BUTTERWORTH HEINEMANN

Annotating PDB files with scene information

Gregory S. Couch, Eric F. Pettersen, Conrad C. Huang, and Thomas E. Ferrin

Computer Graphics Laboratory, University of California, San Francisco, California

We have implemented extensions to the Brookhaven Protein Data Bank (PDB) file format for incorporating scene information such as viewing parameters, additional molecular information (e.g., van der Waals radii and atom colors), and user-defined graphics. These extensions were made in conformance with the PDB standard and provide sufficient information to render the scene in various styles such as space-filling images and ribbon diagrams. For the past 5 years these extensions have been used in the MidasPlus molecular modeling system and have proved both powerful and sufficient for generating complex molecular images. We propose that the extensions to the PDB presented here be adopted by the molecular modeling community for incorporation into visualization programs.

Keywords: Protein Data Bank, annotation, metafile

INTRODUCTION

Transforming molecular coordinates into an image suitable for publication requires additional information not found in the coordinates alone. For instance, when illustrating a particular molecular feature, it is necessary to select the best angle to view the molecule of interest and how the atoms should be colored; i.e., the scene information. We break down the generation of images into two steps: (1) annotating the coordinates with the additional scene information, and (2) rendering the image. Rendering the image is separate because it is more likely to be noninteractive—it may take several minutes to compute an image depending on the type of renderer (e.g., a ray tracer), on the size of the input, and on the resolution of the generated image. In this article, a solution is presented to the first part of image generation: annotating the coordinates.

Implicit in annotating molecular coordinates is the choice of how the coordinates are represented. While molecular coordinates are available in a variety of formats, owing to our emphasis on proteins and DNA, we picked the widely accepted Brookhaven Protein Data Bank (PDB)[1] file format. Another advantage of the PDB file format is that Brookhaven is the largest source of protein coordinate files. The choice of using PDB format has not been a limitation as most small-molecule modelers can generate PDB files.

Annotating coordinates can be done with either an interactive molecule viewer or manually. Scientists much prefer to use an interactive viewer. Once the molecules are oriented, colored, and otherwise annotated, the viewer can generate an annotated PDB file as input to another program. Keeping the annotating and the rendering phases separate helps to give the scientist a variety of choices on how to render the image because different renderers can be utilized as long as they understand the annotation protocol. This separation of functionality also allows for the insertion of "filter" programs between the viewer and the renderer, thus allowing for programmatic modification of annotations before rendering.

The sample images in Color Plates 1–4 show a single annotated PDB file rendered with a variety of renderers and filters. All of the images were prepared using software that comes with the MidasPlus molecular modeling system.[2]

HOW TO ANNOTATE

To annotate a file means to embed within the original file additional information. Three typical ways of annotating files are as follows:

Metacomments
: Metacomments are interpretable information embedded in comments. Adobe's Document Structuring Conventions[3] for PostScript files is a good example of this technique. The advantage of this approach is that the annotations are just comments and thus are ignored automatically by programs that do not understand them.

Semantic extension
: Extending the semantics means adding new elements to the file language in the same syntax style. Adding new fields to a Crystallographic Information File (CIF)[4] to make them suitable for macromolecules (mmCIF)[5] is a good example of this technique. The advantage of semantic extensions is that the annotations are integrated

Color Plates for this article are on p. 193. Address reprint requests to Dr. Ferrin at the Computer Graphics Laboratory, University of California, San Francisco, California 94143-0446.
Received 1 November 1994; revised 19 January 1995; accepted 25 January 1995

655 Avenue of the Americas, New York, NY 10010

0263–7855/95/$10.00
SSDI 0263-7855(95)00003-O

directly into the language and are ignored automatically by programs that do not understand them.

Syntax modification

With syntax modification, the contents of the existing file are embedded inside a different language. The Standard Generalized Markup Language (SGML) and the HyperText Markup Language (HTML)[6] are good examples of this approach. The advantage of this technique is that the annotations are independent of the language of the original file. The disadvantage is that, unlike the previous two methods, previously working programs probably will no longer function with the new language.

By examining the syntax of PDB files, one can determine which technique is best suited for annotating these files. PDB files are composed of records, with one record per line, and an identifying keyword at the beginning of the line. At first inspection, the metacomment technique looks like it would work well as the PDB "REMARK" record type can potentially be used for embedding arbitrary text. However, the PDB standard requires that REMARK records be numbered, must contain references "in inverse chronological order," and must appear after "JRNL" records and before "SEQRES" records. These restrictions on REMARK records, therefore, make them unsuitable to use for annotating a PDB file with other arbitrary information.

Using the syntax modification technique for annotating a PDB file has the significant disadvantage that any programs that currently understand PDB format must be modified in order to understand the new language. Given the wealth of existing computational chemistry software that read and write PDB files, this approach was judged undesirable.

Our approach to the problem was to utilize the semantic extension annotation technique. This was facilitated by the existence of the PDB "USER" record, which has no predefined semantics and which should be ignored by any program that conforms to the PDB standard as specified by Brookhaven. By utilizing USER records, we avoided adding a new record type keyword for each kind of annotation and also avoided the problem that the current PDB specification reserves all unallocated keywords for future use. Thus, our approach allowed us to insert new information into PDB files in a way that is compatible with existing standard-conforming programs.

WHAT TO ANNOTATE

In addition to scene information, other useful annotations include van der Waals radii, user-defined objects (e.g., arrows and electron density contours), and state information from our interactive molecular viewer. This list is not exhaustive, but has proved sufficient for our needs.

It is also useful to place several molecules from multiple PDB files into one scene description. To accomplish this, the scene annotations are given once, and the annotated PDB records for each molecule are concatenated together with a requisite END record between consecutive molecules. In addition, the ATOM, HETATM, and TER record serial numbers continue from where the serial numbers of the previous molecule left off (this imposes a limitation of 99999 atoms).

In the following sections, each paragraph describes one annotation. Each annotation starts with the word "USER" and is followed by a keyword in column 7 that indicates the kind of annotation, and then any necessary parameters. Annotation keywords are case insensitive and parameters (e.g., filenames and label text) are case sensitive. The Fortran and C format specifications for these annotations are given in the appendix.

ANNOTATION HEADERS

The following annotations appear at the beginning of the annotated file. Graphics terminology is from Ref. 7.

USER PDBRUN *version*

This annotation gives the version number of the annotation specification used to create the annotations. The description given here corresponds to version 6. Future versions of the specification will use a different version number and presumably support additional features such as those described below (see Discussion). Although clearly desirable, higher numbered versions are not necessarily backward compatible with earlier versions of the specification. (For example, version 6 of our proposed standard is not backward compatible with version 5.)

USER EYEPOS *x y z*

USER ATPOS *x y z*

The eye position, the "look at" position, and implicit "up" vector parallel to the *y* axis are used to compute the world coordinate system to eye coordinate system transformation. Both the eye position and the look-at position are given in world coordinates.

USER WINDOW *left right bottom top hither yon*

USER FOCUS *focal-length*

The window boundaries and the focal length are used to compute the eye coordinate system to clipping coordinate system transformation. *Left, right, bottom,* and *top* are given in eye coordinates. Normally, *left* and *right* will be identical except for the sign; likewise for *bottom* and *top*. When the file describes a left eye or right eye stereo image, however, the values will be asymmetric. *Hither* and *yon* are distances of the clipping planes from the eye. The *focal-length* is the distance to the projection plane from the eye and is typically between the *hither* and *yon* distances. If the *focal-length* is zero, then an orthographic projection is used rather than a perspective projection.

USER VIEWPORT x_{min} x_{max} y_{min} y_{max}

The location of the viewport is given in the screen coordinate system that generated this scene. The dimensions may be used as image size and orientation indicators.

USER BGCOLOR *red green blue*

This annotation specifies the background color. *Red, green,* and *blue* are floating point values between 0 and 1, inclusive.

ANNOTATION TRAILERS

The following annotations appear at the end of the annotated file. These records specify information that is of particular interest to the scientist and may be highlighted by the renderer.

USER ANGLE *atom$_0$ atom$_1$ atom$_2$ atom$_3$ value*
These records specify angular information. *Atom$_0$, atom$_1$*, . . . are atom serial numbers. If *atom$_3$* is nonzero, it is a dihedral angle, otherwise a simple bond angle. *Value* is the angle in degrees.

USER DISTANCE *atom$_0$ atom$_1$ value*
These records specify distance information. *Atom$_0$* and *atom$_1$* are atom serial numbers. *Value* is the distance in angstroms.

PER MOLECULE ANNOTATIONS

The following annotations appear before each embedded PDB file. Note that the coordinates of a molecule are transformed for the particular view given by the USER EYEPOS and USER ATPOS records above.

USER FILE *model filename*
Each embedded PDB file or graphics object file is identified by the original *filename* that the original data were fetched from and by its *model* ordinal.

USER MARKNAME *name*
This annotation defines a tag that can be used to mark atoms with subsequent USER MARK records. Marks can be thought of as named selection sets.

USER MARK *name*
This annotation marks the following ATOM or HETATM record with the given *name*. Unlike the USER COLOR and USER RADIUS records (see below), marks are not inherited. A single atom may have multiple marks.

USER CNAME *red green blue color-name*
This annotation defines a named color. *Red, green,* and *blue* are floating point values between 0 and 1, inclusive. *Color-name* is an arbitrary identifier that is provided so that programs can symbolically represent color information. Color names must be defined before they are used.

USER COLOR *red green blue color-specification*
This annotation sets the color of the following ATOM or HETATM record. If a color is not given for an atom, then it inherits the value of the previous USER COLOR record. The *color-specification* is either a single color name (see USER CNAME record above) or a comma-separated triplet [color-name$_0$, color-name$_1$, interpolation-factor], where interpolation-factor is a floating point value between -1 and 1, inclusive, and the absolute value indicates the proportion of hue, lightness, and saturation (HLS) components of the two named colors.* The [*red, green, blue*] triplet is redundant (because the values are already implied by *color-specification*), but are provided as a convenience so that the *color-specification* may be ignored by any particular renderer. This record allows programs to optionally use and display the semantic information conveyed by a color name (e.g., a user can refer to the color name "slate-gray" rather than the floating point RGB color values), but also allows for simple programs to deal only with RGB color values.

USER RADIUS *radius*
This annotation sets the atomic radius of the following ATOM or HETATM record. If a radius is not given for an atom, then it inherits the value of the previous USER RADIUS record.

USER OBJECT
USER ENDOBJ
This annotation denotes the start and end of a user-defined object specification.

USER CHAIN *atom$_0$ atom$_1$*
This annotation indicates a pseudo-bond. A chain annotation is used to denote that two atoms should be shown as bonded even though they are too far apart, such as when the backbone carbons of a protein are shown connected. *Atom$_0$* and *atom$_1$* are atom serial numbers.

3D GRAPHICS ANNOTATIONS

The OpenGL 3D graphics interface[9] provides the inspiration for the set of graphics primitives described below. These annotations appear between USER OBJECT and USER ENDOBJ records.

USER GFX BEGIN *type*
USER GFX END
The graphics BEGIN and END bound a graphics primitive. Within a graphics primitive, these can be any number of vertices (see below) and optional color changes. *Type* is one of the following:

POINTS
Each vertex represents a point.

MARKERS
Markers are like points, but they are represented differently. Markers are used for singleton atoms, whereas points would be used for a molecular surface.

LINES
Every two vertices describes a line.

LINE-STRIP
Each vertex is connected to the previous vertex in a line.

LINE-LOOP
A LINE-LOOP is like a LINE-STRIP, but the last vertex is connected to the first.

TRIANGLES
Every three vertices describes a triangle.

TRIANGLE-STRIP
The first three vertices are a triangle, and each subsequent vertex forms a connected triangle with the previous two.

TRIANGLE-FAN
The first vertex is the center of a fan and the subsequent vertices form triangles around the center.

QUADS
Every four vertices describes a quadrilateral.

QUAD-STRIP
The first four vertices are a quadrilateral, and each subsequent two vertices forms a quadrilateral with the previous two.

POLYGON
The vertices form a polygon.

*Color interpolation is done using the HLS color model[8] because it provides superior interpolation results compared to an RGB color model. HLS color interpolation is as follows: $H = H_0 + |f| * (H_1 - H_0)$, $L = L_0 + |f| * (L_1 - L_0)$, $S = S_0 + |f| * (S_1 - S_0)$—where the hue is interpolated in a counterclockwise direction when the interpolation-factor, f, is positive, and clockwise when it is negative.

USER GFX COLOR *red green blue color-spec*
User-defined graphics object color; same format as the USER COLOR record above. Color records may appear outside or within graphics primitives and apply to the following primitive or vertex.

USER GFX NORMAL n_x n_y n_z
This annotation defines a normal vector for the next vertex.

USER GFX VERTEX *x y z*
This annotation defines a vertex with given coordinates.

USER GFX FONT *font-size font-name*
This annotation switches subsequent label text to the given font name and font size.

USER GFX TEXTPOS *x y z*
This annotation sets the position of the next label (see below).

USER GFX LABEL "*text*"
This annotation places a label starting at the current text position. The label text must be between double quotes. The quoted text follows the ANSI C language rules for string constants, that is, a backslash character starts an escape sequence, a backslash double quote character sequence indicates a double quote, and so on.

DISCUSSION

One might ask if our annotation scheme should be folded into the official PDB specification and specific annotations added to the PDB files from Brookhaven. While we argue that it is clearly beneficial to the molecular modeling community to standardize a set of annotations, we oppose the idea that these annotations should be present in a deposited PDB file. Scene annotations are not experimentally derived. Annotations are for illustrative purposes and their inherent subjectivity means they will potentially vary from scientist to scientist.

The advantages of our approach to annotating PDB files are as follows:

- A scientist starts with any PDB file that conforms to the PDB standard.
- An annotated PDB file is produced that still conforms to the standard and that the scientist can still easily understand.
- There are no auxiliary files to worry about. Interchanging data among scientists will not require them to collect all relevant information from a variety of sources and send them, using a variety of data formats.
- The annotated file is editable with a standard text editor or other text-processing software tools. In particular, if a interactive molecular viewer does not know how to add an annotation that a particular renderer needs, the scientist may do so manually.
- Scientists with only limited programming experience should still be able to write programs that manipulate annotations, since they consist of simple lines of text.
- Rendering is decoupled from annotating, making it easy to incorporate new scene renderers.

Notwithstanding the advantages noted above, there are limitations to our method:

- The annotations are verbose and may consume a great deal of space. However, this situation is mitigated by the omission of the most common scene annotations (i.e., USER COLOR and USER RADIUS records) when they duplicate a preceding annotation.
- The annotations listed are limited to a single scene—they do not allow multiple viewpoints to be specified.
- More types of annotations could be defined: for instance, bond order (double, triple, resonance), bond color (for ball-and-stick representations), per-residue representation styles (ribbon, ball-and-stick, CPK), extensible properties (backbone rms deviation in NMR en-

SAMPLE ANNOTATION

```
USER   PDBRUN     6
USER   EYEPOS          25.740        21.520       173.147
USER   ATPOS           25.740        21.520        10.132
USER   WINDOW        -29.271        29.271       -36.580     36.580   135.097    193.638
USER   FOCUS         163.015
USER   VIEWPORT          0.000       801.000         0.000  1001.000
USER   BGCOLOR    0.000    0.000     0.000
USER   FILE  1gcn
HEADER      HORMONE                                              17-OCT-77   1GCN
. . .
USER   COLOR    1.000    1.000     1.000 white
USER   RADIUS     1.800
ATOM       1     N   HIS         1       49.668     24.248  10.436  1.00 25.00
. . .
END
USER   FILE    1gcn
HEADER        HORMONE                                           17-OCT-77   1GCN
. . .
USER   COLOR    0.000    0.000     1.000 blue
USER   RADIUS      1.800
ATOM       1     N   HIS         1       39668      24.248  10.436  1.00 25.00
. . .
END
```

sembles, sequence homology classification, etc.), and other OpenGL[9] features (e.g., material properties, textures, and light sources). Of course, future extensions to the annotation specification can incorporate these attributes as need arises.

Other popular molecular visualization programs use different approaches to combining scene information with PDB files. We briefly discuss a few of these programs below.

Raster3D version 2.0[10] (with its companion interactive viewer) also separates interactive annotation and rendering into two phases, but abandons the PDB format in favor of a more space-efficient one, thus losing compatibility with existing programs that understand PDB format.

RasMol version 2.4[11] combines an interactive viewer with a renderer. This technique has the advantage that one program is used for both functions, but loses the flexibility of easily incorporating different renderers or being able to use filter programs to modify the annotations.

MolScript version 1.4[12] is a PostScript renderer that follows the "syntax modification" approach by embedding PDB files into its own input file format. This file format is very flexible, but has the disadvantage of being unintelligible to PDB-conforming programs.

Kinemage[13] is both a file format and an interactive viewer for personal computers. Its chief advantage is that it supports multiple views and bond rotations, but its data files are not in PDB format and are also difficult to edit.

CONCLUSION

The annotation method we describe has been used successfully for the past 5 years as part of the MidasPlus molecular modeling system. C and C++ libraries for parsing standard PDB records as well as the annotations described above are freely available from our laboratory.[14]

We have attempted to address only the annotation of PDB-formatted files. Analogous annotation conventions should be established for other molecular file formats. In particular, we are interested in working with other groups and standardizing a similar set of extensions for mmCIF.

ACKNOWLEDGMENTS

This work was supported by the NIH National Center for Research Resources, Grant 2P41RR-01081, and by the National Science Foundation, Grant IRI-9116999. We thank Julie Newdoll for her assistance in preparing the Color Plates.

APPENDIX: ANNOTATION RECORD FORMATS

For the C format specification, the leading `USER` **is left out, and the format given below starts in column 7. The Fortran formats include skipping over the word** `USER`**. Annotations marked with a dagger (†) are required.**

Annotation	C format	Fortran format
	Annotation headers	
PDBRUN†	`PDBRUN %2d`	6A1,6A1,X,I2
EYEPOS†	`EYEPOS %9.3f %9.3f %9.3f`	6A1,6A1,3(X,F9.3)
ATPOS†	`ATPOS %9.3f %9.3f %9.3f`	6A1,5A1,X,3(X,F9.3)
WINDOW†	`WINDOW %9.3f %9.3f %9.3f %9.3f %9.3f %9.3f`	6A1,6A1,6(X,F9.3)
FOCUS†	`FOCUS %9.3f`	6A1,5A1,2X,F9.3
VIEWPORT	`VIEWPORT %9.3f %9.3f %9.3f %9.3f`	6A1,8A1,3(XF9.3)
BGCOLOR	`BGCOLOR %5.3f %5.3f %5.3f`	6A1,7A1,3(X,F5.3)
	Annotation trailers	
ANGLE	`ANGLE %6d %6d %6d %9.3f`	6A1,5A1,4(X,I6),X,F9.3
DISTANCE	`DISTANCE %6d %6d %9.3f`	6A1,8A1,2(X,I6).X,F9.3
	Per-molecule annotations	
FILE†	`FILE %4d %-.56s`	6A1,4A1,X,I4,X,56A1
MARKNAME	`MARKNAME %-.57s`	6A1,8A1,X,57A1
MARK	`MARK %-.57s`	6A1,4A1,X,57A1
CNAME	`CNAME %5.3f %5.3f %5.3f %-.38s`	6A1,5A1,3(X,F5.3),X,38A1
COLOR	`COLOR %5.3f %5.3f %5.3f %-.38s`	6A1,5A1,3(X,F5.3),X,38A1
RADIUS	`RADIUS %7.3f`	6A1,6A1,X,F7.3
OBJECT	`OBJECT`	6A1,6A1
ENDOBJ	`ENDOBJ`	6A1,6A1
CHAIN	`CHAIN %6d %6d`	6A1,5A1,2(XI6)
	3D graphics annotations	
GFX BEGIN	`GFX BEGIN %-.32s`	6A1,9A1,32A1
GFX END	`GFX END`	6A1,7A1
GFX COLOR	`GFX COLOR %5.3f %5.3f %5.3 %-.38s`	6A1,9A1,3(X,F5.3),X,38A1
GFX NORMAL	`GFX NORMAL %9.3f %9.3f %9.3f`	6A1,10A1,3(X,F9.3)
GFX VERTEX	`GFX VERTEX %9.3f %9.3f %9.3f`	6A1,10A1,3(X,F9.3)
GFX FONT	`GFX FONT %3d %-.53s`	6A1,8A1,X,I3,X,53A1
GFX TEXTPOS	`GFX TEXTPOS %9.3f %9.3f %9.3f`	6A1,9A1,3(X,F9.3)
GFX LABEL	`GFX LABEL "%-,56s"`	6A1,9A1,X,56A1

REFERENCES

1 Abola, E., Bernstein, F., Bryant, S., Koetzle, T., and Weng, J. Protein Data Bank. In *Crystallographic Databases—Information Content, Software Systems, Scientific Applications* (F. Allen, G. Bergerhoff, and R. Seivers, eds.). Data Commission of the Intl. Union of Crystallography, Bonn/Cambridge/Chester, 1987, pp. 107–132. Available via the World-Wide Web in http://www.pdb.bnl.gov

2 Ferrin, T., Huang, C., Jarvis, L., and Langridge, R. The MIDAS display system. *J. Mol. Graphics* 1988, **6**(1), 13–27, 36–37. The MidasPlus software package is licensed by the Regents of the University of California; for additional information please contact T. Ferrin or access the World-Wide Web URL http://cgl.ucsf.edu/midasplus.html

3 Adobe Systems Incorporated (ed.). Appendix G: Document structuring conventions—version 3.0. In *PostScript Language Reference Manual,* 2nd Ed. Addison-Wesley, Reading, MA, 1990, pp. 611–708. Available via anonymous FTP from ftp.adobe.com in /pub/adobe/DeveloperSupport/TechNotes/5001.DSC_Spec_v3.0.ps

4 Hall, S., Allen, F., and Brown, I. The crystallographic information file (CIF): A new standard activity file for crystallography. *Acta Crystallogr. Sect. A* 1991, **47,** 655–685. Available via anonymous FTP from ftp.iucr.ac.uk

5 Bourne, P. (ed.). *Proc. of the First Macromolecular Crystallographic Information File (CIF) Tools Workshop*. Department of Biochemistry and Molecular Biophysics. Columbia University, New York, NY, Oct 15–18, 1993. Available via anonymous FTP from ftp.pdb.bnl.gov in /pub/Cif/mmCif.dic-proposed-standard

6 Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H., Secret, A. The World-Wide Web. *Commun ACM* 1994, **37**(8), 76–82. Available via the World-Wide Web in http://info.cern.ch/hypertext/WWW/TheProject.html

7 Foley, J., van Dam, A., Feiner, S., and Hughes, J. *Computer Graphics: Principles and Practice,* 2nd Ed. Addison-Wesley, Reading, MA, 1990.

8 Foley, J., van Dam, A., Feiner, S., and Hughes, J. *Computer Graphics: Principles and Practice* 2nd Ed. Addison-Wesley, Reading, MA, 1990, pp. 592–599

9 Rogelberg, D., and Fullagar, J.C. (ed.). *OpenGL Reference Manual.* Addison-Wesley, 1992. Available via the World-Wide Web in http://www.sgi.com/tech/openGL/opengl.html

10 Merritt, E., and Murphy, M. Raster3D version 2.0: A program for photorealistic molecular graphics. *Acta Crystallogr. Sect. D_J* 1994, **50,** 869–873. Available via anonymous FTP from stanzi.bchem.washington.edu in /pub/raster3d/raster3d.html

11 Sayle, R. and Bissell, A. *RasMol: A program for fast realistic rendering of molecular structures with shadows.* November 1993. Available via anonymous FTP from ftp.dcs.ed.ac.uk in /pub/rasmol/

12 Kraulis, P. Hot papers—(molecular biology)—MolScript—a program to produce both detailed and schematic plots of protein structures. *Scientist* 1993, **7**(2), 16. http://www-bio.unizh.ch/visualization/molscript/molscript.html

13 Richardson, D. and Richardson, J. The Kinemage—a tool for scientific communication. *Protein Sci.* 1992, **1**(1), 3–9. Available via anonymous FTP from suna.biochem.duke.edu in /pub/MACprograms/MAGE-2.5

14 Couch, G. *C and C++ Libraries for Parsing PDB Records,* 1994. Available via anonymous FTP from cgl.ucsf.edu in /pub/libpdb.shar and /pub/libpdb++.shar

15 Finer-Moore, J., Fauman, E., Foster, P., Perry, K., Santi, D., and Stroud, R. Refined structures of substrate-bound and phosphate-bound thymidylate synthase from *Lactobacillus casei. J. Mol. Biol.* 1993, **232** (4), 1101–1116

16 Huang, C., Pettersen, E., Klein, T., Ferrin, T., and Langridge, R. Conic—a fast renderer for space-filling molecules with shadows. *J. Mol. Graphics,* 1991, **9**(4), 230–236, 242

17 Huang, C., Pettersen, E., Couch, G., and Ferrin, T. Stylized representations of nucleotide ribbons. (*in preparation*).

18 Technical report. In *UCSF MidasPlus User's Manual.* Computer Graphics Laboratory, University of California, San Francisco, CA, 1994. Available via the World-Wide Web in http://cgl.ucsf.edu/midasplus.html