International Tables for Crystallography (2006). Vol. F, Section 25.2.7, pp. 725–727.

required to be in PDB format. An additional input file is the parameter file that governs which plots are to be generated and deals with certain aspects of their appearance.

25.2.6.6. *Output produced*

The output of the program consists of a number of PostScript plots, together with a full listing of the individual parameter values for each residue, with any unusual geometrical properties highlighted. The listing also provides summaries for the protein as a whole. Figs. 25.2.6.2 and 25.2.6.3 show parts of one of the PostScript plots generated, showing the variation of various residue properties along the length of the protein chain. Unusual regions, which are highlighted on these plots, may require further investigation by the crystallographer.

25.2.6.7. *Other validation tools*

PROCHECK is merely one of a number of validation tools that are freely available, some of which are mentioned elsewhere in this volume. The best known are *WHATCHECK* (Hooft *et al.*, 1996), *PROVE* (Pontius *et al.*, 1996), *SQUID* (Oldfield, 1992) and *VERIFY3D* (Eisenberg *et al.*, 1997). Tools such as *OOPS* (Kleywegt & Jones, 1996*b*) or the X-build validation in *QUANTA* (MSI, 1997) provide standard tests on the geometry of a structure and provide lists of residues with unexpected features, which make it easy to check electron-density maps at suspect points.

25.2.7. *MolScript* (P. J. Kraulis)

25.2.7.1. *Introduction*

Visualization of the atomic coordinate data obtained from a crystallographic study is a necessary step in the analysis and interpretation of the structure. The scientist may use visualization for different purposes, such as obtaining an overview of the structure as a whole, or studying particular spatial relationships in detail. Different levels of graphical abstraction are therefore required. In some cases, the atomic details need to be visualized, while in other cases, high-level structural features must be displayed.

In the study of protein 3D structures in particular, there is an obvious need to visualize structural features at a level higher than atomic. A common graphical 'symbolic language' has evolved to represent schematically hydrogen-bonded repetitive structures (secondary structure) in proteins. Cylinders or helical ribbons are used for α-helices, while arrows or ribbons show strands in β-sheets.

The domain of the *MolScript* program is the production of publication-quality images of molecular structures, in particular protein structures. The implementation of *MolScript* is based on two design principles: First, the program must allow both schematic and detailed graphical representations to be used in the same image. Second, the user must be able to control the precise visual appearance of the various graphics objects in as much detail as possible.

The original version of *MolScript* was written in Fortran77 and produced only PostScript output (Kraulis, 1991). The current version (v2.1.2, as of January 1999) has been completely rewritten in the C programming language. The new version is almost completely compatible with previous versions. The main new features in version 2 are several new output formats, the interactive OpenGL mode (see below) and dynamic memory allocation for all operations.

This section reviews the basic features of *MolScript*. Detailed information about the program, including instructions on how to obtain the software, can be found at the official *MolScript* web site http://www.avatar.se/molscript/, where the online manual is also available.

25.2.7.2. *Input*

The input to *MolScript* consists of the coordinate file(s) in standard PDB format and a script file which describes the orientation of the structures, the graphics objects to display and the graphics state parameters that control the visual appearance of the objects.

The script may be created automatically by the utility program *MolAuto* (see below), or manually by the user in a standard text editor. The script may invoke other external script files or command macros, and it may also contain in-line atomic coordinate data.

25.2.7.3. *Graphics*

The basic model for the execution of *MolScript* is that of a non-interactive image-creating script processor. There are two stages in the execution. First, the script is parsed and the graphics objects are created according to the commands. This stage is essentially independent of the output format. Second, when the end of the script has been reached, the image is rendered from the graphics objects according to the chosen output format.

25.2.7.3.1. *The coordinate system*

The viewpoint in the *MolScript* right-handed coordinate system is always located on the positive z axis, looking towards the origin, with the positive x axis to the right. The user obtains the desired view of the structure by specifying rotations and translations of the atomic coordinates; it is not possible to change the location or the direction of the viewpoint.

There are two main benefits with this scheme: The first is that it is similar to the way we handle objects in everyday life: we do not normally fly around the object, but rather move it about with our hands. The other benefit is that together with the coordinate copy feature it can be used to compose an image containing several geometrically related subunits.

The disadvantage is that the atomic coordinates must be transformed before the creation of the graphics objects. This may complicate the composition of an image where another structure or geometric object is to be included. For example, if two separate structures have been aligned structurally by some external procedure, then the user must take care not to destroy the alignment in the process of setting the viewing transformation.

25.2.7.3.2. *The graphics state*

The graphics state consists of the parameters that determine the exact visual appearance of the graphics objects. The default values of the graphics state parameters are reasonable, so that an image of acceptable quality can be produced quickly. However, to obtain high-quality images which emphasize the relevant structural features, the user must usually fine-tune the rendering by modifying the graphics state parameters appropriately for the various graphics objects.

The graphics state may be modified by using the 'set' command at any point in the script file. The change can have an effect only on graphics commands below that point in the script. When a graphics command is processed, the object is created according to the current values of the graphics state parameters at that exact point in the script file. It is this property of the graphics state that gives the user a very high degree of control over the composition and appearance of the graphics objects.

A new feature in version 2 of *MolScript* is the ability to set the colour of residues on a residue-by-residue basis in schematic secondary-structure representations. It is also possible to set the colour of atoms and residues according to a linear function of the B factors.

25.2.7.3.3. *Graphics commands*

The graphics commands create the graphics objects to be rendered in the final image. The commands need an atom or residue selection (see below) as argument. The visual attributes, and in some cases the dimensions, of the objects are determined by the graphics state parameters.

The graphics objects include the most common ways to represent atoms, such as simple line drawings, ball-and-stick models and CPK (Corey–Pauling–Koltun) spheres approximating the van der Waals radii of the atoms. The graphics objects representing high-level structures are mainly designed for protein structures, and comprise arrows for β-sheet strands, cylinders or helical ribbons for α-helices, and coils for non-repetitive peptide chain structures. The coil object can also be used to represent oligonucleotide backbone structures.

25.2.7.3.4. *Atom and residue selection*

A set of basic atom and residue selections are provided in *MolScript* for use as arguments to the graphics commands. Arbitrary subsets of atoms or residues can be specified by joining together the basic selections using a form of Boolean operators. Unfortunately, the Boolean expression feature may sometimes be difficult to understand for the non-expert user. One should consider the entire expression as a test to be applied to every atom or residue. Any atom or residue for which the Boolean expression evaluates to 'true' will be selected as argument for the command.

25.2.7.3.5. *External objects*

Externally defined objects described by points, lines or triangular surfaces may be included in the image. The objects may optionally be transformed by the most recent transformation applied to the coordinate data. This feature allows import of arbitrary geometry created by some external software, *e.g.* molecular surfaces, electron-density representations or electrostatic field lines. The graphics state parameters apply to the rendering of the external objects in the image.

25.2.7.4. *Output*

The current implementation of the *MolScript* source code makes it possible to add new output formats. The intention is that all output formats should produce visually identical images given the same input. Unfortunately, this goal is hard to achieve due to various technical issues, such as the different formalisms used to describe lighting and material properties.

25.2.7.4.1. *PostScript*

PostScript (Adobe Systems Inc., 1985) is a page description language for controlling high-quality printers. More information can be found at the Adobe Inc. web site, http://www.adobe.com/print/.

The PostScript output mode relies on the painter's algorithm for hidden-surface removal. The most distant graphics segments are output first, continuing with the segments closer to the viewpoint, which may obliterate previously rendered segments. The implementation of this procedure is straightforward, and gives good results provided that the graphics objects are subdivided into sufficiently small segments. The PostScript mode allows more than one plot (image) to be rendered on a single page.

25.2.7.4.2. *Raster3D*

The *Raster*3*D* suite of programs (Merritt & Bacon, 1997) produces high-quality images using a ray-tracing algorithm. *MolScript* can produce the input file required for the 'render' program, which is the core program of the *Raster*3*D* suite. The web site for *Raster*3*D* is http://www.bmsc.washington.edu/raster3d/.

The *Raster*3*D* mode features highlighting, transparency and shadows to produce *MolScript* images of very high visual quality.

25.2.7.4.3. *VRML97*

The VRML97 standard (Virtual Reality Modeling Language, formerly VRML 2.0) allows storage and transmission of 3D scenes in a system-independent manner over the web. Software to view VRML97 files is typically included in any modern web browser. A web site containing more information on VRML97 is http://www.vrml.org/.

The VRML97 mode allows hyperlinking of objects. The *MolScript* implementation is optimized to produce output files that are as small as possible, but the file size is strongly dependent on the value of the 'segments' parameter in the graphics state.

25.2.7.4.4. *OpenGL*

OpenGL is a standard API (Applications Programming Interface) for interactive 3D graphics. It is available on most current computer systems. For more information, see the web site http://www.opengl.org/.

The OpenGL output mode allows a certain degree of interactivity, in contrast to the other output modes. It is possible to initiate execution of the *MolScript* program in OpenGL mode in one window on the screen, while keeping the script file in a separate text-editor window. The image is rotatable in 3D in the OpenGL window. The script can be edited in its window, and the modified script can be re-read and displayed directly by the *MolScript* program in its OpenGL window. This simplifies to some extent the iterative fine-tuning of the script.

25.2.7.4.5. *Image files*

Raster image files in several different formats can be created by *MolScript*. Currently these include SGI RGB, encapsulated PostScript (EPS), JPEG, PNG and GIF image formats. The JPEG, PNG and GIF formats require that external software libraries are available during the compilation and linking of the *MolScript* program. Software libraries for several of these image formats are available on the web; links are given at the official *MolScript* web site http://www.avatar.se/molscript/.

The image file formats essentially capture the raster image created by the OpenGL implementation. The EPS format was a variant of the PostScript output mode in version 1 of *MolScript*, but for various reasons this has changed in version 2 to an encoding of the OpenGL raster image.

25.2.7.5. *Utilities*

A utility program called *MolAuto* is included in the *MolScript* software distribution. It reads a standard-format PDB coordinate file to produce a first-approximation script file for *MolScript*. This is a simple way to produce a starting point for further manual editing.

The *MolAuto* and *MolScript* programs have been designed to work well as software tools in the UNIX environment. This allows the programs to be embedded in more comprehensive software systems for automated creation and/or storage of images. An example of such a system is the web interface to the RCSB Protein Data Bank (PDB, http://www.rcsb.org/pdb/), which employs *Mol-*

Script (among other tools) for visualization of the coordinate data sets.

25.2.8. *MAGE*, *PROBE* and kinemages

(D. C. Richardson and J. S. Richardson)

25.2.8.1. *Introduction to aims and concepts*

MAGE and the kinemages it displays (Richardson & Richardson, 1992, 1994) provide molecular graphics, organized in an unusual way, that are of interest to crystallographers for uses that range from interactive illustrations for teaching to a representation of all-atom van der Waals contacts, calculated by *PROBE* (Word, Lovell, LaBean *et al.*, 1999), to help guide model-to-map fitting.

A kinemage ('kinetic image') is an authored interactive 3D illustration that allows open-ended exploration but has viewpoint, explanation and emphasis built in. A kinemage is stored as a human-readable flat ASCII text file that embodies the data structure and 3D plotting information chosen by its author or user. *MAGE* is a pure graphics display program designed to show and edit kinemages, while *PREKIN* constructs molecular kinemages from PDB (Protein Data Bank; Research Collaboratory for Structural Bioinformatics, 2000) files. The latest versions (currently 5.7) of *MAGE* and *PREKIN* are available free for Macintosh, PC, Linux, or UNIX from the kinemage web site (Richardson Laboratory, 2000). The programs operate very nearly equivalently on different platforms and, by policy, later versions of *MAGE* can display all older kinemages. A Java 'Magelet' can show small kinemages directly in suitable web browsers, with their first-level interactive capabilities of rotation, identification, measurement, views and animation.

MAGE has no internal knowledge of molecular structure. A collaboration between the author and the authoring program (*e.g. PREKIN*) builds data organization into the kinemage itself. This two-layer approach has great advantages in flexibility, since an author can show things the programmer never imagined, including non-molecular 3D relationships. Overall, kinemages demand less work and less expertise from the reader or viewer than do traditional graphics programs, but that ease of use depends on the effort involved in thoughtful authoring choices, aided by the extensive on-screen editing capabilities described below.

MAGE has been designed to optimize visual comprehension: the understanding and communication of specific 3D relationships inside complex molecules. Display speed has been given priority, to ensure good depth perception from smooth real-time rotation. The interface is extremely simple and transparent, and the colour palette is tuned for comparisons, contrasts and depth cueing. Immediate identification and measurement are always active; views, animations, or bond rotations can be built in by the kinemage author. Text and caption windows explain the intentions of the author, while a simple hypertext capability allows the reader to jump to the specific view and display objects being described; however, most kinemages can also be successfully understood just by exploring what is available within the graphics window.

Kinemages are suitable for structure browsing or producing static 2D presentation graphics, but those aspects have been kept secondary to effectiveness for interactive visualization and flexibility of author specification. Features and representations have deliberately been chosen to be fast, simple and informative rather than either showy or traditional, as illustrated by the following examples and their rationales. Mouse-controlled rotation in *MAGE* depends only on the direction of drag, so that the behaviour of the image is independent of absolute cursor position within the window. Labels are available but seldom needed, since the data structure builds in a 'pointID' that is displayed whenever the point is picked. Instead of using half-bond colouring which tends to chop up the image, *PREKIN* provides separate colours and button controls for main chain *versus* side chains, and it can prepare a partial 'ball-and-stick' representation with colour-coded balls on non-carbon, non-hydrogen atoms (see Fig. 25.2.8.1). Hydrogen atoms are crucial for some research uses, but to minimize the clutter from twice as many atoms, *PREKIN* sets up their display under button control; in addition, a 'lens' parameter can be specified for the list, allowing display only within a radius of the last picked centre point. For effective perception of conformational change, while avoiding either the confusion of overlays or the potential misrepresentation of computed interpolation, *MAGE* features simple animation switching between known conformations. Very importantly, since molecular information resides mostly in chemical bonds and spatial proximity, kinemages emphasize fully 3D representations, such as vectors, dots, or 'ball and stick's, rather than surface graphics that obscure internal structure. A space-filling representation (the 'spherelist') is available, but it is suggested that it is used very sparingly – for example, to show the size and shape of a small-molecule ligand. If an extensive surface is needed, a dot surface is more informative, since the underlying atoms and bonds can be seen at the same time. Nothing matches a well rendered ribbon for conveying overall 'fold'; *PREKIN* calculates and *MAGE* displays simple ribbon schematics (see Fig. 25.2.8.2) which can be rendered by *Raster3D* (Merritt, 2000) or *POV-Ray* (POV-Ray Team, 2000) for a static 2D illustration, but for interactive use they serve mainly as introduction and context for more detailed 'ball-and-stick', vector and dot representations.

For kinemages, the representation style is not a global choice that applies to everything shown, but rather is a set of local options (varied across space or sequence) chosen to provide appropriate emphasis and comprehensible detail within context.

25.2.8.2. *Use as a reader of existing kinemages*

Viewing a pre-existing kinemage file requires almost no learning process: the interface is sufficiently 'transparent' that interaction is mainly with the molecule rather than with the program. Six simple operations cover all basic functionalities: (1) drag with the mouse to rotate the displayed object; (2) click on a point to identify it; (3) turn things on or off, or animate if that option is present, with labelled buttons; (4) choose preset views from the Views pull-down menu; (5) read the author's explanations in the text and caption windows; (6) change to the next kinemage in the file with the Kinemage pull-down menu. At a slightly more complex level, one can recentre, zoom the scale, move the clipping planes and save a view; measure distances, angles and dihedrals or 'Find' by point name (from the Tools pull-down menu); change Display menu options such as stereo or perspective; or consult the Help menu. There are keyboard shortcuts for convenience (such as 'a' to animate or 'c' for cross-eye *versus* wall-eye stereo), but they are never the only method and they are defined on the menus. Demo5_4a.kin (Richardson Laboratory, 2000) provides a brief guided introduction to using kinemages.

25.2.8.3. *Use for teaching*

Simplicity of interface, attention to presentation issues and free cross-platform availability make *MAGE* and kinemages especially well suited for teaching and learning about macromolecular structure or about crystallographic concepts such as handedness and symmetry. Suggestions can be found in Richardson & Richardson (1992, 1994) and in file KinTeach.txt (Richardson Laboratory, 2000). A large body of teaching material is available in kinemage form, including supplements for textbooks on protein structure and general biochemistry, the Protein Tourist files and kinemages for specific papers in *Protein Science*, and a great many web sites involving kinemages, some of which contain course

references