

# Avogadro: An Advanced Semantic Chemical Editor, Visualization, and Analysis Platform

Marcus D Hanwell<sup>\*1,2</sup>, Donald E Curtis<sup>3</sup>, David Lonie<sup>4</sup>, Tim Vandermeersch<sup>5</sup>, Eva Zurek<sup>4</sup>, Geoffrey R Hutchison<sup>1</sup>

<sup>1</sup>Department of Chemistry, University of Pittsburgh, 219 Parkman Avenue, Pittsburgh, PA, 15260, USA

<sup>2</sup>Department of Scientific Computing, Kitware, Inc., 28 Corporate Drive, Clifton Park, NY, 12065, USA

<sup>3</sup>Department of Computer Science, Coe College, 1220 First Avenue NE, Cedar Rapids, Iowa 52402

<sup>4</sup>Department of Chemistry, State University of New York at Buffalo, Buffalo, New York 14260-3000

<sup>5</sup>Avogadro development team

Email: Marcus D Hanwell<sup>\*</sup> - [marcus.hanwell@kitware.com](mailto:marcus.hanwell@kitware.com); Geoffrey R Hutchison - [geoffh@pitt.edu](mailto:geoffh@pitt.edu);

<sup>\*</sup>Corresponding author

## Abstract

**Background:** The Avogadro project has developed an advanced molecule editor and visualizer designed for cross-platform use in computational chemistry, molecular modeling, bioinformatics, materials science, and related areas. It offers flexible, high quality rendering, and a powerful plugin architecture. Typical uses include building molecular structures, formatting input files, and analyzing output of a wide variety of computational chemistry packages. By using the CML file format as its native document type, Avogadro seeks to enhance the semantic accessibility of chemical data types.

**Results:** The work presented here details the Avogadro library, which provides a framework, application programming interface, three-dimensional visualization capabilities; and has direct applications to 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 through the library itself. The application and library can each be extended by implementing a plugin module in C++ or Python 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.

**Conclusions:** Avogadro is freely available under an open-source license from

<http://avogadro.openmolecules.net>.

## 1 Introduction

Many fields 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], BALLView [6], VESTA [7], and XCrySDen [8] [9], 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, Windows, and BSD).

The choice of software capable of building chemical structures in three dimensions is far smaller. There are existing commercial packages, such as Spartan [10], CAChe/Scigress [11], GaussView [12], Materials Studio [13] and CrystalMaker [14], which are polished and capable of constructing many different types of molecular structures. They are, however, not available for all operating systems (most of them only run on Microsoft Windows), and are not easily extensible, customized, or integrated into automated workflows. Licensing costs can be prohibitive. If the company were to change its direction or focus, this can lead to a loss of a significant research investment in a commercial product. Furthermore, in most cases, these programs use custom, proprietary file formats, and semantic and chemical data can be lost in conversion to other data formats.

The selection of free, open-source, cross-platform, three-dimensional, molecular builders was quite limited when the Avogadro project was founded in late 2006. Ghemical [15] 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 to address the issues they had observed in Ghemical and other packages. The Molden [16] application was also available, able to build up small molecules and analyze output from several quantum codes. However, it suffers from a restrictive license and it uses an antiquated graphical toolkit, which is not native on most modern operating systems.

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. The creation of an open and extensible framework that implements many of the necessary foundations for a molecular builder and visualizer would facilitate more effective research in this area. Further, the open, standardized

Chemical Markup Language (CML) file format [17,18] would be used, to secure semantic and chemical data and allow easy interoperability with other chemistry software.

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 [19], Gabedit [20] and some highly specific editors such as MacMolPlt [21] which focus on particular computational packages (i.e., GAMESS-US for MacMolPlt). Whilst offering many interesting and useful features, these projects suffer from the same issues centering around effective reuse of existing code, well commented and documented code, and easy extension to add new features and adapt for specialized areas.

The Avogadro project was started in earnest in 2007, and over the first 5 years of development has been downloaded over 270,000 times [22], been translated into over 20 languages [23], and has over 20 contributors [24]. From the beginning, the project has strived to make a robust, flexible 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 platform. The development team has also been members of the Blue Obelisk movement, following the three pillars outlined by the group: Open Data, Open Standards, and Open Source [25,26].

## 2 The Graphical User Interface

The first thing most people will see is the main Avogadro application window, as shown in Figure 1.

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 [27] [28] 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. It is very easy for new users to install Avogadro and build their first molecules within minutes. Thanks to the Open Babel library [29], Avogadro supports a large portion of the chemical file formats that are in common use.

The vast majority of this functionality has been written using the interface made available to plugin writers,



Figure 1: The Avogadro graphical user interface (on Mac OS X), showing the editing interface for a molecule.

and is loaded at runtime. We will discuss these plugin interfaces and descriptions of the plugin types later.

### 3 Semantic Chemistry

Avogadro has used CML [17,18] as its default file format from a very early stage; this was chosen over all other file formats because of the extensible, semantic structure provided by CML, and the support available in Open Babel [29]. The CML format offers a number of advantages over others in common use, including the ability to extend the format. This allows Avogadro and other programs to be future-proof, adding new information and features necessary for an advanced semantically-aware editor at a later time, while still remaining readable in older versions of Avogadro.

Through the use of Open Babel [29], a large array of file formats can be interpreted. When extending Avogadro to read in larger amounts of the output from quantum codes, it was necessary to devote significant development resources to understanding and adding semantic meaning to the quantum code outputs. This work was developed in a plugin, which was later split out into a small independent library called OpenQube [30] [31]. More recently a large amount of work has been done by the Quixote project [32], JUMBO-Converters, and the Semantic Physical Science workshop to augment quantum codes to output more of this data directly from the code. Since CML can be extended, it is possible to reuse existing conventions for molecular structure data, and add new conventions for the additional quantum data.

#### 3.1 Building a Molecule: Atom by Atom

After opening Avogadro a window such as that shown in Figure 1 is presented. By default, the draw tool is selected. 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 is drawn between the start point and the final position where the mouse is 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 it to the currently selected element, dragging changes the atom back to its previous element and draws a new atom bonded to the original. If the bonds are left-clicked then the bond order cycles between single, double, and triple. Similarly, typing the atomic symbol (e.g., “C-o” for cobalt) changes the selected element, or typing the numbers “1,” “2,” and “3” changes the bond order.

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.

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 (e.g., altering the bond length, bond angles, or dihedral angles). These three tools allow for a great deal of flexibility in building small molecules interactively on screen.

Once the molecular structure is complete, the force field 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 force field 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 implemented to provide information on molecule properties and to precisely change parameters, such as the cartesian coordinates of the atoms in the molecule.

### 3.2 Building a Molecule: From Fragments

In addition to building molecules atom-by-atom, users can insert pre-built fragments of common molecules, ligands, or amino-acid sequences, as shown in Figure 2.

In all cases, after inserting the fragment, the atom-centered manipulate tool is selected, allowing the fragment to be moved or rotated into position easily.

Users can also insert a SMILES [33] [34] string for a molecule. In this case, a rough 3D geometry is generated using Open Babel and a quick force field optimization.

### 3.3 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, NWChem, Gaussian, Q-Chem, and MOPAC. The graphical dialogs present the features required to run basic quantum calculations; some examples are shown in Figure 3.

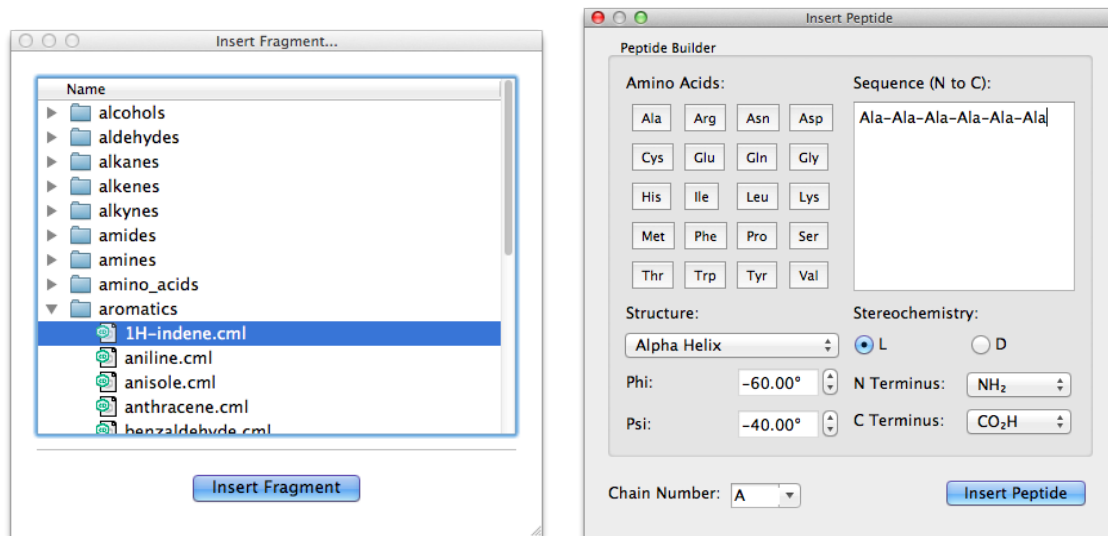


Figure 2: Dialogs for inserting pre-built fragments for (left) molecules, and (right) amino-acid sequences.

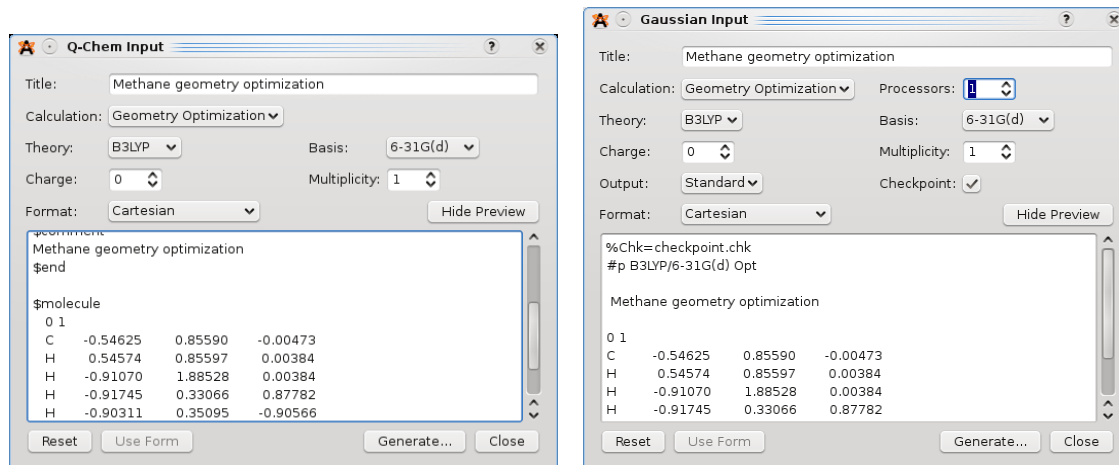


Figure 3: Dialog for generating input for Q-Chem (left) and Gaussian (right). Note that both dialogs are similar in interface, allowing users to use multiple computational chemistry packages.

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 also be edited by hand 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, and then reload the output file into Avogadro once the calculation is complete. This feature will be extended to other quantum codes in future versions of Avogadro.

The GAMESS-US plugin is one of the most highly developed, featuring a basic dialog present in most of the other input deck generators, as well as an advanced dialog exposing many of the more unusual and complex calculation types. In addition to the advanced dialog, the input deck can be edited inline and features syntax highlighting as used in many popular editors aimed at software developers. This can indicate simple typing errors in keywords, as well as harder to spot whitespace errors that would otherwise cause the hand-edited input deck to fail when being read by GAMESS-US. Some new features in the GAMESS plugin will even allow for the submission of GAMESS-US jobs to computational clusters.

### **3.4 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 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 could be created. The alignment tool was created in just a few hours for a specific research project. This is a prime example where extensibility was very important for performing 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.



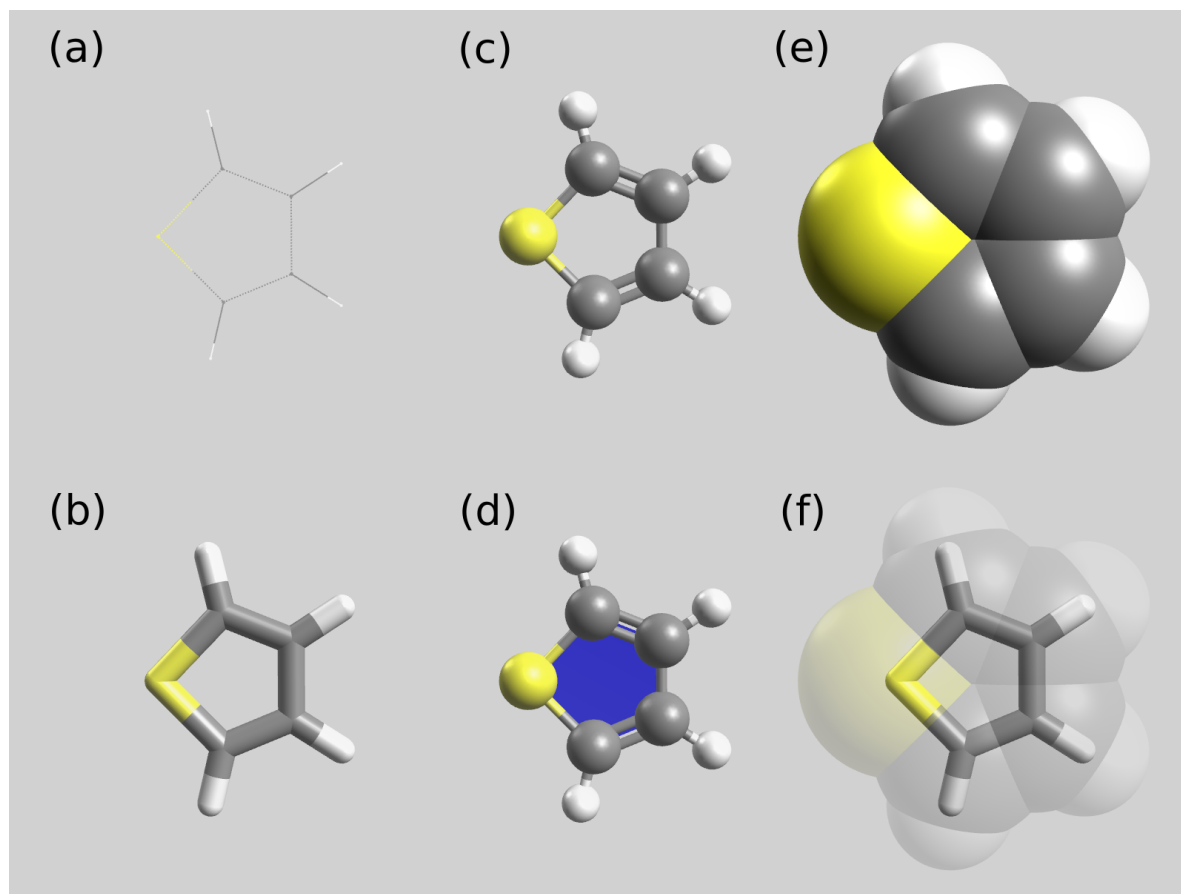


Figure 4: Several molecular representations of thiophene, (a) wireframe, (b) stick/licorice, (c) ball and stick, (d) ball and stick with ring, (e) Van der Waals/CPK and (f) transparent Van der Waal's with stick.

## 4 Visualization

The Avogadro application uses OpenGL to render molecular representations to the screen interactively. OpenGL offers a high-level, 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 be used even on modest, older 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.

### 4.1 Standard Representations

In chemistry, there are several standard representations of molecular structure, originally based upon those possible with physical models. The Avogadro application implements each of these representations shown

in Figure 4 as a plugin. These range from the simple wireframe representation, stick/licorice, ball and stick, and Van der Waals spheres.

It is also possible to combine several representations, such as ball and stick with ring rendering (Figure 4 (d)), and a semi-transparent Van der Waals space-filling representation with a stick representation to elucidate molecular backbone (Figure 4 (f)).

## 4.2 Electronic Structure

Quantum codes were originally developed for line printers, and unfortunately little has changed since then in the standard log files. There are several formats developed for use in other codes and specifically for visualization and analysis, but there is little agreement on any standard file format in the computational quantum chemistry community. A plugin was developed in Avogadro to visualize the output of various quantum codes, and get the data into the right format for further visualization and analysis.

Initially support was added and extended in Open Babel for Gaussian cube files. This format provides atomic coordinates and one or more regularly spaced grids of scalar values. This can be read in, and techniques such as the marching cubes algorithm can be used to compute triangular meshes of isosurfaces at values of electron density for example. Once the code has been developed to visualize these isosurfaces, it became clear that it would be useful to be able to calculate these cubes on the fly, and at different levels of detail depending upon the intended use.

The first format, which was somewhat documented at the time it was developed, is the Gaussian formatted checkpoint format. This format is much easier to parse than the log files generated as the program runs, and provides all of the detail needed to calculate scalar values of the molecular orbital or electron density at any point in space. Once a class structure had been developed for Gaussian type orbitals, the approach was extended to read in several other popular output file formats including Q-Chem, GAMESS-US, NWChem, and MolPro. MOPAC support was added later, along with support for the AUX format and Slater type orbitals used in that code. All of these codes output their final configurations using the standard linear combination of atomic orbitals, meaning that parallelization is extremely simple.

The plugin was developed to take advantage of the map-reduce approach offered by QtConcurrent in order to use all available processor cores. This offers almost linear scaling as each point in the grid can be calculated independently of all other points. An alternate approach to calculating the molecular orbitals was developed in a second plugin, and the relevant code was split out into a support library later named “OpenQube”. This library has now been split off into a separate support project outside of Avogadro, and

was added as an optional backend in VTK during the 2011 Google Summer of Code, bringing support for several output file formats and calculation of cube files that can later be fed into more advanced data pipelines.

There are several related projects for adding semantic meaning to this type of output, including the JUMBO-Converters project and Quixote. It is hoped that more codes will adopt semantic output in the future, using a common format so that data exchange, validation, and analysis become easier across several codes. This was the subject of a recent meeting with several computational chemistry codes beginning to use FoX in order to output CML. Development has begun on code to read in CML output, either directly from the codes or from conversion of other formats using Open Babel or the JUMBO-Converters. If enough semantic structure can be added to CML, and the converters support a large enough range of the output, this could replace most of the parsing code present in OpenQube. Semantic meaning is one of the most difficult to extract from log files, and coming together as a community will help projects like Avogadro to derive more meaning from the outputs of these codes.

### **4.3 Secondary Biological Structure**

Avogadro uses the PDB reader from Open Babel, and is able to visualize the secondary structure annotated in those files. A simple plugin was initially developed that rendered 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.

### **4.4 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 the package. 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.

## 4.5 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 and 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.

## 5 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 minimize 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.

Avogadro has close ties to several other free, cross-platform, open-source projects to reuse as much code as is practical. These projects include Nokia Qt to provide a free, cross-platform graphical toolkit; Open Babel [29] for chemical file input/output, geometry optimization, and other chemical perception; Eigen [35] for matrix and vector mathematics; OpenGL/GLSL for real-time, three-dimensional rendering; and POV-Ray for ray-traced rendering.

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 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 for 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.

## 5.1 Plugin 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.

The application searches several directories for plugins at start up, 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 are 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.

## 5.2 Display Types

Display types are one type of plugin, referred to as engines internally. Their primary focus is 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, 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; 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 display and ring rendering to highlight all rings in the structure. Figure 4 (d) and (f) show two such combinations of multiple display types.

## 5.3 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

Table 1: List of default display type (engine) plugins










Name	Description
Axes	Renders x, y, z Cartesian axes from the origin
Ball and Stick	Standard ball and stick representation
Cartoon	Secondary biological structure ( $\alpha$ helix and $\beta$ sheet)
Dipole	Render direction/magnitude of dipole moment if present
Force	Renders arrows showing forces on atoms from force field
Hydrogen Bond	Renders hydrogen bonds as dotted lines
Label	Shows labels on atoms and bonds, configurable
Overlay	Overlay of color gradient used for electrostatic properties
Polygon	Renders closed polygons of metallic centers
Ribbon	Basic secondary structure ribbon rendering
Ring	Renders rings in structure, different colors depending on ring size
Simple Wireframe	Very simple wireframe display
Sticks	Stick or liquorice rendering style for atoms and bonds
Surface	Renders triangular isosurface meshes
Van der Waals Spheres	Van der Waals sphere rendering (no bonds, space-filling)
Wireframe	Wireframe with more features such as bond order rendering

does not handle the mouse event passed to it.

One of the other central tools is the draw tool, which 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.

Table 2: List of default mouse tool plugins

Icon	Tool Name	Description
	Draw Tool	Build and edit atoms
	Navigate Tool	Move the camera, rotate, pan, and zoom
	Bond Centric Manipulate Tool	Alter bond lengths, angles, and torsions
	Manipulate Tool	Move atoms and selected fragments
	Select Tool	Select individual atoms, bonds, or fragments
	Auto Rotate Tool	Continuously rotate a molecule for presentations
	Auto Optimize Tool	Continuously optimize molecular geometry using molecular mechanics
	Measure Tool	Determine bond lengths, angles, and dihedrals
	Align Tool	Rotate and translate to a specified frame of reference

## 5.4 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 Avogadro menus. More recently a crystallography extension was added, giving access to a much wider range to functionality useful to practitioners in that area, including Miller Plane visualization, slab and surface generation. New builders for nanotubes, nanoparticles, and DNA are also planned for upcoming releases.

## 5.5 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 charge and index to more easily view various aspects of the molecule’s structure.



Table 3: List of default extension commands

Name	Description
Create Surfaces	Create surface meshes from molecular orbital/electron density data
GAMESS	Prepare input files for GAMESS-US, featuring syntax highlighting, advanced properties
Insert Fragment	Insert molecular fragments from a library of common fragments
Insert Peptide	Build up and insert peptide fragments
Molecular Mechanics	Use Open Babel’s force fields for geometry optimization and conformer searches
MOPAC	Prepare input for and run MOPAC
POV-Ray	Ray-trace the displayed structure using POV-Ray
<b>Properties</b>	
Angle Properties	Table of all bond angles (editable)
Atom Properties	Table of all atoms with common properties
Bond Properties	Table of all bonds with common properties
Molecule Properties	Common properties of the molecule (including molecular weight, etc.)
Torsion Properties	Table of all dihedral angles (editable)
Spectra	Visualize spectra from output files
Super Cell Builder	Expand atoms with space group, replicate specified repeats and perform simple bonding
Unit Cell	Change crystallographic unit cell display and parameters
Vibrations	Show and animate molecular vibrations

Table 4: List of default color plugins

Name	Description
Atom Index Color	Color based on atom ID (from atom 1, 2, etc.)
Charge Color	Color based on predicted electrostatic partial charge
Custom Color	Color all atoms a specific, custom color
Distance Color	Color based on distance from one end of the molecule
Element Color ( <b>Default</b> )	Standard color scheme, giving each atom a color defined by its element
Residue Color	Color based on amino acid or nucleic acid residue (i.e., glycine, histidine, etc.)
SMARTS Color	Color atoms matching a specific SMARTS pattern with a custom color

By defining a plugin interface for coloring atoms, bonds, or residues, developers can easily offer flexible rendering options to highlight important information without requiring a user to tediously set colors on specific atoms or functional groups. Default color plugins are listed in Table 4, illustrating the variety of options. Each plugin is usually only 40-50 lines of C++ code.

## 6 Python Interface

Python bindings are provided for all of the core API. Python code can be used in two ways: the first is the interactive Python terminal, and the second is to write Python plugins. Python plugins can be extensions, tools, or display types. Writing a Python plugin requires the same functions to be implemented as a native C++ plugin. The advantage of Python plugins is that it’s easier to make prototypes since no compilation is required. Python plugins can also easily be shared with other users.

The Python bindings also interface with the PyQt python bindings for the Qt toolkit, which enables Python code to use all of Qt's features when writing a plugin.

## 7 Quantum Calculations

Once quantum calculations have been performed, it is desirable to be able to visualize various properties including the molecular orbitals and electron density of the final state of the system. The OpenQube library houses most of the code that reads in the relevant matrices and factors. This code takes care of scanning the output files for the output, and putting the output into a standard format as expected by the calculation code.

Once the data has been parsed and normalized, some initialization is necessary. Once the initialization of the calculation is complete, these data structures can be treated as read-only. This allows the calculation to be multithreaded, but to share one copy of the input data. The scalar value of the molecular orbital or electron density is calculated as a linear combination of atomic orbitals. The parallelization takes place across a regularly spaced cube with each point being calculated in an independent thread sharing the input arrays and matrices. The QtConcurrent framework abstracts away thread pool management, using as many cores as available on the host system.

Once a cube has been calculated, it is passed to the marching cubes algorithm to calculate an isosurface. Each isosurface is calculated in a separate thread; at this stage the cube is read only, and so both the positive and negative isosurface of a molecular orbital can be calculated in separate threads. Further parallelization of this stage is possible, but typical isosurfaces do not take long to calculate, and offloading this calculation to the GPU is likely to be more useful as the result would already be in GPU memory for rendering.

A class hierarchy with a standard API is provided for quantum output. Adding support for new codes involved developing a new parser, and ensuring the Gaussian or Slater set is populated with the correct ordering and the expected normalization scheme. The s, p, and d-type Gaussian orbitals are supported, with f and g support planned in order to support the increasing number of calculations using these higher-order orbitals. The Basis Set Exchange hosted by EMSL provides access to the basis sets in common use, although at present these basis sets are normally read in directly from the output files.

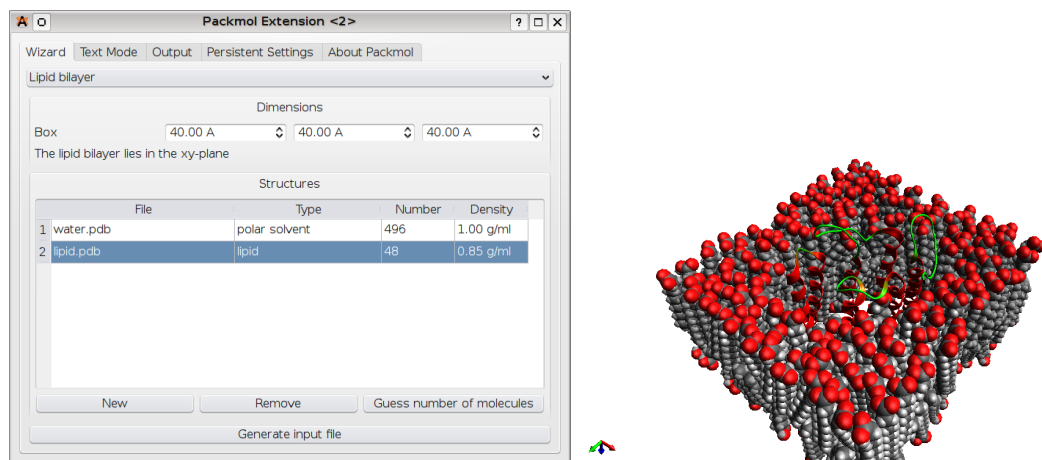


Figure 5: The PackMol extension for Avogadro (left) and a lipid layer (right) as produced by the PackMol program.

## 8 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.

### 8.1 Packmol

Packmol is a third-party package designed to create initial “packed” configurations of molecules for molecular dynamics or other simulations [36] [37]. Examples include surrounding a protein with solvent, solvent mixtures, lipid bilayers, spherical micelles, placing counterions, adding ligands to nanoparticles, etc. Typically, users may have equilibrated “solvent boxes” which have been run for long simulations to ensure proper density, and both short and long-range interactions between solvent molecules. Using such solvent boxes allows placing solute molecules, such as proteins, in an approximately correct initial structure. The solute is added into the box, and solvent molecules with overlapping atoms are removed. While these utilities are often enough, creating complex input files is not always easy.

For more complicated systems, Packmol can create an initial configuration based on defined densities, geometries (e.g., sphere, box, etc.), and the molecules to be placed. An Avogadro developer wrote an external plugin to facilitate use of Packmol, including estimating the number of molecules in a given volume.

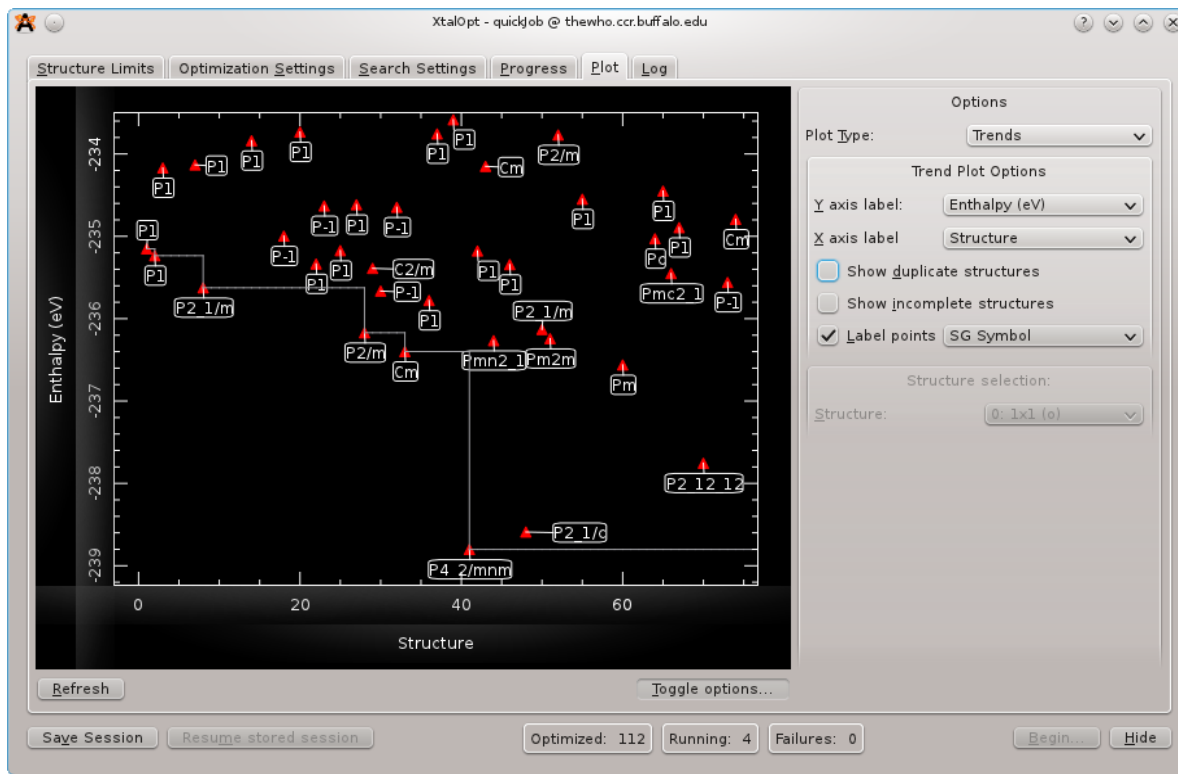


Figure 6: The XtalOpt package showing a plot of stability vs. search progress for a  $\text{TiO}_2$  supercell.

The plugin is not currently distributed with Avogadro as a standard feature, although it is planned for some future version. It serves as an example of how Avogadro can facilitate a workflow with a text-oriented package (Packmol), including saving files in the PDB format required by Packmol, generating an input file, and reading the output for visualization, analysis, and further simulations.

## 8.2 XtalOpt

The XtalOpt [38, 39] 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 6) 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 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.

## 9 Conclusions and Future Directions

Currently, two future versions of Avogadro are under development. The first is Avogadro version 1.1, which adds additional features and refinement, particularly including crystallography support developed through the XtalOpt project. The second is a more substantial development for Avogadro version 2.0, where many of the core data structures are being rewritten in order to offer greater flexibility and scalability.

## 10 Availability and Requirements

**Project Name:** Avogadro

**Project home page:** <http://avogadro.openmolecules.net/>

**Operating system(s):** Cross-platform

**Programming language:** C++, bindings to Python

**Other requirements (if compiling):** CMake 2.6+

**License:** GNU GPL v2

**Any restrictions to use by non-academics:** None

## 11 Acknowledgements and Funding

We wish to thank the many contributors to the Avogadro project, including developers, testers, translators, and users. We thank SourceForge for providing resources for issue tracking and managing releases, Launchpad for hosting language translations, and Kitware for additional dashboard resources. MDH and GRH thank the University of Pittsburgh for support. MDH also thanks Kitware and the Engineering Research Development Center for support. EZ and DL acknowledge the NSF (DMR-1005413) for financial support.

## 12 Authors' contributions

GRH and DEC are the founders of the Avogadro project. MDH is the current lead developer and maintainer of Avogadro. GRH, DL and TV are active developers. DL and EZ are founders of the XtalOpt project which is discussed in this work. TV developed the PackMol plugin. All authors read and approved the final manuscript.

### **13 Competing interests**

The authors declare that they have no competing interests.

## 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* 2012, [<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* 2012, [<http://www.ballview.org>].
7. Momma K, Izuma F: **VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data**. *J. Appl. Cryst.* 2011, **44**:1272–1276.
8. Kokalj A: **XCrySDen—a new program for displaying crystalline structures and electron densities**. *J. Mol. Graphics and Modeling* 1999, **17**(3-4):176–179.
9. Kokalj A: **Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale**. *Comput. Mater. Sci.* 2003, **28**(2):155–168.
10. **Spartan** 2012, [<http://www.wavefun.com>].
11. **SCIGRESS** 2012, [<http://www.fujitsu.com/global/services/solutions/tc/hpc/app/scigress/>].
12. **GaussView 5** 2012, [[http://gaussian.com/g\\_prod/gv5.htm](http://gaussian.com/g_prod/gv5.htm)].
13. **Materials Studio** 2001-2007. [Accelrys Software Inc.].
14. **CrystalMaker** 2012, [<http://www.crystallmaker.com>].
15. **Ghemical** 2012, [<http://www.uku.fi/~thassine/projects/gchemical>].
16. 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.
17. Murray-Rust P, Townsend J, Adams S, Phadungsukanan W, Thomas J: **The semantics of Chemical Markup Language (CML): dictionaries and conventions**. *Journal of Cheminformatics* 2011, **3**:43, [<http://www.bibsonomy.org/bibtex/2fefad2c4ce89d6d69c43ebef250db9cd/fairybasslet>].
18. Murray-Rust P, Rzepa H: **CML: Evolution and Design**. *Journal of Cheminformatics* 2011, **3**:44, [<http://www.bibsonomy.org/bibtex/21e7fc4450ba8cbb0ad2ab803b395ceb/fairybasslet>].
19. **The CCP1 GUI Project** 2009, [<http://www.cse.scitech.ac.uk/ccg/software/ccp1gui>].
20. Allouche AR: **Gabedit** 2012, [<http://gabedit.sourceforge.net>].
21. Bode BM, Gordon MS: **Macmolplt: a graphical user interface for GAMESS**. *J. Mol. Graphics and Modeling* 1998, **16**(3):133–138.
22. **Avogadro Downloads** 2012, [<http://sourceforge.net/projects/avogadro/files/stats/timeline?dates=2006-04-14+to+2012-03-29>].
23. **Avogadro Translations** 2012, [<https://translations.launchpad.net/avogadro/trunk/+translations>].
24. **Avogadro Contributors** 2012, [<http://www.ohloh.net/p/avogadro/contributors>].
25. Guha R, Howard MT, Hutchison GR, Murray-Rust P, Rzepa H, Steinbeck C, Wegner J, Willighagen EL: **The Blue Obelisk - Interoperability in Chemical Informatics**. *Journal of Chemical Information and Modeling* 2006, **46**(3):991–998, [<http://www.bibsonomy.org/bibtex/2c3adb68c56afef407bcd2eb8d7c25def/fairybasslet>].
26. O’Boyle N, Guha R, Willighagen E, Adams S, Alvarsson J, Bradley JC, Filippov I, Hanson R, Hanwell M, Hutchison G, James C, Jeliazkova N, Lang A, Langner K, Lonie D, Lowe D, Pansanel J, Pavlov D, Spjuth O, Steinbeck C, Tenderholt A, Theisen K, Murray-Rust P: **Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on**. *Journal of Cheminformatics* 2011, **3**:37, [<http://www.bibsonomy.org/bibtex/2c4a22b9baa08515e7318f2bb57333f0f/fairybasslet>].

27. **Compiling Avogadro on Windows** 2012, [[http://avogadro.openmolecules.net/wiki/Compiling\\_on\\_Windows](http://avogadro.openmolecules.net/wiki/Compiling_on_Windows)].
28. **Compiling Avogadro on Linux** 2012, [[http://avogadro.openmolecules.net/wiki/Compiling\\_on\\_Linux\\_and\\_Mac\\_OS\\_X](http://avogadro.openmolecules.net/wiki/Compiling_on_Linux_and_Mac_OS_X)].
29. O’Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR: **Open Babel: An open chemical toolbox**. *J. Cheminf.* 2011, **3**:33.
30. **OpenQube** 2012, [<http://www.openchemistry.org/>].
31. **OpenQube Source** 2012, [<http://github.com/OpenChemistry/openqube>].
32. Adams S, de Castro P, Echenique P, Estrada J, Hanwell MD, Murray-Rust P, Sherwood P, Thomas J, Townsend JA: **The Quixote project: Collaborative and Open Quantum Chemistry data management in the Internet age**. *Journal of Cheminformatics* 2011, **3**:38, [<http://www.bibsonomy.org/bibtex/25479ed903177c988fc14266200510ffa/fairybasslet>].
33. Weininger D: **SMILES, a chemical language and information system. 1 Introduction to methodology and encoding rules**. *J. Chem. Inf. Comp. Sci.* 1988, **28**:31–36.
34. **OpenSMILES** 2012, [<http://opensmiles.org/>].
35. **Eigen** 2012, [<http://eigen.tuxfamily.org>].
36. Martínez L, Andrade R, Birgin EG, Martínez JM: **Packmol: A package for building initial configurations for molecular dynamics simulations**. *J. Comp. Chem.* 2009, **30**(13):2157–2164.
37. Martínez JM, Martínez L: **Packing optimization for automated generation of complex system’s initial configurations for molecular dynamics and docking**. *J. Comp. Chem.* 2003, **24**(7):819–825.
38. **XtalOpt** 2011, [<http://xtalopt.openmolecules.net>].
39. Lonie D, Zurek E: **XtalOpt: An Open-Source Evolutionary Algorithm for Crystal Structure Prediction**. *Computer Physics Communications* 2011, **182**:372–387.