ELSEVIER

MoldaNet: A network distributed molecular graphics and modelling program that integrates secure signed applet and Java 3D technologies

Hiroshi Yoshida,* Henry S. Rzepa,† and Alan P. Tonge†

**Department of Chemistry, Faculty of Science, Hiroshima University, Higashi-Hiroshima 739-8526, Japan*
†Department of Chemistry, Imperial College, London SW7 2AY, UK

MoldaNet is a molecular graphics and modelling program that integrates several new Java technologies, including authentication as a Secure Signed Applet, and implementation of Java 3D classes to enable access to hardware graphics acceleration. It is the first example of a novel class of Internet-based distributed computational chemistry tool designed to eliminate the need for user pre-installation of software on their client computer other than a standard Internet browser. The creation of a properly authenticated tool using a signed digital X.509 certificate permits the user to employ MoldaNet to read and write the files to a local file store; actions that are normally disallowed in Java applets. The modularity of the Java language also allows straightforward inclusion of Java3D and Chemical Markup Language classes in MoldaNet to permit the user to filter their model into 3D model descriptors such as VRML97 or CML for saving on local disk. The implications for both distance-based training environments and chemical commerce are noted. © 1999 by Elsevier Science Inc.

INTRODUCTION

The evolution of the World Wide Web (WWW) since 1993, and the introduction of Java technology in 1995, has introduced a new Internet-based computational chemistry computing application and database paradigm.[1] Using a WWW browser as a so-called thin-client connected to a TCP/IP-based network makes possible the operation of applications (referred to as *applets*) on a network-connected client computer. Java, an object-oriented programming platform-independent language first released by Sun Microsystems in 1995, allows Java-based applets to run on any system that supports a Java Virtual Machine (JVM), such as the Netscape or Internet Explorer browsers. Java-based applets can be mounted on a WWW or application server to be delivered to WWW clients in a similar manner to HTML-based documents. We were attracted to such a distributed computing model as a valuable tool for creating distance learning-based course components in chemistry.

We have developed a Java-based molecular modelling program named *MOLDA for Java*.[2] This program evolved from a platform-dependent version of Molda written in BASIC, which had been developed in the early 1980s.[3] MOLDA for Java was written in Java in the first instance to achieve platform independence. In this initial implementation, MOLDA for Java was created as a conventional application and not as a network-deliverable applet, using version 1.02 of the Java Development Kit (JDK) released by Sun Microsystems. This was because of the need for its user to read and write local files, actions that conventional applets are not permitted in order to prevent potential rogue actions on the user's computer system (e.g., reading or deleting files) by a hostile attacker. Before running a Java-based application, the user must install an appropriate Java run-time environment for the operating system as well as the Molda application, which imposes a significant level of local complexity and raises issues of software maintenance and upgrades. Since Molda was developed primarily as a teaching and training tool to be used in local clusters of computers, we felt this remained an issue to be resolved.

A second problem that must be addressed is the graphical performance of the software. In MOLDA for Java, only wire-frame models were permitted, since high-quality 3D graphics such as space-filling and ball-and-stick models refreshed too slowly on the screen to be useful. While these problems can in part be solved by implementing highly efficient screen-drawing algorithms such as used in, e.g., RasMol,[4] the more recent

Address correspondence to: H. Yoshida, Department of Chemistry, Faculty of Science, Hiroshima University, Higashi-Hiroshima, 739-8526, Japan. E-mail: h.rzepa@ic.ac.uk.

Journal of Molecular Graphics and Modelling 16, 144–149, 1998

655 Avenue of the Americas, New York, NY 10010
1093-3263/98/$–see front matter
PIIS1093-3263(00)00004-2

generations of personal computers implement effective hardware acceleration solutions to this problem, which we needed to address.

To overcome these problems, we have made use of two newly introduced Java technologies. With the release of Web browsers supporting version 1.1 or 1.2 of the Java Development Kit, it has now become possible to create so-called trusted applets, which introduce the ability to work outside the security restrictions normally imposed on applets (referred to as the *sandbox restriction*). Trusted applets are provided with a verifiable digital signature based on so-called X.509 standards,[5] authorised by a recognised third party Certification Authority as described in Tonge et al.[6] By signing the applet, we have been able to create a distributable chemical application that can acquired by the user on demand, and that has no local preinstallation requirements other than that of the browser itself. Implementing the new Java 3D class libraries available in JDK 1.2 also allows us high access to the low-level graphics card device drivers. This permits the creation of a scene graph-based programming model for creating and manipulating 3D geometry and a simple and flexible mechanism for representing and rendering complex 3D environments.

In the present work, we report how these two features were implemented to create a network transportable molecular graphics and modelling applet termed *MoldaNet*. The new features allow the user to read and write the files to a local file store and greatly enhance the 3D rendering speeds. The Java3D classes also include the ability to filter a model into 3D model descriptors such as VRML97.[7] We have also included some component classes deriving from Chemical Markup Language (CML)[8] to illustrate how the interoperability with other Java class libraries can be used to rapidly extend the functionality of the program. Also included are standard import and export functions for popular computational chemistry packages such as MOPAC[9] and GAUSSIAN.[10] This approach is similar to that used in the creation of Java beans from the ChemSymphony[11] suite of tools.

JAVA API IMPLEMENTATION

The Java 3D API is a programming interface used for writing 3D graphics applications and applets. The Java 3D API uses a scene graph-based programming model and provides a simple and flexible mechanism for representing and rendering complex 3D environments. The scene graph contains a complete description of the entire scene, or virtual universe. This includes the geometric data, the attribute information, and the viewing information needed to render the scene from a particular point of view. The use of this Java technology allows high-quality 3D graphics to be implemented in a platform-independent manner on low-cost personal computers. Schematic Java code for drawing a space-filling model is given in Table 1.

Table 1. Schematic Java code for drawing a space-filling model

```
public class DisplayMolecule extends Panel{
   public BranchGroup createSceneGraph(SimpleUniverse u) {
      TransformGroup tg, tr;
      Transform3D td;
      Material m;
      Appearance a;
      // Create the root of the branch graph
      BranchGroup objRoot = new BranchGroup();
      // Create a Transformgroup to scale all objects so they
      // appear in the scene.
      TransformGroup objScale = new TransformGroup();
      Transform3D t3d = new Transform3D();
      objScale.setTransform(t3d);
      objRoot.addChild(objScale);
      TransformGroup objTrans = new TransformGroup();
      objScale.addChild(objTrans);
      // Create Atoms and add them into the scene graph.
        x[i],y[i] and
      // z[i] are cartesian coordinates of ith atom.
         for(int i=0; i< number of Atoms; i++) {
         tg = new TransformGroup();
         objTrans.addChild(tg);
         td = new Transform3D();
         Vector3d p =
            New Vector3d(x[i],y[i],z[i]);
         td.set(p);
         tr = new TransformGroup(td);
         tg.addChild(tr);
         m = new Material(
            ambientColor[atomicNumber[i]],
            emissiveColor[atomicNumber[i]],
            diffusedColor[atomicNumber[i]],
            specularColor[atomicNumber[i]],
            100.0f);
         a = new Appearance();
         a.setMaterial(m);
         Sphere sph = new Sphere(radiusOfAtom[atomic
  Number[i]],
            Sphere.GENERATE_NORMALS,
  qualityOfGraphics, a); tr.addChild(sph);
}
   return objRoot;
}
public DisplayMolecule() {
   Canvas3D c = new Canvas3D(null);
   add ("Center", c);
   SimpleUniverse u = new SimpleUniverse(c);
   BranchGroup scene = createSceneGraph(u);
   u.getViewingPlatform().setNominalViewingTransform();
   u.addBranchGraph(scene);
   }
}
```

THE PROGRAM DEVELOPMENT ENVIRONMENT

We used the Java Development Kit (JDK) version 1.2 beta 4 for the development of MoldaNet. The user interface design was accomplished using JBuilder 2 provided by Inprize (Borland). The resulting applets were given digital object signed certificates as previously described.[6] We note that to be fully functional, MoldaNet needs to operate with a Java Virtual

environment, which also supports version 1.2 of the JDK for the Java3D features. The released versions of commercial browsers such as Netscape (version 4.5) or Internet Explorer (version 4.01) at the time of writing did not support the Java3D features, and as a temporary measure, we tested MoldNet using the local AppletViewer provided with JDK version 1.2 beta 4. We anticipate that JDK version 1.2 will be integrated with the release of new browsers.

RESULTS AND DISCUSSION

Overview of molecular modelling functions in MoldaNet

When the URL for MoldaNet (http:/origin.ch.ic.ac.uk/molda/) is accessed via a WWW browser, the following actions are invoked.

1. A copy of the MoldaNet applet byte code is transferred from the server to the browser, and loaded into the Java Virtual engine. If a Java "Just-in-Time" or JIT compiler is installed, the Java applet byte code will be translated to machine-specific instructions for optimal local execution speeds.
2. The browser will at this stage detect that a digital certificate is included within the applet, and will invoke a dialog to request that the user grant additional security permission to the applet as shown in Figure 1. At this stage, the user can request to see a copy of the digital certificate as issued by the Certification Authority (CA). It is presumed at this stage that the user will trust the credentials of the CA, and hence by proxy, the authenticity of the person or organisation in whose name the certificate has been signed. If the certificate is accepted by the user, it is entered into the certificate database created by the Browser on the local file store. The user has the option of indicating that if this certificate is found to precisely match identical certificates downloaded in future, the security permissions requested may be granted automatically without the need for any dialogs. In this manner, the user can build up a database of digital certificates from various sources.
3. We note at this stage that it is also possible for the MoldaNet server, prior to downloading the MoldaNet applet to the client, to request if the client also has a digital certificate, referred to as a *Class 2 X.509-based certificate*. This would serve to authenticate the client to the server, while the process described above serves to authenticate the server-based applet to the client. By means of an access control list, it would be possible to arrange that only clients with appropriately recognised Class 2 digital certificates could actually access the server. This provides a powerful alternative to the more conventional server access control mechanism of issuing login accounts and passwords.
4. Once the client user has granted permission, the standard MoldaNet menus (summarised in Table 2) are displayed on the screen (Figure 2). A brief description of the menus follows here.

File menu The File menu deals with molecular coordinate file input and output. The coordinates of fundamental organic compounds, such as aliphatic and aromatic molecules, amino acids, nucleotides, and sugars, are provided as templates. Molda,[2,3] XMol XYZ,[12] and Protein Data Bank[13] format files can be read-and-write. Molecular coordinates can also be saved in CML and VRML 1.0/97 file formats. The molecular coordinates created by MoldaNet can be transformed into the input files of several computational chemistry packages, such as TINKER,[14] MOPAC,[8] and GAUSSIAN.[9] The output structures obtained by these computational chemistry packages can be imported to MoldaNet.

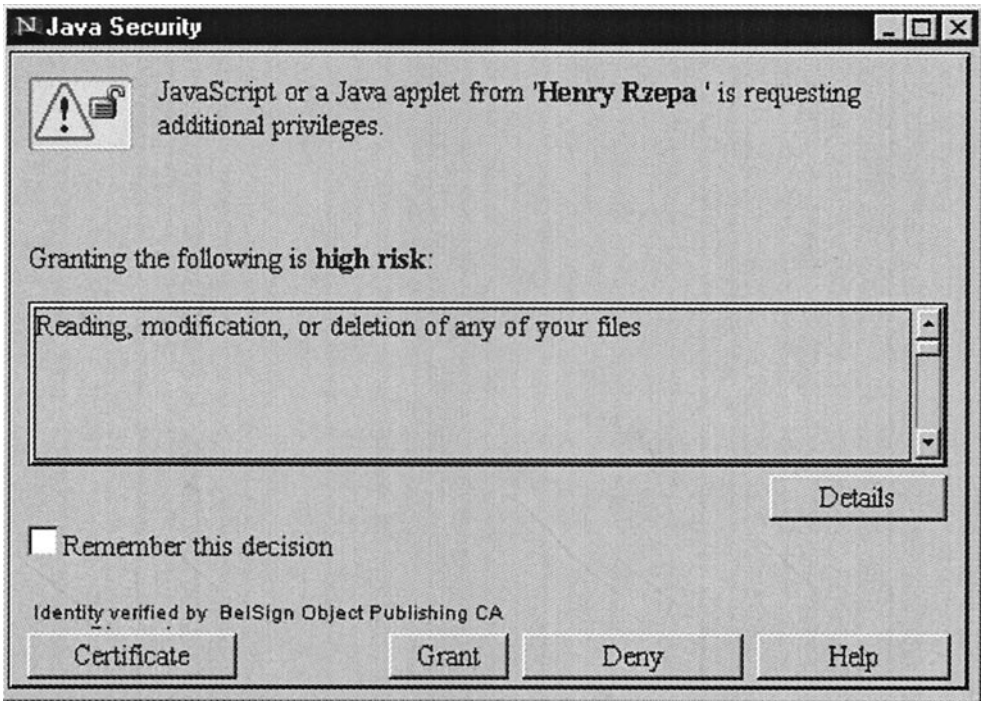

Figure 1. A certificate dialog that appears when MoldaNet is accessed via a WWW browser.

Table 2. Menus/submenus in MoldaNet

Menu	Submenu 1	Submenu 2	Description
File	Open		Read molecular structure data in Molda file format
	Templates		Read fundamental organic and biological compounds
	Save		Save molecular structure data in Molda file format
	Save as		Save molecular structure data in Molda file format as specified file name
	Save as a Substituent		Save molecular structure data as a substituent
	Save in VRML		Save molecular structure data in VRML1.0/2.0 (Dreiding sticks, ball-and-stick and space-filling models are available)
	Import	Protein Data Bank	Import PDB files
		CML	Import CML files
		XMol	Import MSC XMol files
		TINKER	Read the TINKER output coordinates
		MOPAC7	Read the MOPAC output coordinates
		Gaussian98	Read the GAUSSIAN output coordinates
	Export	Protein Data Bank	Export PDB files
		CML	Export CML files
		XMol	Export MSC XMol files
		TINKER	Make the TINKER input file
		MOPAC7	Make the MOPAC input file
		Gaussian98	Make the GAUSSIAN input file
	Exit		Exit MoldaNet
View	Show Axes		Show *x* and *y* axes
	Show Atom Number		Locate the number on each atom
	Auto Scaling		Scale the size of molecule automatically
	Translation Enable		Enable the translation of a molecule by the mouse operation
	Change Structure Automatically		Change bond lengths automatically when a specified atom is replaced by another element
	Generate Bonds Automatically		Generate bonds automatically when molecular coordinates are imported
Model	Input	Atom	Input the number of atoms, the Cartesian coordinate, and the atomic numbers
		Bond	Generate a bond between specified two atoms
		Alkane	Make the all-*trans* conformation of an *n*-alkane molecule by input of the number of carbon atoms
		Hydrogen	Add hydrogen atoms to the carbon atoms
		Element	Replace a specified atom by another element
	Delete	Atom	Delete a specified atom
		Bond	Delete a specified bond
		Component	Delete a specified group of atoms
	Merge	Molecule	Replace an atom or group of atoms of the molecule by another molecule
		Substituent	Substitute an atom with a common substituent or registered substituents
	Change	Atom Coordinate	Change atom coordinate by the mouse operation
		Bond Length	Change bond length by the mouse operation
		Bond Angle	Change bond angle by the mouse operation
		Dihedral Angle	Change dihedral angle by the mouse operation
	Move		Move a molecule by the specification of atoms or cartesian coordinates
Display	3D Graphics Mode		Draw high-quality 3D molecular graphics by the use of Java 3D APIs
	Refresh		Refresh the screen

Table 2. Continued

Menu	Submenu 1	Submenu 2	Description
	Preferences		Set the model type (ball-and-stick or space-filling), the quality of graphics, the colour of light and the direction of a light
Analyze	Coordinate		Give coordinates of a specified atom
	Distance		Calculate distance between two specified atoms
	Angle	Calculate bond angle	
	Dihedral Angle	Calculate dihedral angle	
Help	About MOLDA		Show information about this program

View menu The View menu deals with changes in screen appearance (such as showing *x* and *y* axes and locating the number on each atom).

Model menu The Model menu deals with the construction of molecular models. The all-*trans* conformation of an *n*-alkane molecule can be generated in a one-step operation by using the [Input]-[Alkane] menu. A specified atom may be replaced by another element by using the [Input]-[Element] menu. Any molecules made by Molda can be connected with each other using two modes of merging. The [Merge]-[Substitution] menu enables connection with common substituents or registered substituents by users. Two molecules can be connected by the use of the [Merge]-[Molecule] menu, as shown in Figure 2. Atom positions, bond lengths, bond angles, and dihedral angles can be easily changed by mouse operation.

Display menu The Display menu deals with high-quality

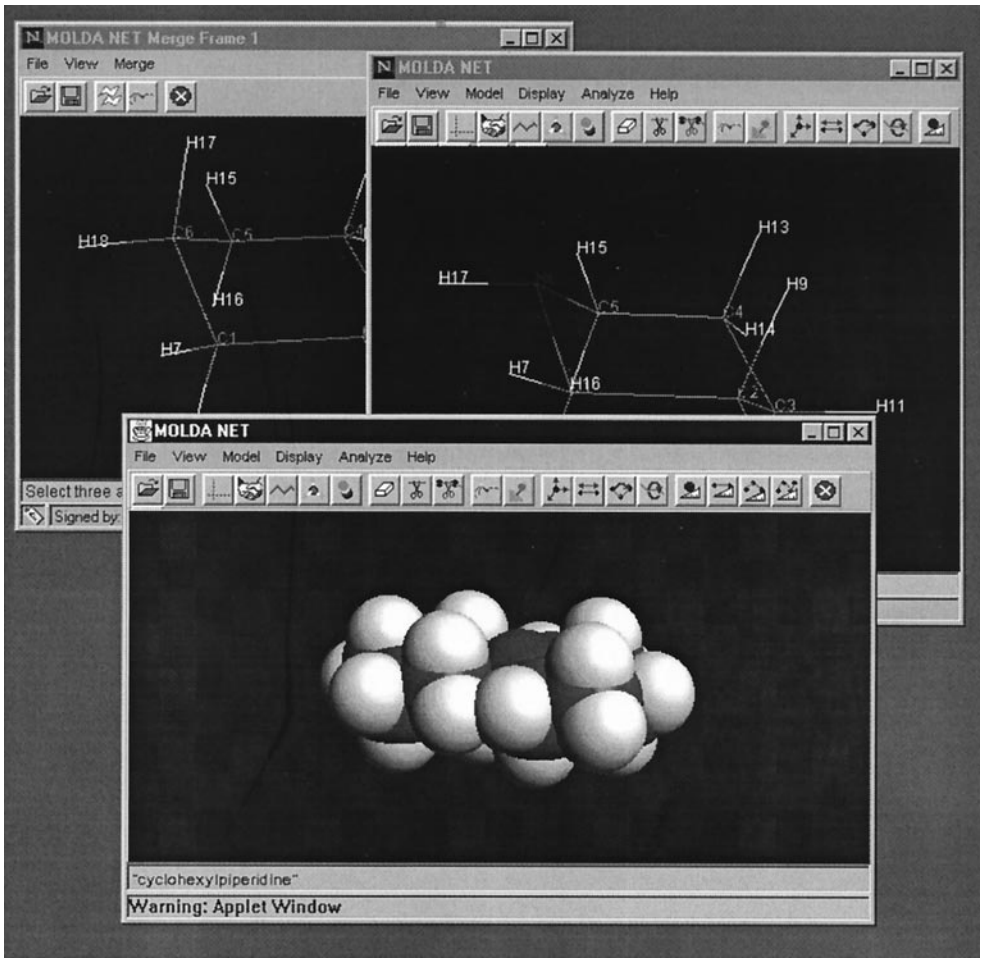

Figure 2. A menu screen of MoldaNet, illustrating the combining of two molecules by the use of the [Model]-[Merge]-[Molecule] menu, and a space-filling model drawn using the Java 3D API.

3D molecular graphics using the Java 3D API classes. Details are given below.

Analysis menu The Analysis menu deals with the calculation of bond distances, bond angles, and dihedral angles of a molecule.

3D molecular graphics by the use of Java 3D API and VRML viewing

The Java 3D API is a new technology for creating high-quality 3D models on a computer network and intended for the creation of integrated multimedia resources. Space-filling and ball-and-stick models (Figure 2) are available in MoldaNet by using the Java 3D API classes. The quality of graphics, the colour of a light, and the position of a light can be defined by the user. MoldaNet can also be used to generate molecular structures in VRML1.0/97 formats. This means that any molecular structure created by MoldaNet or converted from popular computational chemistry packages can be distributed on the computer network and viewed in 3D by using a VRML viewer. To render 3D molecular models in VRML, an appropriate VRML viewer such as CosmoPlayer must be installed as a plug-in to the Web browser.[7] Use of MoldaNet, however, via the use of the Java 3D API, enables 3D molecular graphics to be implemented in a distributed manner on a computer network without the need for the user to pre-install any external applications or plug-ins.

CONCLUSIONS

Molecular graphics and modelling systems developed during the 1970s almost invariably ran on expensive mainframe computers and minicomputers. In 1984, Molda was developed to show the potential of stand-alone/on-line low-cost personal computers (PCs) for molecular modelling.[3] Molda was the first in a long line of personal computer-based molecular graphics and modelling programs to introduce the techniques of molecular modelling to nonspecialist chemists in the 1990s, admittedly in a platform-specific manner. In the present work, *MoldaNet* is presented as the next evolutionary step in this process, using a WWW browser paradigm based on using Internet, Intranet, and Extranet models. By using a Java-enabled WWW browser, MoldaNet runs platform independently and does not require any pre-installation of software by the user. This means that molecular modelling and high-quality 3D molecular graphics can be performed easily on the well-known platforms such as Windows, Macintosh, and UNIX, which have high-speed network connections, and in particular that these tools can be implemented in a distributed manner with relatively low software maintenance requirements. The new paradigm represents a new type of computational tool for application in chemistry, chemical engineering, and chemical education. It certainly offers an alternative approach to the single platform/single operating system vendor model, and represents a move toward object-oriented modular and interoperable chemistry tools, which can take full advantage of the global Internet.

ACKNOWLEDGMENTS

We thank the JISC (UK) for their award of a JTAP grant, Hiroshima University Venture Business Laboratory for travel funding for H.Y., and help from Christian Buysschaert of GlobalSign in implementing digital object certificates.

REFERENCES

1 Rzepa, H.S. Internet-based computational chemistry tools. In: *The Encyclopaedia of Computational Chemistry* (Schleyer, P.V.R., Allinger, N.L., Clark, T., Gasteiger, J., Kollman, P.A., Schaefer, H.F., III, and Schreiner, P.R., eds. John Wiley & Sons, Chichester, 1998, pp. 1426–1436
2 Yoshida, H., and Matsuura, H. *J. Chem. Software* 1998, **4**(3), 81–88
3 Ogawa, K., Yoshida, H., and Suzuki, H. *J. Mol. Graphics* 1984, **2**(4), 113–116; Yoshida, H., and Matsuura, H. *J. Chem. Software* 1997, **3**(4), 147–156
4 Sayle, R.A., and Milnerwhite, E.J., *Trends Biochemical Sci*. 1995, **20,** 374–376
5. Sameshima, Y., and Kirstein, P.T. *Comput. Networks ISDN Syst.*, 1996, **28,** 513–523
6. Tonge, A.P., Rzepa, H.S., and Yoshida, H. *J. Chem. Inf. Comput. Sci.* 1999 (May Issue)
7. Casher, O., Page, C.S., Leach, C., and Rzepa, H.S. *Chem. Br.* 1998, **34**(9), 26–30
8. Murray-Rust, P. Chemical markup language, a simple introduction to structured documents. In: *XML, Principles, Tools and Techniques* (Connolly, D., ed.) O'Reilly, 1997, pp. 135–149
9. Stewart, J.J.P. MOPAC: A general molecular orbital package (version 6.0). *Quantum Chemistry Program Exchange* **455**.
10. Keith, C.T., Petersson, G.A., Montgomery, J.A., Raghavachari, K., Al-Laham, M.A., Zakrzewski, V.G., Ortiz, J.V., Foresman, J.B., Cioslowski, J., Stefanov, B.B., Nanayakkara, A., Challacombe, M., Peng, C.Y., Ayala, P.Y., Chen, W., Wong, M.W., Andres, J.L., Replogle, E.S., Gomperts, R., Martin, R.L., Fox, D.J., Binkley, J.S., Defrees, D.J., Baker, J., Stewart, J.P., Head-Gordon, M., Gonzalez, C., and Pople, J.A. *Gaussian 94* (revision D.3). Gaussian, Pittsburgh, PA, 1995
11. Krassavine, A. *Chimia* 1998 **52,** 669
12. Minnesota Supercomputer Center. *XMol User Guide* (version 1.9). http://www.msc.edu/msc/docs/xmol/
13. Protein Data Bank, Brookhaven National Laboratory, http://pdb.pdb.bnl.gov/
14. Ponder, J.W. *TINKER: Software Tools for Molecular Design*. http://dasher.wustl.edu/tinker/