References

1. R.A. Sayle and E.J. Milner-White, *Trends in Biochemical Sciences* **20**, 374-376 (1995).
2. Y. Ueno and K. Asai, *Proceedings, Intelligent Systems for Molecular Biology* **5**, 329-332 (1997).
3. B.A. Myers, R.G. McDaniel, R.C. Miller, A. Ferrency, P. Doane, A. Faulring, E. Borison, A. Mickish and A. Klimovitski, *CHI'97 Conference Companion: Human Factors in Computing Systems, Atlanta, GA.* 214-225 (1997). http://www.cs.cmu.edu/ amulet/
4. J. Bresehnam, *Commun. ACM* **20**, 100-106 (1977).
5. D.W. Fellner and C. Helmberg, *ACM Trans. Graph.* **12**(3),251-276(1993).
6. D. Field, *ACM Trans. Graph.* **4**(1),1-11(1985).
7. K. Namba, D.L.D. Caspar and G. Stubbs, *Biophys.J.* **53**, 469-475 (1988).
8. A.S. Ivanov, A.B. Rumjantsev, V.S. Skvortsov and A.I. Archakov, *CABIOS* **13**(1), 111-113 (1997).
9. S.H. Bokhari, *IEEE Computer* **28**(8), 74-79 (1995).
10. R. Koradi, M. Billeter and K. Wurich, J. Mol. Graphics, **14**, 51-55 (1996).
    http://www.molbiol.ethz.ch/wuthrich/software/molmol/
11. InsighII, Molecular Simulation Inc.
12. T.E. Ferrin, C.C. Huang, L.E. Jarvis and R. Langridge, J. Mol. Graphics, **6**, 13-27 (1988). http://www.cgl.ucsf.edu/midasplus.html
13. Programmer's Hierarchical Interactive Graphics System (PHIGS), ISO standard 9592:1988(E).
14. R.J. Rost, J. Friedberg, and P, Nishimoto, *IEEE Computer Graphics & Applications* **9**(4), 14-26 (1989).
15. Silicon Graphics Inc.
    http://www.sgi.com/Technology/openGL/paper.design/opengl.html
16. Dore: Dynamic Object Rendering Environment.
    ftp://sunsite.unc.edu/pub/packages/development/graphics/Dore
17. P.S. Strauss and R. Carey, *Computer Graphics* **26**(2), 341-349 (1992).
18. B. Paul, The Mesa 3-D graphics library.
    http://www.ssec.wisc.edu/ brianp/Mesa.html
19. U. Flobr, *Byte* **10**, 76-88 (1996).

Appendix: A Circle Algorithm

From the rotation matrix, the points (X_n, Y_n) on the circle of radius R can be generated by following differential equations with θ being a differential polar angle between the points.

$$\begin{aligned} X_{n+1} &= X_n cos\theta - Y_n sin\theta \\ Y_{n+1} &= X_n sin\theta + Y_n cos\theta \end{aligned} \tag{1}$$

Letting $\theta = 1/R$ radian, which gives contiguous points, and using Taylor's expansion for $sin\theta$ and $cos\theta$ up to the second order, where the accumulating error of the order θ^3 is negligible for $n \leq 2\pi R$, we get the approximation form of Eq.1.

$$\begin{aligned} X_{n+1} - X_n &= \frac{-Y_n}{R} - \frac{X_n}{2R^2} \\ Y_{n+1} - Y_n &= \frac{X_n}{R} - \frac{Y_n}{2R^2} \end{aligned} \tag{2}$$

This is the differential equation for the generation of an arc. We separate the fractional part of X_n and Y_n into two positive numerators on denominators of R and $2R^2$ using integer registers, $ix_n, iy_n, fy_n, fy_n, gy_n, gy_n$.

$$X_n = ix_n + \frac{fx_n + \frac{gx_n}{2R}}{R}\ ,\ Y_n = iy_n + \frac{fy_n + \frac{gy_n}{2R}}{R} \tag{3}$$

Since the changes of the coordinate in each step are always decimal fractions, we only calculate the fractional part.[6] When fx_n reaches to R it carries to ix_n if it reaches to R, and it borrows from if negative. The method is simplified by ignoring terms smaller than $1/R$ in Eq.2.

$$\begin{aligned} fx_{n+1} - fx_n &= -iy_n \\ fy_{n+1} - fy_n &= ix_n \end{aligned} \tag{4}$$

In each step, carries and borrows of numerator registers fx_n and fy_n on a denominator R generate the next point of arch. By incorporating the second fractions, gx_n and gy_n that carries to fx_n and fy_n in Eq.3, we can calculate accurate points of arch. Additional calculations are obtained from Eq.2.

$$\begin{aligned} gx_{n+1} - gx_n &= -2fy_n - ix_n \\ gy_{n+1} - gy_n &= 2fx_n - iy_n \end{aligned} \tag{5}$$

4.3 Graphics Acceleration Hardware

Flobr [19] has reviewed the current three-dimensional graphics technologies, OpenGL, QuickDraw3D(Apple Computer Inc.), and Direct3D(Microsoft Corp.), with their acceleration hardware on personal computers. The hardware provides different levels of functions: calculation of geometry and lighting color composition, rasterization of primitives, and frame-buffer control with quick pixel operations. Their advanced polygon function can be utilized in our graphics library by mapping a frame buffer on the acceleration hardware into the main memory to be accessed as an image buffer. The point list will be cached in their local memory for the fast rendering and geometry calculations. These functions can be implemented as plug-in extension modules to our graphics library that provide a seamless environment on variety of different hardware types in both software development and operational tasks.

Additionally, our circle algorithm used in the scan-conversion of a sphere without shift operation is more suitable for low-cost hardware implementation than the other algorithms. The normal vectors on the surface of the sphere can also be obtained for further lighting calculations.

5 Conclusion

Our graphics library, with its novel functions has been successfully implemented to provide high-throughput rendering for a molecular structure viewer on conventional computer hardware. The point list facilitates flexible management of the display list for molecular graphics. The scan-line conversion of spheres and cylinder primitives and the z-buffered bit-block transfer increases the rendering performance. Although our library lacks smooth-shaded and texture-mapped polygons, extension plug-in modules can supply those functions for a runtime application. The design of our graphics library is generally useful for software systems with portable three-dimensional graphics. The source code for our software is available upon e-mail request and our molecular structure viewer will be released at ftp://etlport.etl.go.jp/pub/bioinfo.

Acknowledgments

We would like to thank Dr. N. Otsu, Dr. K. Itou, Dr. Y. Akiyama and Mr. K. Onizuka for their helpful discussions and encouragement. We are indebted to Dr. D.L.D. Caspar, Dr. K. Namba and Teijin Ltd. for the development of the first prototype program.

strated by the molecular graphics system MIDAS[12] with its modular software design. Therefore, several general graphics libraries, such as PHIGS[13], PEX[14], and DORE[16] have been designed with an object-oriented display list database. In this approach, the hardware acceleration have been necessary for interactive rendering with the sphere and cylinder primitives that are made up of polygons. Another issue is the required flexibility of the display list[17]. For example, changing the representation of the molecular model from a built-in wire frame to a ball-stick model requires a reconstruction of the display list. Additionally, these graphics libraries have not been adapted to personal computers.

OpenGL and its predecessor, IRIS GL (Silicon Graphics Inc.), have provided flexible management of the display list to the application program by the immediate rendering of graphics primitives with state variables for their attributes. However, the rendering of spheres and cylinders still depend on polygons and therefore hardware acceleration. Consequently, Mesa[18], a portable OpenGL implementation, is still unable to provide practical rendering performance.

Our graphics library has been designed to solve this problem by introducing the direct scan-conversion of spheres and cylinders. The API of our graphics library is similar to the OpenGL, which facilitates a porting of the applications. It is also possible to adapt to OpenGL-based systems, providing the OpenGL interface with greater ease of use.

4.2 Comparison with RasMol

In contrast to those graphics libraries, the programs on personal computers have been developed differently, with an emphasis on optimizing code for specific graphical tasks. For instance, RasMol is one of the most successful pieces of software in molecular biology with its outstanding rendering performance. We have also found novel technologies for raster graphics in the source code of RasMol. Although, it was difficult for us to modify the program for some customized functionality without a substantial rearrangement of its source code.

We propose the use of a point list to separate the graphics rendering code from protein structure data without performance loss. RasMol requires the introduction of a twin-line primitive that paint half of a line with different color. The point list allows this additional primitive as well as a flexible manipulation of the display list in an application program.

Recently, OpenGL introduced the vertex array that is similar to our point list. However, it is rather complicated and lacks the support of the screen coordinate that is indispensable for the rearrangement of RasMol.

Table 2: Rendering Performance Benchmark Test

software	wire frame	VDW surface	ball-stick
MOSBY	10.3	4.0	1.3
RasMol	30.0	6.0	12.0
MOLMOL [10] (OpenGL) or InsightII [11]	16.0	$0.05 \sim 0.5$	$0.2 \sim 0.7$
with a graphics accelerator *	>400	$0.8 \sim 8.2$	$6.9 \sim 11.0$

Numbers are in 1000 atoms per second. They are estimated from the elapsed time in rendering for PDB data 1ATN consisting of 4945 atoms and 5126 bonds. An ordinary workstation, INDY , Silicon Graphics Inc., equips MIPS R4000 (100MHz) with FPU, 64MB main memory, 8k data + 8k instruction cache, 8 bit video graphics (1280x1024) is used. *: The results with a graphics accelerator were obtained on another machine, an INDIGO Solid Inpact2 with MIPS R10000, which is about 5 times faster in general processing. A moderate and the fastest numbers were shown depending on the quality of sphere divided into polygons.

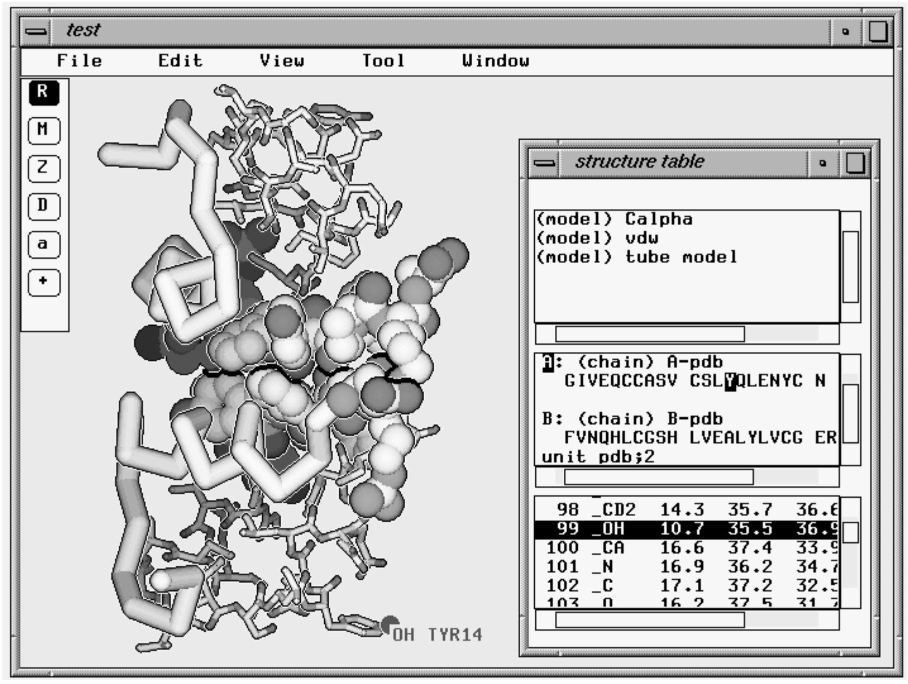
The graphics rendering benchmark results (Table2) on a machine without graphics acceleration hardware showed a better performance than applications built on the OpenGL for VDW and ball-stick. However RasMol is the fastest, current rendering performance of our viewer was acceptable for practical use.

There are several extension plug-in modules which can enhance the functions of our viewer program as described previously [2]. The electron density plug-in can read a density map file and display its iso-surface on the molecular model. The Supperposition plug-in works for fitting two selected fragments of molecules. Filter plug-ins can be prepared to access files in different formats. A variety of extension modules with molecular modeling and visualizations of calculation results with the atomic coordinates are possible to enhance functions of our viewer.

4 Discussion

4.1 Comparisons with Other Graphics Libraries

Most graphical application program in various fields maintain an underlying display list database to be depicted and manipulated. It was also well demon-



```
(model) Calpha
(model) vdw
(model) tube model
```

```
A: (chain) A-pdb
  GIVEQCCASV CSLYQLENYC N

B: (chain) B-pdb
  FVNQHLCGSH LVEALYLVCG ER
unit pdb;2
```

```
 98 _CD2   14.3   35.7   36.6
 99 _OH    10.7   35.5   36.9
100 _CA    16.6   37.4   33.9
101 _N     16.9   36.2   34.7
102 _C     17.1   37.2   32.5
103 _O     16.2   37.5   31.7
```

Figure 4: MOSBY, a molecular structure viewer. An insulin dimmer is depicted with boundary outlines between subunits. A table window lists the atomic coordinates, ts amino acid sequence, and groups of representations.

- A table window provides the amino acid sequence view of the protein. The selection of residue also cooperates with the molecular model window, and vice versa.
- The rock animation with four cyclic frames of horizontal rotation is always smooth for large molecules.
- Improved quality of the cylinder and sphere.
- The edge-line between the primitives [7] makes very clear and easily distinguished pictures especially in gray scale.
- Support of Large molecular pictures with up to $\sim$ 100,000 atoms and $\sim$ 10,000 residues in an image data of 1600x1200 pixels by default.
- Plug-in style extension modules.

Table 1: The Application Program Interface of The Graphics Library

category	functions
Rendering	YovMove3w(), YovLine3w(), YovPoly(), YovVert3w(), YovGon(), YovPutStr(), YovSphere3w(), YovTube3w(), YovCircle2w(), YovCirclef2w(), YovDots2i(), YovBox2i(), YovFBox2i(), YovRBox2i(), YovRFBox2i(), YovClear(), YovClearView(), YovClearBox2i() (*tailing 3w can be either of 3w,3s,3i,2w,2s,2i*)
Point List	YovCreatPt(), YovPtWorld(), YovPtScreen(), YovMove(), YovLine(), YovVert()
Graphics Attributes	YovSetPen(), YovSetEdge(), YovSetBack(), YovSetAttr(), YovSetOpac(), YovSetLineWidth(), YovSetDash(), YovSetBrush() YovSetRound(), YovSetPattern()
World Coordinates	YovWide(), YovDepth(), YovCenter3(), YovSee3(), YovMag(), YovPan3(), YovProject3(), YovScreen2() YovScreenDepth(), YivSize2Pixel(), YdvPixel2Size(), YovViewport3(), YovWorld3(), YovClip2i(), YovClipDepth(), YovPenPos3w()
Picking	YovSetPickTag(), YovStartPick(), YovEndPick()
Image Buffer Ccontrol	YbvCreateImage(), YovSetImage(), YovMapImage(), YovShowImage(), YbvCreateSubImage(), YovImageSize(), YovPutImage()
Window System Interface	YiwOpenWin(), YowCloseWin(), YowEnableWin(), YiwMainLoop(), YowOnMouse(), YowOnKeyboard(), YowOnRepaint(), YowOnResize(), YowOnNotify(), YowSetInfo(), YawInfo(), YowDoRepaint(), YiwCheckInput(), YowSetWin(), YiwWin(), YowAdjustWin(), YiwWidth(), YiwHeight()

Figure 3: A Sample Source Code using the Point List. Two data is created in the point list to draw a line that will be translated into the screen coordinate only if necessary.

```
long pt1,pt2;
  /* create data */
  /* in point list */
YovMove3w( 0.,0.,0.);
pt1=YavCreatePt();
YovMove3w( 1.,0.,0.);
pt2=YavCreatePt();
```

```
  /* ex.  do zoom */
YovMag(1.1, TRUE);
...
  /* draw a line */
YovVert(pt1);
YovVert(pt2);
YovLine();
```

support for smooth-shaded and texture-mapped polygons, but these functions can be supported by an extension plug-in module for a runtime application built on our library. There are Application Program Interfaces (APIs) for normal vectors and texture coordinates, making it possible to render those polygons in ways that are usually ignored. Taking advantage of the dynamic linking functions provided by the operating system, a program can append the functions of an extension plug-in module. Since the polygon-rendering code in our library which is linked to the program is interchangeable, an external module can replace it with another code which allows for smooth shading and texture mapping. Thus, our graphics library determines its function at a runtime of the application program incorporating with its extension modules. This design has greatly simplified the basic specification of our graphics library.

3 Implementation

3.1 The Graphics Library

The graphics library was implemented in the C language on an X-Window system running on several UNIX workstations with recent operating systems (Silicon Graphics, Inc.; Sun Microsystems, Inc.; Hewlett Packard Co.; Digital Equipment Corp.; and IBM-PC with Linux [9]) using the Xlib library. More than 16 Mbytes of main memory and an 8-bit color or gray-scale monitor are required for the library to run. The API of our graphics library is summarized in Table1 and a sample code using the point list is given in Figure3.

The graphical user interface (GUI) functions were also developed with the graphics library and Xlib. This allows for a GUI application with a GUI definition file and only a small amount of source code. Together with the dynamic linking functions, the graphics library was organized into our programming library called ASHLEY, which stands for an application support hybrid library for easy programming.

3.2 A Molecular Structure Viewer

A molecular structure viewer, MOSBY, has been successfully equipped with the basic browsing functions for the atomic structure of proteins. Molecular models of wire frame, ball-stick, tube, and VDW surface are supported with real-time viewing manipulation with depth-cueing, clipping, and perspective by mouse operations. Items in the tool bar at the left side select the mode of mouse operation. (Figure 4). Special features in our viewer is as follows:

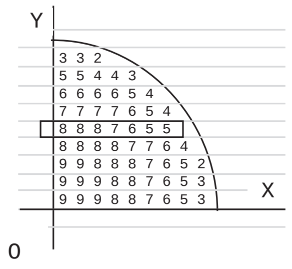


Figure 2: The Typical Scan-conversion of a Sphere. Scan lines with the same pixel length have different shadings. The number represents the z-component of the normal vector on the sphere.

lines and circles, we have found an algorithm which is effective for the subpixel calculations to improve the quality of the sphere (Figure 2). The algorithm described in Appendix eliminates the shift operations in several similar works.[5] It can readily be extended to the elliptical arch for the scan-line of a sphere because the z-buffer is in a finer scale than pixel coordinates. The cylinder primitive, for which the scan-lines are all elliptical arches, is also rendered by this algorithm.

2.4 The Image Buffer Configuration

The image data can be allocated with the z-buffer in the main memory as much as possible that can also be used as frames of an animation. An additional attribute plane can be used for applications to set arbitrary values for each pixels of the image. On rendering primitives a specified attribute value is stored to the plane as well as the depth value to the z-buffer. Setting the subunit identification code of the protein molecules to this attribute allows the extraction of an outline for the different subunits by comparing its adjacent values [7] (Figure 4). We have configured 4 bits of this plane in the 16-bit z-buffer, as these attribute values leave a depth range of 12 bits.

2.5 Z-buffered Bit-Block Transfer

The image data rendered with z-buffer values can be transferred to other images with z-buffer tests. This allows for the quick duplication of a sphere primitive. It can also be accelerated by hardware as well as the normal bit-block transfer operation in the two-dimensional image. This is an abstracted function for the use of template spheres that has also been employed by a molecular viewer program ONIX [8].

2.6 Dynamic Linking Extension Module

We have applied a plug-in style software component architecture [2] that extends the functions of our graphics library. For example, our graphics library lacks

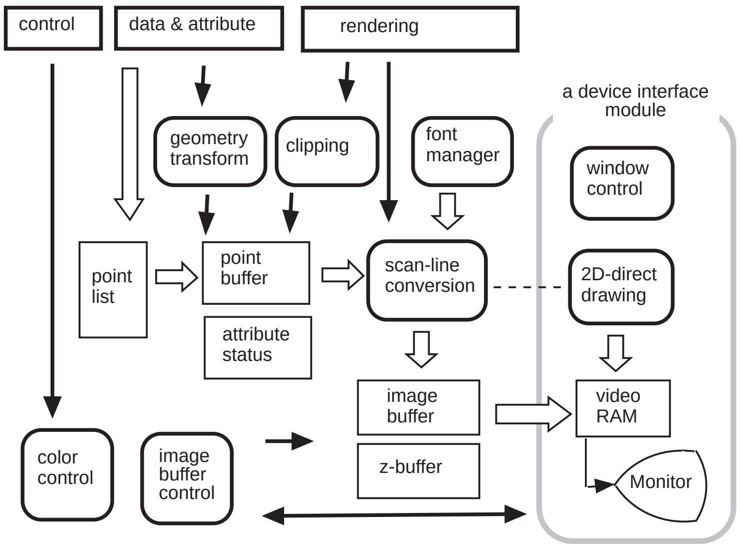


Figure 1: The Block Diagram of our Graphics Library. The current implementation for X-Window system includes a direct drawing of two-dimensional primitives using Xlib. The font manager draws character text both for X-Window and the image buffer.

provide a display list of graphical primitives, a coordinate in the connecting objects which could be shared among graphics primitives appears many times in each of the separated primitives and is calculated redundantly. In our graphics library, the point list simply provides an index of those coordinates used in rendering function, and the display list is built in an application program (Figure3). It is important to have the subpixel fraction of the screen coordinates for smooth animation on small rotations of graphics primitives. The concept of the point list is also employed by Amulet[3] in the two-dimensional graphics.

2.3 Scan-Conversion of a Sphere

Because each scan-line of a sphere forms an arch with a different radius in the x-z plane, the scan-conversion of a sphere is achieved by a digital differential analyzer (DDA) for generation of the arch. While most graphics libraries employ sphere and cylinder primitives made from polygons, which consume a significant amount of the processing resources, the direct scan-conversion is practically efficient also in RasMol.

Although Bresenham's algorithm[4] is well known for the scan-conversion of

Moreover, portability is the key to creating state-of-the-art software tools which will become a source of discussion in the research community. If a software program with innovative structure analysis and calculations were portable, it would be of interest to various researchers for use in their area of biological research. Recent advances in technology for computer hardware have improved affordable personal computers with necessary hardware for three-dimensional graphics tasks. Thus, portable software tools in structural studies for biomolecules are in great demand.

We have designed a graphics library that is portable among a wide range of hardware platforms, as part of our new software-development project creating molecular graphics tools for protein structure analysis and prediction studies. Our goal is to provide a software platform which will run on common hardware and will allow users to add new functions with only an average knowledge of programming. In this project, we have designed our software system with respect to two issues: portability and extensibility. Our method of the plug-in style module extension by means of dynamic linking has been reported previously[2]. This paper describes our design of a portable graphics library for a molecular structure viewer program with high-throughput rendering.

2 Designing the Graphics Library

2.1 Overview of Our Graphics Library

Our graphics library provides simple rendering functions in the specified three-dimensional world coordinates. Primitives of lines, boxes, character text, circles, polygons, spheres, and cylinders are supported. These primitives are rendered into the off-screen image buffer that is then transferred to the raster graphics hardware for display on a monitor. This simple design makes our graphics library portable.

Figure 1 summarizes the block diagrams for our graphics library. The world coordinate systems for graphical primitives are translated to the device-dependent screen coordinates with a perspective or parallel projection and depth cueing. The attributes of graphics primitives, color, line width, line pattern, are the in-state variables used in rendering. The z-buffer is used for hidden surface elimination. The processor-intensive primitives such as curves, surfaces, and lighting attributes are not supported (Table1).

2.2 The Point List

The point list is an array that maintains pairs of the world and screen coordinates as an alternative to the display list. While most graphics libraries

A High-throughput Graphics Library Designed for a Portable Molecular Structure Viewer

Yutaka UENO and Kiyoshi ASAI
Electrotechnical Laboratory 1-1-4 Umezono, Tsukuba 305 Japan

We have equipped our graphics library with efficient functions so that a molecular structure viewer program can provide both portability and high-throughput rendering without hardware acceleration. The library renders graphics primitives into off-screen image memory with novel functions such as a point list for the vertices of three-dimensional graphical primitives, scan conversions of sphere and cylinder primitives, and z-buffered bit-block transfer. A molecular structure viewer program was implemented with the graphics library, giving reasonable rendering performance on conventional UNIX workstations with the X-Window system. The use of dynamic linking also lends a flexible extension facility to this software system. An advanced polygon renderer can be provided as an plug-in style extension module as well as other functional modules that are specific to the application program. The design of our graphics library is not only effective in molecular graphics but is also applicable for general three-dimensional graphics software systems.

1 Introduction

The application of computer graphics to molecular modeling and visualization has successfully allowed a number of software systems to provide various ranges of functionality. The real-time manipulation of graphical models is usually achieved by using hardware acceleration to render graphical primitives. Graphics libraries ensure the compatibility and portability of the application software, as well as a long lifetime for such software, even when there are alterations in the hardware architecture.

While most interactive molecular graphics programs depend on hardware acceleration, RasMol[1], a comprehensive molecular structure viewer program, provides reasonably interactive three-dimensional graphics on ordinary workstations and personal computers without special hardware acceleration. With the increasing demand for visualizing protein structures in molecular biology, the program has been broadly used among a wide range of researchers. The program renders molecular pictures into off-screen memory without the use of a conventional graphics library. In fact, no graphics library has allowed this kind of rendering performance on systems without graphics acceleration. Although graphics libraries provide abstracted interfaces to various kinds of acceleration hardware, an effective new graphics library without graphics acceleration hardware has been considered to be desirable for the advanced portability of software systems.