17. MODEL BUILDING AND COMPUTER GRAPHICS

17.1. Around *O*

BY G. J. KLEYWEGT, J.-Y. ZOU, M. KJELDGAARD AND T. A. JONES

17.1.1. Introduction

The first protein structures to be solved were built as wire models. It was only at the end of the 1970s and the beginning of the 1980s that the necessary hardware and software became available to allow crystallographers to construct their models using computers. The first commercially available computer graphics systems were very expensive, and by today's standards rather primitive in their line-drawing capabilities. They were usually controlled by mini-computers loaded with 64k words of memory and removable disks capable of holding 1 Mbyte of data. The limited amount of addressable memory was a severe limitation in software production. Furthermore, each computer graphics system had its own graphics programming library that was totally incompatible with those of other systems. Despite these limitations, the benefits of using computer-graphics-based systems became apparent and they were fairly rapidly adopted by the laboratories that could afford them. The main benefits were not in the construction of the initial model, but rather as a tool in crystallographic refinement (Jones, 1978). Making small manual changes to a model being refined was difficult and time-consuming work, but rather easy to accomplish even with low-powered graphics systems.

The most widely used program of the early days, *Frodo*, was available on most of the contemporary computer graphics systems. A major step forward occurred with the development of laboratory-scale 32-bit computers with virtual memory operating systems. In particular, the first Digital Equipment VAX models rapidly became the machines of choice in crystallographic laboratories. This allowed the contemplation of real-time improvements in models under construction (Jones, 1982; Jones & Liljas, 1984). Soon afterwards, colour became available in commercial graphics systems. This was much more than a cosmetic enhancement, since colour could be used to convey information vital to the crystallographer, such as main-chain/side-chain status codes for skeletonized electron density (Jones & Thirup, 1986). Unfortunately, there was still no common graphics programming standard, and moving to a new graphics system was a major effort (Pflugrath *et al.*, 1984).

The next major advance in hardware occurred with the development of the workstation, combining the computer and graphics in one package. Although pioneered by Sun, the major player in the crystallographic community was a small Californian company, Silicon Graphics, which rapidly became large. Running the Unix operating system, workstations flourished, but still lacked a graphics environment that was portable between different hardware platforms. This changed when OpenGL was adopted as an industry standard. At the same time, prices stabilized and began to drop in terms of price/performance. Only in the late 1990s have price/performance indicators plummeted with the arrival of PC/graphics-board combinations capable of meeting the expectations of the current generation of crystallographers. The crystallographic workstation on every desk has finally arrived.

17.1.2. *O*

O was designed by Alwyn Jones to overcome some of the drawbacks associated with using *Frodo*. These problems had arisen because of the history of the program. In particular, *O* was designed to use a general-purpose memory allocation system to store any kind of model-related data. This would allow the display of any number of molecules and the use of databases for modelling. Although the latter had been introduced in a *Frodo* variant (Jones & Thirup, 1986), *O* was designed to take the concept of database use further, some would say to the extreme. Furthermore, *O* was designed to make it easier for different developers to work on the code without interfering with each other. In the event, only Morten Kjeldgaard and Jin-Yu Zou made any developments with the program.

Much of the data used by *O* are kept in a memory allocation system, the *O* database. This database is used to save parameters used by the program (including such things as keywords), macromolecular coordinates and information derived from them (such as graphical objects). As such, the program has no built-in limitations concerning what can be saved and used. A set of coordinates can be downloaded from the Protein Data Bank (PDB) (Bernstein *et al.*, 1977) and stored as a series of vectors that describe the sequence, the residue names, the coordinates, the atom names, the unit cell *etc*. Some of these vectors contain residue-related data (*e.g.* the sequence), others contain atomic data (*e.g.* the atomic temperature factors), while yet others concern the molecule as a whole. The program therefore uses a strict naming convention in handling these data. Each molecule has a name, and the program then forces its own nomenclature for the standard atomic, residue and molecular properties. The user remains free to create new data outside *O*, bring them into the program by adopting the naming convention, and then make use of them to generate or manipulate graphical images. For example, a series of amino-acid sequences can be aligned with a computer program outside *O* and information on the degree of sequence conservation can be generated as a series of *O* data blocks. These can then be read into *O* and used to colour a Cα trace of a model, for example.

The program also has a strong macro capability that can be used to configure quite complex interactive tasks. It can also be used by a programmer outside *O* to generate data and a series of instructions for later interactive use.

Similarly, data generated within *O* can be exported to *O*-aware programs, significantly reducing the complexity of some crystallographic calculations. For example, real-space averaging of electron-density maps requires, as a minimum, both a series of operators describing the noncrystallographic symmetry (NCS) and a mask. These can be generated from scratch in *O* and improved and used by *O*-aware programs without the crystallographer needing to be concerned about the myriad details of axis definitions, rotations and translations.

Plotting is carried out within the *O* system *via* an intermediate metafile. When a user creates objects for display within *O*, calls are made to a set of low-level routines that create the OpenGL instructions on the workstation. Some objects are described in their entirety within the *O* database, but others are not. Molecular objects fall into the former category, whereas electron-density maps fall into the latter. There are two sets of plot commands, therefore, that are appropriate for each class. To plot an object made from a molecule, the user merely issues a *plot_object* command, and the appropriate metafile is written out, complete with viewing data. To plot other things, the user activates the *plot_on* command and then starts creating objects. Every time a low-level graphics routine is

Copyright © 2006 International Union of Crystallography

references