connected line segment, ‘auger’ removes everything within a marked circle on the screen and ‘undo p’ recovers from a mistake, back for ten steps. If, for example, side chains are being shown in a focus around the active site, one could prune away those that don’t interact at all, and then move the second-shell side chains to a separate list with the word ‘off’ in its first line. ‘Text Editable’ (Edit menu) enables writing explanations in the text and caption windows, while the graphics window is still active for reference. ‘Save As’ (File menu) will save the whole edited kinemage file and reload to show the revised kinemage in its startup view. As well as a bitmap screen capture or files for rendering, a PostScript file can also be written to print out a 2D picture of the current graphics window, either in colour or ‘black on white’.

At this stage, a word processor can be used to look at the plain ASCII kinemage file, with its text, its views and the hierarchy of group, subgroup and list display objects in human-readable and clearly identified forms. Lists (*e.g.* @vectorlist {name}) can be of vectors, dots, labels, words, balls, spheres, triangles, or ribbons. Any part of the file can be edited, using its existing format as a guide or looking at another kinemage file that provides a desired template. Among the few operations that currently must be edited outside rather than inside *MAGE* are moving things between different lists or groups (for instance, setting up a new list of just active-site side chains in a different colour and controlled by their own button) and adding ‘master’ buttons that control object display independent of the group heirarchy (*e.g.* side chains can be turned off and on together for all subunits or models if ‘master = {side ch}’ is added to the first line of each of those lists). The kinemage should be saved without formatting, as a plain text file.

More complex modifications are possible in *MAGE*, using advanced on-screen editing and construction features from the Edit menu. ‘Draw new’ activates tools that can add labels, draw hydrogen bonds (with shortened, unselectable lines) and make a variety of geometrical constructs by building out from the original atoms (*e.g.* add a Cβ to a Gly, or draw helix axes and measure their distance and angle). ‘Show Object Properties’ lets one see, and edit, the names and parameters of the object hierarchy for any point picked, which allows renaming buttons, simplifying the button panel, adding animation, editing labels, or deleting entire display objects. ‘Remote Update’, on the Tools menu, can call *PREKIN* to set up rotations for the last-picked side chain or a mutation of it, and can then call *PROBE* to update all-atom contacts interactively as the angles are changed. On the kinemage web site (Richardson Laboratory, 2000), Demo5_4a.kin includes an introduction to the drawing tools and Demo5_4b.kin to the format and to editing. Make_kin.txt is a more complete tutorial on the process of constructing kinemages. Mage5_4.txt and Pkin5_4.txt document the features of the *MAGE* and *PREKIN* programs. File KinFmt54.txt (which also constitutes the MIME standard chemical/x-kinemage) is a formal description of the kinemage format for 3D display.

All in all, making a simple kinemage is trivial, but making really good ones for use by others is much like making a good web page. There are tools that make the individual steps easy, but one needs to exercise restraint to keep it simple enough to be both fast and comprehensible, patience to keep looking at the result and modifying it where needed, and judgment about both content and aesthetics.

25.2.8.7. *Software notes*

MAGE and *PREKIN* were written in C for Macintosh, PC, Linux, SGI and other UNIX platforms by David C. Richardson, who also maintains and extends them (with the help of Brent K. Presley for the Windows 95/98/NT port). *PROBE* (in C) and *REDUCE* (in C++) were written by J. Michael Word for SGI UNIX, Linux and PC Windows, but can be compiled on other platforms. The contact-dot additions to *O* and *XtalView* were written by Simon C. Lovell, J. Michael Word and Duncan E. McRee. For the modified *XtalView* (version 4.0), see http://www.scripps.edu/pub/dem-web; for *O* scripts and files, see http://origo.imsb.au.dk/~mok/o; the rest of the software, plus source and documentation files, is available free from the kinemage web or ftp site (Richardson Laboratory, 2000).

25.2.9. *XDS* (W. Kabsch)

25.2.9.1. *Functional specification*

The program package *XDS* (Kabsch, 1988*a*,*b*, 1993) has been developed for the reduction of single-crystal diffraction data recorded on a planar detector by the rotation method using monochromatic X-rays. It includes a set of five programs:

(1) *XDS* accepts a sequence of adjacent, non-overlapping rotation images from a variety of imaging plate, CCD and multiwire area detectors and produces a list of corrected integrated intensities of the reflections occurring in the images. The program assumes that each image covers the same positive amount of crystal rotation and that rotation axis, incident beam and crystal intersect at one point, but otherwise imposes no limitations on detector position, or directions of rotation axis and incident beam, or on the oscillation range covered by each image.

(2) *XPLAN* provides information for identifying the optimal rotation range for collecting data. Based on detector position and unit-cell orientation obtained from evaluating one or a few rotation images using *XDS*, it reports the expected completeness of the data by simulating measurements at various rotation ranges specified by the user, thereby taking into account already-measured reflections.

(3) *XSCALE* places several data sets on a common scale, optionally merges them into one or several sets of unique reflections, and reports their completeness and quality of integrated intensities.

(4) *VIEW* displays rotation-data images as well as control images produced by *XDS*. It is used for checking the correctness of data processing and for deriving suitable values for some of the input parameters required by *XDS*. This program was coded in the computer language C by Werner Gebhard at the Max-Planck-Institut für medizinische Forschung in Heidelberg. The other programs are written in Fortran77, with the exception of a few C subroutines provided by Abrahams (1993) for handling compressed images.

(5) *XDSCONV* converts reflection data files as obtained from *XDS* or *XSCALE* into various formats required by software packages for crystal structure determination. Test reflections previously selected for monitoring the progress of structure refinement may be inherited by the new output file, which simplifies the use of new data or switching between different structure-determination packages.

25.2.9.2. *Components of the package*

25.2.9.2.1. *XDS*

XDS is organized into eight steps (major subroutines) which are called in succession by the main program. Information is exchanged between the steps by files (see Table 25.2.9.1), which allows repetition of selected steps with a different set of input parameters without rerunning the whole program. ASCII files can be inspected and modified using a text editor, whereas types DIR and BIN indicate binary random access and unformatted sequential access files, respectively. All files have a fixed name defined by *XDS*, which makes it mandatory to process each data set in a newly created directory. Clearly, one should not run more than one *XDS* job at a time in any given directory. Output files affected by

references