Fast and Scriptable Molecular Graphics in Web Browsers without Java3D

E.L. Willighagen and M. Howard

Introduction
Jmol is a free, open source molecule viewer for chemistry and biochemistry. It is cross-platform, running on Windows, Mac OS X, and Linux/Unix systems. The software consists of three parts: the JmolApplet is a web browser applet that can be integrated into web pages; the Jmol application is a standalone Java application that runs on the desktop; and the JmolViewer is a development tool kit that can be integrated into other Java applications.

Features
Jmol as many features to offer in the fields of biochemistry and chemistry. It supports rendering of secondary structures or proteins and nucleic acids, animation of vibrations and animations originating from quantum mechanical calculations, and rendering of unit cell information of crystallographic information. Jmol supports many file formats among which CIF/mmCIF, CML, GAMESS output, Gaussian 94/98/03 output, Ghemical, HIN (HyperChem), Jaguar output, NWChem, MDL MOL/SD, MOPAC 93/97/2002 output, PDB, Q-Chem, SHELX, Spartan output, and XYZ. Files might be gziped to reduce download time when used with the JmolApplet.

The origins of Jmol
Jmol was originally intended to be a fully functional replacement for XMol which was a molecular viewing program developed at the Minnesota Supercomputer Center. Although the program executables were distributed, the source code was not available to users, and since the program has not been maintained, the free binary versions have become obsolete. For example, the SGI version of XMol does not run on IRIX 6.x because IRIX has switched to a new executable format.
XMol’s demise left a need for a similar tool. Dan Gezelter, the originator of Jmol, chose to avoid the same problems by making Jmol open source. Although it has not completely met the goal of functionally replacing XMol, Jmol duplicates many of the most useful features, and in some ways, it has surpassed XMol functionality. Jmol was started as an OpenScience project which is dedicated to writing and releasing free and open source scientific software. Later, Bradley A. Smith took over the project and did a lot of work in streamlining the project as well as the software. Under his leadership new releases found its way to the user base, and many new features were added, some of which contributed by users. In the end of 2002, Egon Willighagen became the new project leader and a start was made with integrating Jmol with The Chemical Development Kit, something that was already planned in September 2000 by Dan, Egon and Christoph Steinbeck.
Miguel joined the project at the end of 2002, with the explicit goal of helping build Jmol into a viable replacement for the Chime plugin (www.mdlchime.com) in the first few months Miguel contributed the Spanish translation and made an initial pass at the RasMol/Chime script interpreter. He then some performance improvements and began working on rendering performance and speed. It quickly became apparent that the Java2D graphics did not provide the performance and functionality needed. In addition, it became clear that the core Jmol classes would not support the performance and space requirements for working with macromolecules with 10s of thousands of atoms.
In the spring of 2003 Miguel began designing and implementing a high-performance software-based graphics engine. In order to ensure web deployment of the Jmol applet it was important that the engine run on version 1.1 Java Virtual Machines and that no specialized graphics hardware be required. Miguel also rewrote the core classes in order to support much larger molecules. In addition, he separated all file IO from the Jmol core, making it much easier to support new file types.
An extended test period began at the end of 2003. During 2004 a small set of users around
the world contributed to Jmol development by testing Jmol, explaining chemical concepts, and explaining scripting behaviors of RasMol and Chime. Jmol version 10, an open source replacement for the Chime plugin, was released in December 2004. The results can be viewed online at http://www.jmol.org.

Graphics Engine
The Jmol graphics engine is a completely written in Java. There is no Java3D, OpenGL or any type of hardware acceleration. The graphics engine is a special purpose graphics engine, not a general purpose 3D graphics library. The rendering engine was built especially for displaying molecules, so it does a very good job of drawing spheres and cylinders. It is a completely software implementation of a z-buffer. There is a int[] pixelBuffer which holds ARGB values and a short[] zBuffer which holds the pixel depth at that point. During the repaint cycle, the entire scene is rendered into this pixelBuffer. Then the entire scene is drawn as one image with a single Graphics.drawImage(...) operation. It does not use a display list of triangular meshes. That would be too slow. Instead, there are optimized routines for drawing spheres and cylinders.

Since the engine is not using Java3D, it is not using any hardware acceleration specific for graphics. This would suggest a significant performance drop. However, Java3D can only render triangles. Rendering a curved surface requires tessellating the surface to a set of triangles that approximate the surface. If one is only rendering spheres and cylinders, the CPU cost of managing all the triangles may well exceed the cost of rendering the sphere as pixels, as Jmol does. Moreover, Sun's 1.4.2 JVM nowadays uses processor extensions like SSE and SSE2 speeding up floating point calculations.

Though this article does not present a full analysis, some numbers are given below:

<table>
<thead>
<tr>
<th>Structure</th>
<th>CPU Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspirin</td>
<td>100</td>
</tr>
<tr>
<td>protein 114D</td>
<td>100</td>
</tr>
<tr>
<td>protein 1OHG</td>
<td>35</td>
</tr>
<tr>
<td>membrane</td>
<td>17</td>
</tr>
<tr>
<td>rhino virus</td>
<td>2</td>
</tr>
</tbody>
</table>

The calculations were done on an AMD Athlon XP 2200+ with 1GB of memory running Jmol with dimensions equal to 500x500 pixels in balls-and-sticks rendering mode. We believe that such performance is more than adequate.

Conclusion
Jmol has grown since the start into an excellent example of what Java can offer the scientific and educational community. The uptake of the opensource project in this community has been large: at the time of writing the JmolApplet is used at at least sixteen websites and the JmolViewer is integrated in at least four Java programs.

More information on Jmol can be found on the Jmol website at http://www.jmol.org/ and users are welcomed to participate in the project on Jmol's wiki website at http://wiki.jmol.org/

Acknowledgments
Though this poster is written by the stated authors, the Jmol project has evolved by work of many others, which the authors greatly acknowledge, and especially the two former project leaders Dan Gezelter and Bradley Smith.