Acta Cryst. (1994). D**50**, 869–873

Raster3D Version 2.0. A Program for Photorealistic Molecular Graphics

By Ethan A. Merritt* and Michael E. P. Murphy

Department of Biological Structure SM-20, University of Washington, Seattle WA 98195, USA

(*Received* 10 *March* 1994; *accepted* 6 *June* 1994)

Abstract

Raster3D Version 2.0 is a program suite for the production of photorealistic molecular graphics images. The code is hardware independent, and is particularly suited for use in producing large raster images of macromolecules for output to a film recorder or high-quality color printer. The *Raster3D* suite contains programs for composing illustrations of space-filling models, ball-and-stick models and ribbon-and-cylinder representations. It may also be used to render figures composed using other graphics tools, notably the widely used program *Molscript* [Kraulis (1991). *J. Appl. Cryst.* **24**, 946–950].

Introduction

The use of photorealistic rendering for the presentation of molecular graphics has a number of advantages. One notable feature is that the sense of depth conveyed in the rendered image can obviate the need for a stereo pair. This is advantageous, for example, in the preparation of transparencies for lecture presentation, a setting in which stereo pairs are not easily viewed. The rapid increase in the power of workstations typically used for interactive molecular graphics has allowed many applications to incorporate display options which render solid surfaces using the workstation's graphics hardware and supporting software libraries. Although this approach has clear advantages in the tight integration of interactive work with the production of raster images for the presentation of results, there is still a need for hardware-independent rendering software to produce the highest quality illustrations. The rendering algorithms implemented in workstation hardware and software are chosen to support real-time interactive use and, therefore, do not necessarily represent the best that can be achieved if this restriction is removed. In particular, it is rare to find support for shadowing, despite the fact that it is one of the most effective visual cues to depth within the image. Furthermore the resolution of images displayed on a workstation screen is limited to the number of pixels on the screen, typically 1280 × 1024, and screen photography or other methods of recording the image directly from the display are therefore subject to this inherent limit. The *Raster3D* suite of programs is designed to bypass these limitations while maintaining as much ease of use as is practical.

Because *Raster3D* is targeted at a very specific application, space-filling and schematic representation of molecular structure, the choice of rendering algorithms can be optimized for this intended use. This allows a considerable gain in image quality and speed when compared to more general programs which must handle a wider variety of applications. For example, typical workstation rendering hardware and associated software requires decomposition of all surfaces into triangularly faceted approximations. Typical workstation implementations then use a surface shading approximation (Gouraud, 1971) which additionally requires an appropriate surface normal to be associated with each vertex of each triangular facet. This is used to calculate the 'correct' shading at each vertex, and the interior of the facet is then filled in by interpolated blending of the vertex colors. By contrast *Raster3D* uses an analytic description of simple objects such as spheres and cylinders, while still allowing more complex surfaces to be built up from a triangular mesh. This allows the shading of spheres and cylinders to be 'perfect'. Furthermore, the external knowledge that the triangular mesh decompositions describe ribbons, arrows, and helices (rather than some more general surface) permits a fairly simple implementation of a more accurate surface-shading algorithm (Phong, 1975). *Raster3D* supports a single primary light source which illuminates the scene from any specified direction, and which optionally casts shadows. The scene is simultaneously illuminated by a secondary light source, which always shines from the direction of the viewer and hence casts no visible shadows. The effects of these two light sources can be handled by two parallel *Z*-buffer calculations, a much simpler and faster approach than is required by general purpose ray-tracing programs for the handling of a potentially more complex specification involving many light sources. In practice the images produced using *Raster3D* are of noticeably higher quality than is achieved using workstation graphics hardware (Fig. 1), and require much less computational time than equivalent images from a fully general ray-tracing program.

* Author for correspondence.

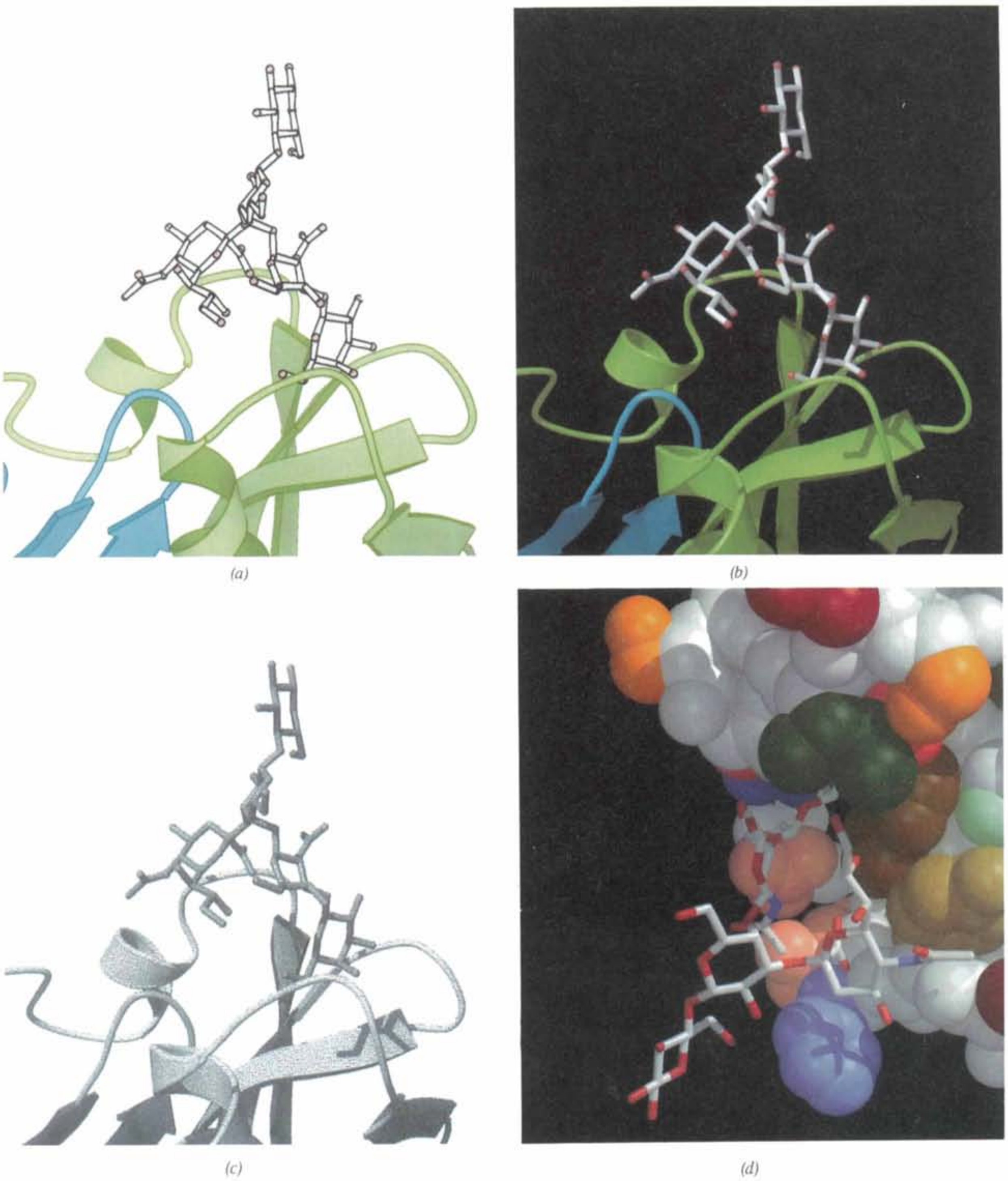

Fig. 1. Representative examples of the use of *Raster3D* to prepare figures. This composite figure was printed as a single image on a 300 dpi dye sublimation printer. (*a*) Schematic representation of the receptor binding site in cholera toxin (Merritt *et al.*, 1994), prepared using version 1.4 of the program *Molscript* (Kraulis, 1991) to generate a PostScript file. (*b*) A *Raster*3D rendering of the same scene generated directly from the same *Molscript* input file, *i.e.* unix commands to generate the images in (*a*) and (*b*) correspond respectively to: `molscript < infile > A.ps; molscript -r < infile | render -tiff B.tif`. (*c*) The same *Molscript* input file was modified slightly for use in preparing a black and white image, then run through *Raster3D* using the *molras*3*d* script to produce a black and white shaded image. (*d*) An alternate depiction of the same binding site composed using the *Raster3D* programs *setup* and *rods*. To render this image as a 1024 × 1152 raster as shown here required 10.7 s of computation on a DEC Alpha 3500X under OSF/1.3.

The programs

The production of a photorealistic molecular graphics illustration may be thought of as comprising three stages. The first stage is the composition of the illustration, in which atomic coordinates are converted into a variety of schematic representations and presented in a specific orientation. The second stage is the rendering of this composed scene into a raster representation, which optionally involves the calculation of perspective, shading, shadowing, multiple light sources and associated specular highlighting *etc.* The third stage consists of transferring this raster representation to some specific output device or format, perhaps with the addition of labels or other annotation. The *Raster3D* suite is primarily a collection of programs for the composition and rendering of molecular graphics images. In addition it contains a small set of auxiliary tools for annotation and further image processing. The various components of the program suite are described briefly below.

Composition

The *Raster3D* distribution contains a number of utilities for composing images in a variety of representations. The *setup* program reads a standard Brookhaven PDB file, assigns colors and atomic radii by matching the atom names, chain identifiers *etc.*, against a file of template records, and produces an output file describing the resulting spheres in a format suitable for input to *render*. The *rods* program performs a similar function in producing a file of cylinder descriptors which represent bonds and, optionally, associated atoms to create a ball-and-stick representation. The *ribbon* program selects Cα and O atoms from a PDB file to fit a smooth ribbon for the entire backbone of each protein chain in the file. The output files from these composition tools may be mixed or combined as desired before feeding the resulting composite image description to *render*. This allows quite complex figures to be built up from relatively simple pieces (Fig. 1*d*). A separate tool for image composition, *preras*3*d*, is also available. This tool provides an alternative method for composing figures containing space-filled, ball-and-stick, ribbon, and curved-tube backbone representations. The ball-and-stick option includes support for solid or dotted bonds.

The generation of image descriptions for *render* is not necessarily limited to the use of these specific tools. Other, existing, programs can be easily modified to generate the required object description file. In particular the widely used program *Molscript* (Kraulis, 1991), has been modified so that version 1.4 fully supports the rendering capabilities of *Raster3D* version 2.0. This not only means that *Raster3D* rendering is possible from any existing *Molscript* script file, but also allows the use of *Molscript* as a 'nearly interactive' composition tool for new figures. An example of the use of *Molscript* to compose a *Raster3D* image is shown in Figs. 1(*a*) and 1(*b*).

Rendering

The central rendering program of the suite employs a fast Z-buffer algorithm to achieve image quality generally limited to full ray-tracing programs. This algorithm and the initial implementation of *Raster3D* were described by Bacon & Anderson (1988). Version 2.0 retains the speed, specular highlighting and shadowing of the earlier version, while adding support for additional object types, Phong shading of triangular mesh ribbons, and a greatly improved user interface.

The *render* program accepts input from a formatted file of object descriptors and produces a raster image in one of three selectable output modes. The current version of the *render* program handles five object types: spheres, triangles, planes and two forms of cylinders. Spheres are specified by center, radius and color. Triangles are specified by three vertices and a color. An isolated triangle is shaded as a flat surface with a uniform surface normal. Successive triangles with common vertices are recognized as a triangular mesh, and are rendered as a smoothly curved surface as discussed below. Planes are specified by three points and a color, and are treated as a triangle of infinite extent. The color intensity of the rendered plane surface, however, decreases with increasing distance from the viewer. Cylinders may either have flat ends, typically used for ribbon and cylinder representation of protein secondary structure, or have rounded ends, typically used for representing bonds. Flat-ended cylinders are specified as two points defining the center of two disks which form the cylinder ends, a cylinder radius and a color. Rounded cylinders are similarly described, but the cylinder ends are capped by hemispheres of the same radius as the cylinder.

Header records in the formatted input file to *render* specify the viewpoint for the image, including perspective and scale, the location of the primary light source, the degree of specular highlighting, the optional calculation of shadowing, and the relative intensity contributions of primary, secondary and ambient lighting to the final picture. Header records also control the raster size of the generated image and the level of anti-aliasing to be employed.

Post-processing

Although the best quality images are produced using a film recorder or a high-resolution color printer, other output devices can also take advantage of some of the features of *Raster*3D. Direct photography of *Raster3D* images displayed on a graphics screen will capture the shadowing and improved shading, albeit at lower resolution than a film recorder. For direct photography from a SGI graphics screen, the auxiliary program *show* provides an annotation tool with a menu driven interface and explanatory messages to guide the user. The annotations are saved in a separate file and may be redisplayed with the image for presentation or subsequent photography purposes. The *show* program is compatible only with images generated using the backwards-compatibility option of *render*.

Some figures may benefit from *Raster3D* processing even when generated for black and white reproduction. The auxiliary tool *molras3d* is a unix shell script which directs the output of *render* through a dithering program which reduces the full color image to black and white pixels. The resulting monochrome image is then converted into either Adobe PostScript or a PCL raster format suitable for printing on HP LaserJet compatible printers (Fig. 1*c*).

Implementation

Object types

Rendering a scene with multiple light sources and support for specular highlighting requires the calculation of the local surface normals for the component objects. In *Raster3D* Version 2.0 the surface normal for spheres and cylinders is calculated analytically for each pixel. The treatment for triangles is somewhat more complex, as described below.

By far the most common use for triangles as objects is to describe an approximation to a smooth surface by using a triangular tessellation. However, if the component triangles are faithfully rendered as flat surfaces this produces an image of a faceted surface rather than the desired smooth surface. This problem is well known, and can be overcome in a variety of ways. Generally this is achieved at the cost of maintaining additional information about the true surface normals of the original smooth surface, or about the connectivity of the component triangles in the tessellation. Because the surfaces being rendered in *Raster3D* are generally ribbon-like, a number of simplifying assumptions can be made which permit the calculation and rendering of a smooth, rather than faceted, surface with little additional overhead. As successive triangles are read from the input file to render, they are checked for common edges. Any triangle which is both preceded and followed by another triangle with a common edge is marked as a 'ribbon triangle' (Fig. 2). During subsequent rendering any points which lie on a ribbon triangle are treated according to an algorithm suggested by Phong (1975) in which points are assigned a surface normal which is interpolated from the surface normals at the three vertices of the triangle. In the present case the triangle's trailing vertex is assigned the surface normal of the preceding triangle; the leading vertex is assigned the surface normal of the succeeding triangle; and the middle vertex is assigned the original surface normal of the triangle itself (Fig. 2). Because ribbon triangles are identified internally by *render*, it is relatively straightforward to modify existing programs to generate *Raster3D* object descriptor files which will benefit from the improved shading algorithm. Minimal changes to *Molscript* (Kraulis, 1991), for instance, were required to provide an updated *Raster3D* interface. If for some specific purpose this shading algorithm is undesirable, it may be bypassed either by disordering the sequence of triangles or by inserting dummy objects into the sequence.

Output

Typically *Raster3D* is used to produce output for eventual printing on a high-quality color printer or on a film recorder. In order to transport the rendered images to any specific hard-copy device it is generally necessary to reformat them into some standard image format. Many general-purpose image processing and format conversion packages are available both commercially and as freely distributed software. Rather than duplicate this functionality within *Raster3D* we have chosen instead to support a limited number of output formats. In its default mode of operation *render* emits an AVS format raster image though *stdout*, which allows a direct unix pipe connection to an external image-format conversion program. Command line options to *render* permit three alternative output formats. One is a private image format provided for backwards-compatibility with some earlier versions of *Raster*3D. The remaining two options allow direct creation of image files compatible with the *libimage* image processing tools supplied with SGI/Irix workstations, or of images files in the widely recognized tagged image file (TIFF) format.

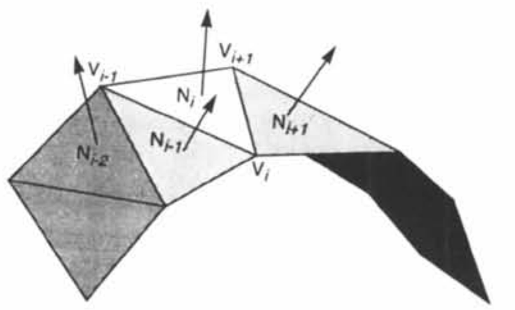

Fig. 2. Application of Phong shading to sequential triangles. Eacn vertex along the edge of the ribbon is assigned an associated vector corresponding to the surface normal of the central triangle of the three successive triangles to which it belongs. During rendering, points which lie on any given triangle are treated as having a surface normal which is a linear interpolation of the vectors assigned to the three vertices bounding that triangle. In the figure, the ribbon triangle with surface normal N_i has vertices V_{i-1}, V_i and V_{i+1}. Vertex V_{i-1} is assigned a vector equal to the surface normal of the trailing triangle, N_{i-1}; vertex V_{i+1} is similarly assigned the surface normal from the succeeding triangle, N_{i+1}. The Phong shading algorithm assures that the rendering used for the ribbon of sequential triangles possesses a continuous and smooth surface normal, rather than exhibiting a faceted appearance. Furthermore, specular highlights which lie within a triangle are correctly rendered by this treatment.

Availability

The *Raster3D* program suite is freely available but unsupported. The core programs have been extensively tested on a variety of platforms, including SGI/Irix, DECstation/Ultrix, Alpha/OSF, and ESV workstations. The auxiliary programs *atoms*, *show* and *molras3d* are architecture dependent, and have been tested primarily on SGI/Irix workstations. Source code, examples of use and documentation for the *Raster3D* suite may be obtained *via* anonymous ftp from site stanzi.bchem.washington.edu, or *via* email to the authors (merritt@u.washington.edu).

We are happy to give credit to all those who have contributed to the development of the *Raster3D* and related programs, including Wayne Anderson, David Bacon, Albert Berghuis, Mark Israel and Stephen Samuel. Registered trademarks mentioned in this text include Ultrix (Digital Equipment Corporation), OSF/1 (Open Software Foundation), PostScript (Adobe Systems), PCL (Hewlett Packard), SGI and Irix (Silicon Graphics), and ESV (Evans and Sutherland). MEPM acknowledges support from the Medical Research Council of Canada.

References

Bacon, D. J. & Anderson, W. F. (1988). *J. Mol. Graphics*, **6**, 219–220.
Gouraud, H. (1971). *IEEE Trans. Comput. C*, **20**(6), 623–628.
Kraulis, P. (1991). *J. Appl. Cryst.* **24**, 946-950.
Merritt, E. A., Sarfaty, S., van den Akker, F., L'hoir, C., Martial, J. A. & Hol, W. G. J. (1994). *Protein Sci.* **3**, 166–175.
Phong, B.-T. (1975). *Commun. ACM*, **18**, 311-317.