A New Plug-in Software Architecture Applied for a Portable Molecular Structure Browser

Yutaka Ueno and **Kiyoshi Asai**
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, 305 Japan
{ueno, asai} @etl.go.jp

Abstract

A new software configuration method using plug-in style components was established for the tool with the incremental development of software used in protein structural study. Our memory database provides the interface of data and functions among plug-in modules and its host program. A molecular structure browser program was developed together with several plug-in modules on our programming library that maintains graphics portability and user @nterfaces. This plug-in software architecture is generally useful for large-scale software development and for prototyping parts of the system.

Keywords: Molecular Graphics, Component Software, Modeling, Protein Structure

Introduction

Molecular structure visualization for the analysis and modeling of biopolymers has recently been innovated by computer graphics technology. During current progress in structural biology, the use of software is fundamental not only for applying available computational procedures to materials of biological interest, but also for prototyping novel methods and calculations with the atomic coordinates of the molecule.

However, most programs for molecular graphics are difficult to modify for additional calculations and customized functionality. Tools for scientific visualization, such as AVS (Upson et al. 1989), Data Explorer (Abram and Treinish 1995), which were made up of component software, are also used in calculating three-dimensional properties of molecules, such as the electrostatic potential. Although these tools are weighted with numerous functions for various fields they have not been employed for advanced computation of atomic structure in bioscience.

While most graphics software depends on acceleration hardware, RasMol (Sayle 1995), a comprehensive protein structure browser, renders molecules in reasonable speed on workstations and personal computers without optional hardware and is accepted among a wide range of researchers. However, modifying a program which differentiates into mock-ups prevents sharing and integration of programs. There is thus a demand for a portable programming environment in computational molecular biology.

We have initiated a new software-development project for protein structure analysis with incremental extension capabilities. Our goal is to provide a software platform which runs on common hardware and allows users to add new functions using only average programming skills. We have designed our software system with respect to two issues, extensibility and portability. This paper describes our simple plug-in software architecture and its application to a molecular structure browser.

Furthermore, the molecular graphics is a medium to demonstrate specific atomic mechanisms of proteins. Through combined use of interactive rotation and comparison of structures by a software tool, one can understand the molecular architecture more directly and draw considerable inferential insight into them. While Kinemage (Richardson and Richardson 1992) provides illustrations for the published articles, our software tool should also be refined to the extent that it will be accepted for these purpose.

System Design

Plug-ins for Extensibility

To provide the desired flexible extensibility, we adapted the dynamic linking for a plug-in style module integration. The shared dynamic library has been implemented on various operating systems, mainly to economize memory consumption. The linked modules share the thread of execution and address space with the host program. Recently, these kinds of plug-in modules have been employed successfully in application systems, such as Photoshop (Adobe Systems, Inc.) or Netscape (Netscape Communications, Inc.). However the interface method for the plug-in module differs for each application and operating system. We propose here a generic application program interface (API) for the plug-in software component using a memory database.

Memory Database. Some memory contents shared among program modules are named with ascii strings as keys for retrieval. Our memory database maintains them by simply prefixing the key name to their

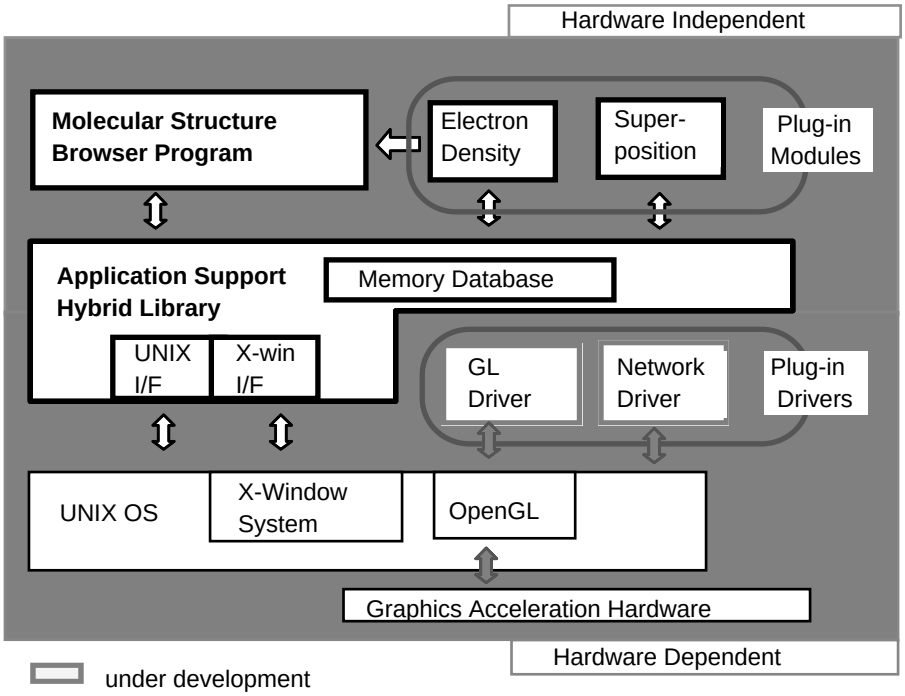

Fig.1. Software Configuration. An application and plug-ins built on the application support hybrid libray. The plug-in Drivers interface to the specific hardware.

content. The key name is used as an alternative to a memory handle, which is an address pointer. Data is created sequentially in a list where the key name is unique inside the list. There is the *order* list that enumerates the address of functions as registered software services to be called by their key names.

A plug-in module is built by an object file linker of the operating system as a shared library. When a plug-in module is loaded into the host program, the function Plugin() is called instead of main(). It deposits data and functions into the memory database to be referred from other modules. CreateBean() creates a datum in a list on memory database with key name and data size, whose address is retrieved by Fetch(). TakeOrder() registers an address of function with its key name as an *order*, which is called by GiveOrder() with arguments.

Symbols of data and functions in a host program are referred to in plug-in modules as if they are linked, but the symbols in a plug-in are only visible inside the plug-in. Since direct finding of the address of a symbol in a plug-in module results in invalid addressing upon detaching the module, we recommend the use of memory database which is aware of the plug-in status. Global variables are also shared, but run the risk of name conflicts.

Since the plug-in interface in memory database is established by plug-in itself, the host program can bind a new plug-in module without knowledge about the plug-in. As long as new plug-ins are consistent with the host program and other plug-ins, they can be added to the software system incrementally.

Graphics and User Interface Library

Graphics and graphical user interface (GUI) libraries are the predominant answer to the problem of portability. We have designed our library to be portable and suitable for sphere and cylinder primitives. The library renders them into an off screen image buffer with Z-buffer and transfers to the video RAM in the same way as RasMol (Sayle 1995) together with our proprietary algorithm of arc generation for antialiasing sphere (unpublished). The image buffer for rendering is customizable and can also act as an animation frames.

The GUI library is also redesigned for the adapting plug-in environment. Individual GUI parts in a dialog box opened by a plug-in are created in memory database so that other modules can access those parameters given by the user interface. The key name of the GUI can contain an *order* name of its callback function.

Implementation

Our program system was configured as an application program and its supporting library (Fig. 1). The Application Support Hybrid Library is responsible for the portability of the software system and the interface to the plug-in module.

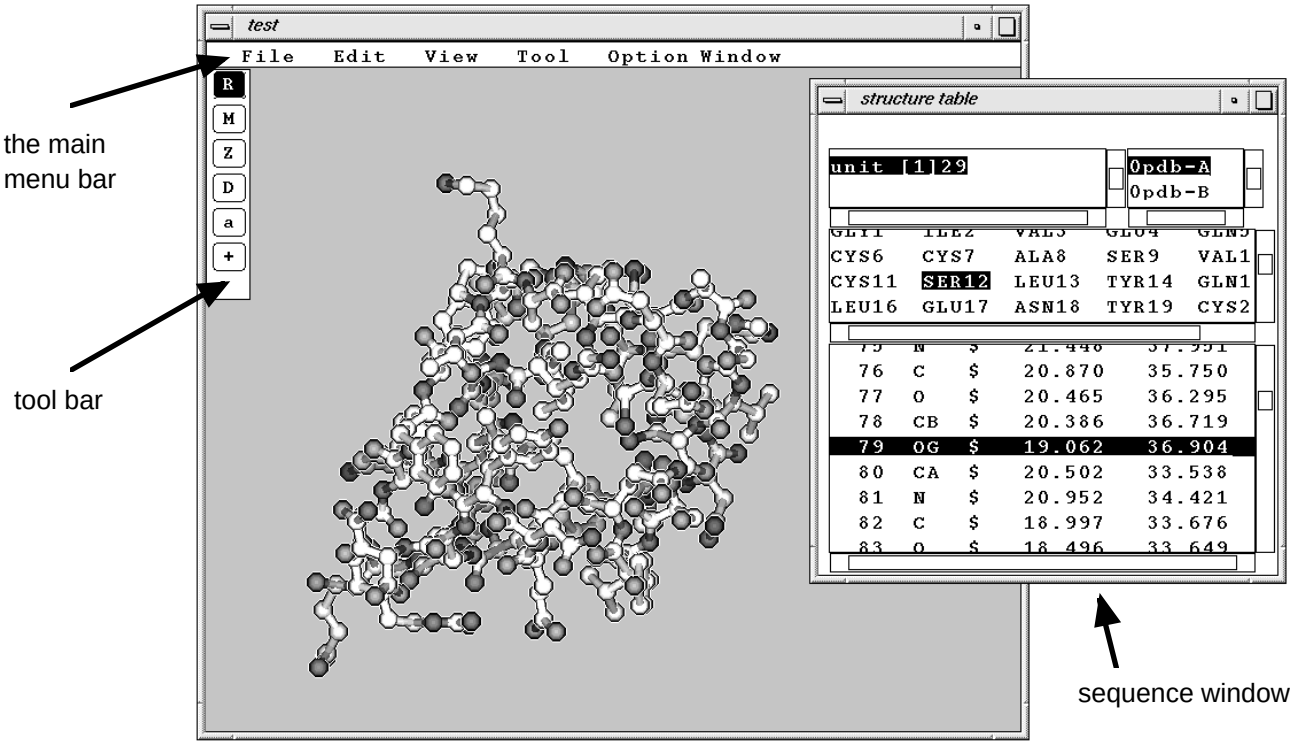

Fig.2. A Typical View of the Molecular Structure Browser. An Insulin monomer was depicted with a sequence window.

ANSI C language was used to maximize portability. The program and plug-ins were tested to work on several UNIX systems (Silicon Graphics, Inc.; Sun Microsystems, Inc.; Hewlett Packard Co.; Digital Equipment Corp. and Linux operating system (Torvalds) on IBM-PC) with a Xlib library for the X Window System. More than 16 Mbytes of main memory and 8 bit color or gray scale monitor were required.

Molecular Structure Browser

The molecular structure browser, MOSBY, was successfully implemented with the following basic browsing functions of atomic coordinates of proteins:

- Molecular models of wire frame, ball & stick, tube and space filling.
- Real-time viewing manipulation of molecules by mouse operations with depth-cueing, clipping, scroll, and perspective.
- A sequence window provides the amino acid sequence view of the protein. The selection of residue also cooperates with the molecular model window and vice versa.

Most operations are in the main menu bar on top of the window (Fig. 2). Items in the tool bar at the left side select the mode of mouse operation. With the default configuration of the program, the size of a molecule is up to ~100,000 atoms and ~10,000 residues, the size of the image data is 1600x1200 in 8 bit color with 16 bits of depth buffer.

The graphics rendering which has not been optimized was several times slower than RasMol, but the rock animation with four cyclic frames provides smooth horizontal rotation that is independent of the number of atoms. Additionally, the qualities of cylinder and sphere were improved by antialiasing and the edge line between primitives (Namba et al. 1988).

There are several plug-ins to enhance functions of our browser program:

An Electron Density Plug-in. This module reads an electron density map file and visualizes the isosurfaces of density function into the molecular model as well as FRODO (Jones, T.A. 1978) by an algorithm similar to the marching cube (Lorenson & Cline 1987). The plug-in has a dialog box for the parameter set of isosurface values that is prepared in a file. By trapping an *order* function of the rendering atomic model, the plug-ins depict various kinds of pictures into the molecular model window.

A Superposition Plug-in. With specified two sets of atoms defined as molecular groups in our browser program, the coordinates of the first group are translated to fit those of the second. Since the plug-in only describes its algorithm, it is easy for users to customize or replace it. Furthermore, functions for editing atomic data in our browser enables another plug-in of molecular modeling with a customized

graphical user interface.

Filter Plug-ins. The browser program loads the filter plug-ins which access atomic data in different kinds of formats. A filter plug-in defines a new *order* that is a function of the filter and registers it into the list of filters. A new format is supported simply by modifying an existing sample filter.

Discussion

The Plug-in as A Software Component

Adler (1995) reviewed the proposed architectures of component software such as OpenDoc (Apple Computer, Inc.), OLE (Microsoft, Inc.), and CORBA (Object Management Group), however understanding them requires considerable technical backgrounds. In contrast, our plug-in architecture was designed with simplicity in mind, so that biologists can focus on solving specific biological problems by computation. Our memory database provides an alternative to the global variable with another name space apart from the symbols resolved by a linker of the operating system.

The use of inter-process communication is another way to provide extensibility of the software system. It requires established communication protocols to synchronize the processes. On the other hand, our plug-in modules with its single thread execution stays in the comprehensive programming model. The scripting language, for example Tcl (Ohsterhout 1994), also enables the functional extension of the system. However, the slower processing speed in interpreting, and the potential barriers for the new language are two disadvantages.

Additionally, a plug-in can be written in any language by means of reference to our C library. The memory database is also applicable to the integration with Java and the native method in C or FORTRAN. Also, an *order* name "www" is reserved to open a Web browser window from a program. This simple software configuration method is generally useful for providing extension capabilities in the application systems.

Future Plan

In current development, an OpenGL (Silicon Graphics, Inc.) driver plug-in is planned to access accelerated hardware. Our library will also be ported to Windows (Microsoft, Inc.) and MacOS (Apple Computer, Inc.).

For advanced development, parallel processing can be implemented as the way of memory tuple in Linda system (Ahuja et al. 1986). The use of signal-handling detaches the faulty plug-in to avoid program termination and enables the update of the plug-in. It benefits considerably in prototyping and the subsequent debugging of plug-in modules. Cooperating with the plug-in distribution server can automate the update and installation of plug-ins via a network with an appropriate licensing protocol.

Conclusion

Our design of a molecular structure browser was successfully implemented with both plug-in style extensibility and portability. The plug-in architecture by memory database and *order* is simple and comprehensive without the use of technical dynamic-linking details. Portability is maintained by our programming library, which includes three-dimensional graphics and user interfaces adapted for modern window systems. The use of our library for software configurations is generally applicable to other systems with extension capabilities.

Availability

Source code of prototype is available upon e-mail request for developers and our molecular structure browser will be released at ftp://ftp.etl.go.jp/pub/bioinfo.

References

Abram, G. & Treinish, L. 1995. An Extended Data-Flow Architecture for Data Analysis and Visualization. *Computer Graphics* 29.2:17-21.

Adler, R.M. 1995. Emerging Standards for Component Software. *IEEE Computer* 3:68-77.

Ahuja, S., Carriero, N. & Gelernter, D. 1986. Linda and friends. *IEEE Computer* 19:26-34.

Jones, T.A. 1978. A graphics model building and refinement system for macromolecules. *J. Appl. Cryst.* 11:268-272.

Lorenson, W.E. & Cline, H.E. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Proceedings of SIGGRAPH'87. *Computer Graphics* 21(4):163-169.

Namba, K., Caspar, D.L.D. & Stubbs, G. 1988. Enhancement and Simplification of Macromolecular Images. *Biophys.J.* 53:469-475.

Ohsterhout, J.K. 1994. *Tcl and the Tk Toolkit.* Addison-Wesley, New York.

Richardson, D. C. & Richardson, J.S. 1992. The Kinemage: A tool for scientific communication. Protein *Science* 1:3-9.

Sayle, R. A. & Milner-White, E.J. 1995. RasMol: Biomolecular Graphics for All. *Trends in Biochemical Sciences* 20:374-376.

Tovalds, L. The Linux Operating System. http://www.linux.org.

Upson, G., Faulhaber, T., Kamins, D., Laidlaw, D., Schlegel, D., Vroom,J., Gurwitz, R. & Dam, A. 1989. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE CG&A.* 9(4):30-42.