EyeChem 1.0: A modular chemistry toolkit for collaborative molecular visualization

Omer Casher, Stuart M. Green and Henry S. Rzepa

Department of Chemistry, Imperial College, London, UK

EyeChem is a network-aware and modular molecular visualization toolkit used within the Iris Explorer Visualization program. Use of the toolkit is illustrated via four typical EyeChem applications, which consist of Explorer maps constructed from chemically oriented modules and developed for use over the new generation of fast networks such as the UK SuperJanet system. EyeChem can also be used to prepare multimedia style visualizations in Quicktime or MPEG format of molecular properties and wavefunctions for archiving via the gopher+ wide area information system. The use of such information in electronic publishing of chemical information is discussed.

Keywords: molecular visualization, networking, explorer, EyeChem, SuperJanet, FDDI

The use of computers for chemical visualization and modeling has been significantly enabled recently by the improved performance and decreased cost of the hardware required and the availability of a wide range of commercial software. However, the cost of developing, documenting, and maintaining highly complex modern software packages is increasing, often proving more expensive than the hardware it is implemented on. It is also probable that the average user makes use of only a small fraction of the capabilities of any one program suite. Molecular builders or editors, for example, are common to virtually all modeling programs, a duplication which does nothing to decrease software costs. Indeed, the lack of common guidelines in this area results in additional requirement for the provision of numerous types of file translators and additional training. As the visualization tools increase in complexity, the ability to customize these packages toward less commercially viable or innovative areas decreases, and new program development becomes more difficult for end users, who have to invest time in user interfaces and visual rendering at the expense of algorithmic development.

Such problems, not uncommon in other scientific disciplines, have been addressed through visualization systems such as AVS or IRIS Explorer.[1] These provide an environment for the development and implementation of modular toolkits as an alternative to complete packages. Use of such tools largely eliminates the task of having to write hardware specific code for visualization primitives or the graphical user interface, and also greatly facilitates the customizing of applications. One such modular toolkit is the commercially available AVS Chemistry Viewer designed for use with the AVS system.[2]

THE SuperJanet PROJECT

The widespread adoption of the global TCP/IP Internet network protocol has enabled most types of workstations and personal computer to be integrated with minimal end user cost. In the UK, the SuperJanet pilot network project was started in June 1992 with the objective of providing high bandwidth communications at an initial 34 Mbit/sec, interconnecting six major academic sites in the UK, subsequently extending to the rest of the UK and to international links. At the start of the project it became apparent that there was little commercially available software which fulfilled our own requirements and would make full use of this network. Our first need was to provide readily modifiable interfaces to a number of existing program systems or databases such as MOPAC, Gaussian 92, ADF, MacroModel or Sybyl, the Brookhaven protein structure databank, and the Cambridge crystal structure depository. Second, all the modules had to be "network aware," preferably on a "peer-to-peer" rather than "client–server" relationship. We had come to regard the peer-to-peer, or "multicasting," approach as essential for conveying convincing experimental or theoretical results to distant colleagues in real-time, as for example in discussing the properties of a molecular wavefunction giving rise to a subtle three-dimensional (3D) stereoelectronic effect.[3] To this end, two or more users must be able to manipulate or even edit molecular images in real-time, preferably with audio and visual links established to facilitate the process. Our final objective was to provide a facile means of converting animated molecular information into a permanent supplement to the conventional printed page publishing medium, as a

Color plates for this article are on page 199.

Address reprint requests to Dr. Rzepa at the Department of Chemistry, Imperial College, London SW7 2AY, UK, e-mail: rzepa@ic.ac.uk.

Received 18 October 1993; accepted 2 November 1993.

means of retaining the impact and meaning of a complex 3D molecular structure or wavefunction.

EXPLORER

Explorer implements a library of graphical objects to create interactive 3D graphics applications. The IRIS Explorer visualization system removes from the programmer the need to address the complex issue of 3D graphical rendering. Rendering is based on the IRIS Inventor toolkit, which enables the user to invoke actions such as picking, shading, or rotation. Explorer also provides animating capabilities and the network interfaces for remote workstation rendering. As with AVS, Explorer comes with an extensive range of modules covering the more visual scientific and engineering disciplines. Additional modules, many of which are contributed by user groups, are also available from public repositories.[1]

THE EYECHEM SUITE

Since no readily available molecular modeling software had remote viewing or animation capabilities for the types of molecular properties of interest to us, we chose to develop what we called the *EyeChem* suite, using C and IRIS Explorer. Although Explorer is well equipped for remote rendering and animation, only a small number of chemistry modules together with their public domain source code were available at the start of our project. We also had access to existing code developed in-house such as our Intercon routines.[4] The modules EyeSybyl, EyeESP, and EyeMonteCarlo are direct implementations of such routines. The complete EyeChem suite provides visualization capability for various calculated molecular properties, such as geometries, progress in geometry optimization, molecular orbitals, electrostatic potentials, and a ribbon rendering option for protein structures.

To the end user, the creation of applications is straightforward, provided all the required modules are available. In a predetermined set of connected modules, known as maps, data appear to flow from module to module in a flow diagram fashion until a new 3D image is created. Most modules are provided within the Explorer environment, and others are readily implemented from existing source code, while some are programmed from scratch using published algorithms, as for example the module to generate a ribbon or thread representation of a protein structure. The primary advantage of modular code is reusability and reliability, and various graphical representations can be generated from the same data file by proper arrangement and connections of selected modules. The minimal user action in using such maps is to provide the name of an input file of coordinates or wavefunctions and the manipulation of the resulting 3D image. A slightly more enterprising user can connect the modules themselves, and hence create a new map which might be of general use. In this manner, an adequate set of appropriately chemically oriented modules provides the capability for considerable user customization. In this paper we will attempt to illustrate these themes by means of four EyeChem interfaces.

Discussion

Each EyeChem application was composed from modules selected from the Librarian (Color Plate 1), placed on the *map editor* and wired together such that the output from one module serves as the input for another. The end point is normally a rendering module which performs the complex task of mapping a complex function in 3D space, applying lighting, providing rotation and zoom, and allowing various rendering options to be changed if desired.

Map 1: MOPAC-93 and Gaussian 92

This interface (Color Plate 2) was built to monitor the progress of a MOPAC-93 or Gaussian 92 calculation by reading a log file written to disk. Since such calculations can take several hours or even days, any aberration or abnormal behavior needs to be quickly identified. This interface will enable this by displaying a ball-and-stick geometry, if necessary as an animation of each cycle in the geometry optimization. Calculated properties of the final optimized molecule can be also read from disk files, such as the molecular electrostatic potential (MEP), molecular orbitals, or electron densities (MOPAC only). As an example of how such a system can be readily customized, we needed to calculate bounded volume and related properties within which MEP values had been calculated. A separate program to perform a Monte Carlo integration had previously been written. This module was readily incorporated into the library, and could be easily integrated into a suitable map.

The component modules shown in Color Plate 2 comprise the log file reader *EyeMopac*, which reads a standard MOPAC (V 5, 6 or 93) output file and a separate file of atomic properties such as atomic radii for all atoms. If the output from MOPAC is subsequently changed in new versions, only this module needs to be modified. There is also, in principle, no reason why MOPAC itself could not be compiled and added as a module. EyeMopac outputs an Explorer molecular pyramid data type, which contains all the atom and bond locations and other properties that might be needed by downstream models. *EyeBalls* is a geometry module which inputs a molecular pyramid data type from EyeMopac, and outputs an Explorer geometry data type containing all the information necessary for a ball-and-stick 3D image. It has default color settings but also accepts an optional "color map" with user-selected color information. *EyeESP* is another reader which inputs information from a MOPAC electrostatic potential file, modified locally in our case to generate a uniform grid of a selected spacing. EyeESP will interpolate missing points and then output all the points as a uniform Explorer lattice data type. Here the lattice is fed to two *IsosurfaceLat* modules which perform the task of contouring the data at two given thresholds and convert the Explorer lattices to Explorer geometries.

Properties of a module that can be readily changed (file names, radii, etc.) are either controlled by widgets within the module interface, or can receive a parameter from any other module. The threshold of IsosurfaceLat for example is controlled by a dial widget, the value of which also passed to IsosurfaceLat<2> and converted to the negative value, in this case appropriate for electrostatic potentials which can be either positive or negative. *EyeMonteCarlo* inputs an Explorer lattice and sends to the standard output the volume and other properties of a selected volume element bounded by a

Table 1. Summary of EyeChem Modules

EyeMopac	Inputs a MOPAC style output file and outputs a molecular pyramid datatype
EyeG92	Inputs a Gaussian92 style output file and outputs a molecular pyramid datatype, with animated display of the steps in geometry optimization cycles
EyeMopacVib	Input a MOPAC output containing a FORCE calculation and animates selected normal vibrational modes
EyeMopacRxn	Inputs a MOPAC output file containing reaction coordinate calculations and animates them
EyeESP	Inputs a MOPAC.grd file containing a grid of electrostatic potential values and a uniform explorer lattice
EyeBalls	Inputs a molecular pyramid datatype and outputs Explorer geometry information for spheres, cylinders and lines
EyeSybyl	Inputs a Sybyl.mol file and outputs a molecular pyramid datatype
EyeBackbone	Inputs a pdb file and outputs a molecular pyramid datatype of only the peptide backbone atoms
EyeRibbons	Inputs a molecular pyramid datatype and calculates guide coordinates, then outputs geometry information for non-uniform rational B-splines (threads) and Bezier surfaces (ribbons)
EyeCrystal	Inputs a cssr formatted crystal data file and outputs a molecular pyramid datatype
EyeMonteCarlo	Inputs a uniform Explorer lattice file and calculates the volume and related properties of a selected bounded surface component

given 3D contour value and calculated using Monte Carlo integration. This module outputs the bounding box coordinates of an Explorer lattice to the geometry module *WireFrame* which, in turn, outputs an Explorer geometry of this bounding box. The *Render* module accepts Explorer geometries and displays the resulting 3D graphics image within its window. Many properties of the image such as orientation, size, lighting, transparency, and colors can be varied by the user using controls provided within the render window. Replacing EyeESP by the MopacView module would result in molecular orbitals or electron densities being displayed. This latter module is an implementation of the DENSITY program distributed with MOPAC-93.[5]

Map 2: Protein structures

The increasing interest in the docking of small molecules, inhibitors, and other biologically active systems with larger biomolecules such as proteins and enzymes requires efficient and seamless interfacing with database files. These themes are illustrated in Color Plate 3 in which the module *EyeCrystal* reads crystallographic coordinates and outputs an Explorer molecular pyramid to EyeBalls. *EyeBackbone* is another reader that extracts information of only those atoms that form the peptide backbone (N, Cα, C, O) in a standard Brookhaven PDB file format. If the coordinates of all the atoms are required, *ReadPDB* could be used in place of EyeBackbone. *EyeRibbons* receives the molecular pyramid output from EyeBackbone and outputs the geometry of the non-uniform rational B-Splines (NURBS) if *Thread* is selected from the switchbox widget, or a NURBS patch surface if *Ribbon* is selected. The algorithms for doing this have been published by Carson.[6] EyeRibbons has a default (white) color setting but, like EyeBalls, can accept an optional color map lattice, in this case from *GenerateColormap*. The lattice of GenerateColormap, set to create the Spanish flag effect of the threads, was previously saved to a file by connecting it to the *WriteLat* module. This lattice file was read by *ReadLat* which output the lattice to GenerateColormap. Render accepts the two molecular geometries and the thread or ribbon, and displays them in superimposed form. One area where the current limitations of rendering hardware is quickly encountered is the visualization of large molecules in Explorer, which is noticeably slow, even using hardware such as Indigo[2] with Extreme graphics. With ribbons and rendered spacefills (spheres) of large molecules, the display problem is not viable. One solution was reducing the polytube representation of the molecule to mere lines, as is done in EyeBalls. Another solution is modifying the texturing of the rendering via the Inventor toolkit.

Map 3: Collaborative remote rendering

Explorer has built in remote rendering capabilities to permit the exporting of 3D molecular images, enabling the simultaneous manipulation and rotation of the molecule at one end and viewing of the actual rotation at the other end in more or less real time. Other features such as *remote picking* (adding or removing an atom or bond at one end and seeing the change at the other) are also possible. This is illustrated in an Explorer map (Color Plate 4) for simultaneous viewing and manipulating of the same molecular system by two or more collaborators seated at different computers and linked through a high speed network. *EyeG92* is a reader similar to EyeMopac but inputs a Gaussian92 log file, which enables all the participants to view a selected cycle in a geometry optimization via a widget control, or to "visualize" how the calculation is progressing in the form of an animation at about 1–2 frames/second. Since Gaussian calculations can be expensive in terms of computer time, such interactive feedback can be essential for checking the correctness of the calculation.

EyeBalls in this case is connected to two local and two *RemoteRenders*. The latter require input of the Internet address of the remote station, either as a name (e.g., argon-fddi.ch.ic.ac.uk) or the equivalent numerical form. When a molecule is rotated or in some way

manipulated at one location, the new camera geometry of Render is sent to the equivalent Render at the other, which in turn implements the change. The connections between Render and RemoteRender<2> and between RemoteRender and Render<2> shown in Color Plate 4 mean that each of the two participants has one window which reflects changes they have made, and one window which reflects changes the remote user has made.

The overall responsiveness of this configuration depends largely on the speed of the interconnecting network, and the local rendering speeds. Our initial trials used the existing UK national network in which the maximum end-to-end data transfer speed was 2 Mbits/second. Since a very considerable part of this bandwidth is occupied by other users, the responsiveness could often be very poor; i.e., a significant latency was experienced. Any attempt to augment our configuration via audio and videoconferencing proved quite impractical. Subsequently, trials were conducted using the 34-Mbit/sec SuperJanet connection via an FDDI workstation card leading to a CISCO FDDI router. This enabled a round cycle time of $\approx$1 s for a local-remote-local render window to be achieved. This is probably limited more by the CPU rendering time rather than the network response. The additional bandwidth also enabled a parallel audio and videoconferencing session, using in our case the public domain software IVS V3.2.[7] In the future, any remaining latency problems should be eliminated by using ATM style interfaces, which hold the promise of point-to-point connection of 150 Mbit/sec or greater.

A multimedia presentation in Quicktime format illustrating the use of this map is available on the Gopher+ server *argon-fddi.ch.ic.ac.uk* in the SuperJanet directory. Gopher clients such as the Macintosh specific Turbogopher or the Windows specific hGopher will recognize these files as Quicktime movies, and following transfer of the file to the users local disk will automatically invoke suitable movie players.

Map 4: Molecular animation for digital media

In this section, we focus on the problem of permanently recording visual information produced using EyeChem modules. Conventional forms of hardcopy, such as Postscript files, are trivially produced by including *snapshot* modules in the Explorer maps. These however are static two-dimensional representations of the molecular properties, and the remaining spatial and time dimensions are lost in this process. While it is possible to recreate at least the spatial dimension by interchanging molecular coordinate files,[8] more complex visual images such as orbitals, or any animation component cannot be so distributed. We proposed instead to adopt the Quicktime multimedia standard introduced two years ago by Apple, and available now for both Apple and DOS/Windows personal computers. This allows both motion and an audio soundtrack to be archived. In parallel, the development of wide area information servers has gathered pace, and one of these protocols, known as Gopher+, already supports multimedia file types such as Quicktime or MPEG video formats.[9] Gopher represents a highly distributed database system, in which a number of geographically distributed servers can be logically connected to a client system using configuration files with entries known as *Bookmarks*. The attraction to the chemical community is that customized chemical bookmarks are readily created and distributed to provide highly focused information to end users, without necessarily inhibiting the serendipitous element of speculative browsing. Furthermore, since both Macintosh and DOS based systems as well as Unix workstations can readily serve as both Gopher servers and clients, the end user cost of participating in such a distributed information system is minimal.

Color Plate 5 shows a typical EyeChem map where a molecular vibration is being animated. The output from a rendering module is input into an animation module. Currently, that module only generates one file that is a sequence of SGI FIT (MovieMaker) images. After several subsequent conversions bit-mapped files in *pict* format are transferred to other computers, where they are compressed and edited into a *Quicktime* multimedia sequence, and where still captions, audio soundtrack and other features can be added using editing suites such as Adobe Premiere. The final production can be archived on a Gopher+ server, where anyone with Internet access and the gopher client software can access the product, and if desired incorporate it into their own multimedia presentations. A number of such animations have been created as supplemental material for our own publications and archived on the Gopher+ server *argon-fddi.ch.ic.ac.uk*. A description of how to visualize such material, together with appropriate programs is available from the same source. The EyeChem executable modules and maps are available on the Iris 5.2 CD-ROM, available from SGI Inc. Enquiries about the source code should be made directly to the authors.

CONCLUSIONS

The EyeChem modules and the corresponding maps represent about 18 man-months of effort, and can be regarded as a pilot project in how visualization and analysis software can be developed in a manner which readily takes advantages of future hardware developments while keeping development time and costs to a minimum. The success of such concepts depends on the ease with which other chemistry modules can be written and contributed by the global chemical community. In particular, we emphasize how the increasing speed of communications networks enables novel concepts such as long-distance discussion of 3D molecular information to be enabled, and how in turn the availability of such information requires entirely new concepts to be developed in scientific publishing.

ACKNOWLEDGMENTS

We thank the JISC (UK) for equipment and SGI for a bursary (to OC). We also thank Drs. Gray Griffin (SGI Mountain View) and Guillermo Suner for valuable discussions.

REFERENCES

1 IRIS Explorer Center (Europe), P.O. Box 50, Oxford OX2 8JU
2 Upson, C. *IEEE Computer Graphics & Applications.* 1989, **4,** 30
3 Halton, B., Boese R., and Rzepa, H.S., *J. Chem. Soc., Perkin Transactions 2.* 1992, 447; Baird, M.S., Al Dulayymi, J.R., Rzepa H.S., and

Thoss, V. *J. Chem. Soc., Chem. Commun.* 1992, 1323; Casher O., O'Hagan D., Rosenkranz, C.A., and Zaidi, N.A.; *ibid.* 1993, 1337
4 Green, S. and Rzepa H.S. *Quantum Chemistry Program Exchange Bulletin.* 1990, **10,** Program 598
5 MOPAC-93: J.J.P. Stewart, Fujitsu Limited, Tokyo, Japan (1993). Available from Quantum Chemistry Program Exchange, University of Indiana, Bloomington, Indiana
6 Carson, M. and Bugg, C.E. *J. Mol. Graphics.* 1986, **4,** 121; *ibid.* 1987, **6,** 103
7 Turletti, T. Project RODEO, INRIA Sophia Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia Antipolis, France
8 Protein Science, Cambridge University Press and the associated program Mage; V 2.4, ProSci @u.washington.edu
9 Johnson, D., Anklesaria, F., Tonske, H., and MacMahill, M., University of Minnesota, gopher@boombox.micro.umn.edu