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 *Raster3D* 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 *Raster3D* suite. The web site for *Raster3D* is http://www.bmsc.washington.edu/raster3d/.

The *Raster3D* 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-*

references