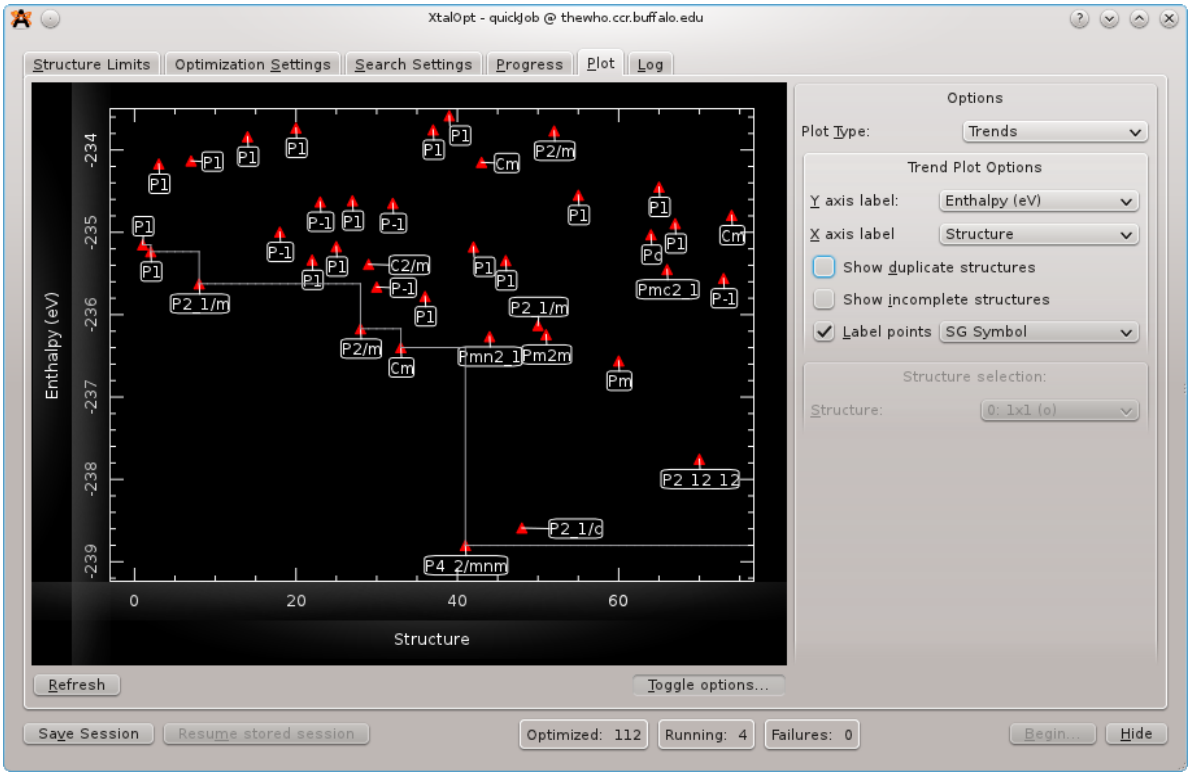

Figure 5: The XtalOpt package showing a plot of stability vs. search progress for a TiO_2 supercell.

allow the user to view, edit, and export the structures generated during the search. Taking advantage of the cross-platform capabilities of Avogadro and its dependencies, XtalOpt is available for Linux, Windows, and Mac.

8 Conclusions

9 Acknowledgements

We wish to thank the many contributors to the Avogadro project, including developers, testers, translators, and users. MDH and GRH thank the University of Pittsburgh for support.

References

1. Sayle R, Milner-White EJ: **RasMol: Biomolecular graphics for all**. *Trends in Biochemical Sciences (TIBS)* 1995, **20**(9):374.
2. **Jmol: an open-source Java viewer for chemical structures in 3D**. *Internet* 2009, [http://www.jmol.org].
3. DeLano WL: **The PyMOL Molecular Graphics System** 2002, [http://www.pymol.org].
4. Humphrey W, Dalke A, Schulten K: **VMD - Visual Molecular Dynamics**. *J. Molec. Graphics* 1996, **14**:33–38.
5. Tarini M, Cignoni P, Montani C: **Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization**. *IEEE Transactions on Visualization and Computer Graphics* 2006, **12**(5):1237–1244.
6. **BALLView**. *Internet* 2009, [http://www.ballview.org].
7. **Materials Studio** 2001-2007. [Accelrys Software Inc.].
8. **CrystalMaker** 2009, [http://www.crystalmaker.com].
9. **Ghemical** 2009, [http://www.uku.fi/~thassine/projects/ghemical].
10. Schaftenaar G, Noordik JH: **Molden: a pre- and post-processing program for molecular and electronic structures**. *J. Comput.-Aided Mol. Design* 2000, **14**:123–134.
11. **The CCP1 GUI Project** 2009, [http://www.cse.scitech.ac.uk/ccg/software/ccp1gui].
12. Allouche AR: **Gabedit** 2009, [http://gabedit.sourceforge.net].
13. Bode BM, Gordon MS: **Macmolplt: a graphical user interface for GAMESS**. *J. Mol. Graphics and Modeling* 1998, **16**(3):133–138.
14. **Open Babel** 2009, [http://openbabel.org].
15. **Eigen** 2009, [http://eigen.tuxfamily.org].
16. **XtalOpt** 2011, [http://xtalopt.openmolecules.net].
17. Lonie D, Zurek E: **XtalOpt: An Open-Source Evolutionary Algorithm for Crystal Structure Prediction**. *Computer Physics Communications* 2011, **182**:372–387.