An affordable approach to interactive desktop molecular modeling

T. E. Ferrin, G. S. Couch, C. C. Huang, E. F. Pettersen and R. Langridge

Computer Graphics Laboratory, Department of Pharmaceutical Chemistry, University of California, San Francisco, CA, USA

The amazing revolution in computer hardware performance and cost reduction has yet to be carried over to computer software. In fact, application software today is often more expensive and less reliable than the hardware. New enhancements in software development techniques, such as object oriented programming and interactive graphics based user interface design, finally may be having a significant impact on the time-to-market and reliability of these application programs. We discuss our experiences using one such set of software development tools available on the NeXT workstation and describe the effort required to port our MidasPlus molecular modeling package to the NeXT workstation.

Keywords: interactive molecular graphics, desktop molecular modeling, NeXT workstation

INTRODUCTION

The historical trend in computing of doubling processor speeds every four years is no longer an accurate description of the industry. Instead, one finds today that due largely to unprecedented levels of integration, many vendor product lines are doubling their performance every 1.5 to 2 years. For example, a 25-MHz Motorola 68040 processor is rated at 20 million instructions per second (MIPS) and 3.5 million floating point operations per second (MFLOPS).[1,2] This represents a threefold improvement in integer performance and a tenfold improvement in floating point performance compared to the 25-MHz Motorola 68030, which was introduced only 27 months before the 68040. Similar claims can be made for the Intel 80386 and 80486 architectures, as well as for such reduced instruction set computer (RISC) based processors as the Sun Microsystems Sparc and the Mips Computer R*n*000 family. As companies like Apple and NeXT incorporate these new fast processor chips into complete systems along with features not generally available on personal computer (PC) class machines (special purpose graphics hardware, direct memory access I/O, high speed network links, and high-capacity/fast-access/small-form factor disk drives and so forth), we are on the verge of fulfilling a vision that is now many years old—the economic viability of a personal interactive three-dimensional molecular modeling workstation on every chemist's benchtop.

Application software, however, is equally as important as computer hardware, and the improvements in hardware performance and price have yet to carry over to computer software. The problems that must be solved are numerous and formidable: A uniform and easily learned user interface is key, as are increases in programmer productivity and reductions in cost and time-to-market of new applications. Also important are the integration of molecular modeling applications with desktop publishing packages and electronic mail systems, the development of data interchange standards, and the integration of commonly used computational chemistry programs, both running on local workstations and at remote sites like supercomputer centers. The PC community is beginning to seriously address applications portability through the consideration of standards like the UNIX operating system[3] and the X window system.[4] This is due largely to the impact of UNIX and X in the scientific and engineering marketplace, where these software systems are found on virtually every product.

REVOLUTION IN COMPUTER HARDWARE

It is instructive to review the relatively brief history of high performance microprocessor technology. The constant stream of improvements in microprocessors is due primarily to three cooperating technologies:

(1) Reduction of the wafer surface area required to fabricate individual semiconductor components and interconnections between these components, which results in greater component density;
(2) Improvements in the manufacturing process, which result in greater yields, larger wafer sizes and faster parts;
(3) Improvements in packaging technology, which result in the ability to accommodate a greater number of external connections to the device.

Although similar claims can be made for other brands of processor chips, we have chosen the Motorola 68000 family for our review, if for no other reason than that the information was available to us.[5]

Address reprint requests to Dr. Ferrin at the Computer Graphics Laboratory, Department of Pharmaceutical Chemistry, University of California, San Francisco, CA 94143-0446, USA.
Received 10 October 1990; accepted 15 October 1990

The MC68000 was introduced in September 1979 and contained approximately 68 000 transistors. The processor ran at an 8-MHz clock rate and achieved about 0.66 MIPS performance. Memory management was provided outside the CPU chip, either through separate medium-scale logic components or through a Motorola MC68451 memory management unit (MMU). The processor contained sixteen 32-bit registers (eight registers for addresses and eight for data) and could directly address byte, word, or long-word data types (word and long-word data types had to be aligned on 16-bit boundaries). Although the chip used 32-bit data paths internally, data was accessed off-chip via a 16-bit databus, using a 24-bit (16 megabyte) addressing scheme, to minimize the required number of external connections to the chip. The external interface was achieved using a 64-pin package and there was no provision for floating point arithmetic. The logic technology was high-density metal oxide semiconductor (HMOS), which yielded a speed-power product four times better than standard metal oxide semiconductors (NMOS) and circuit densities twice that of NMOS. Because the 68000 could not restart aborted instructions, it could not be used to support virtual memory architectures. Nevertheless this chip was the basis for many workstations, including the Stanford University Network SUN Workstation[6] and the Sun Microsystems Sun 1/100 (which sold for $13 000 in 1982).

The 68010, which was introduced in December 1982, offered a higher clock rate and a small instruction cache, thereby providing an incremental performance improvement. The chip also contained the necessary logic for restarting aborted instructions and thus was the first processor chip that could support virtual memory. The 68010 was pin-for-pin compatible with the 68000, offering a simple upgrade path. The Sun Microsystems 2/120 workstation, which was introduced in November 1983, used this chip.

The 68020 contained 200 000 transistors and was introduced in June 1985. It contained a 256-byte instruction cache and a three-stage instruction pipeline for improved performance, but still used a separate MMU and MC68881 floating point unit (FPU) coprocessor. Its performance ranged from 1.5 MIPS to 4 MIPS and from 0.04 to .125 MFLOPS. Data operands could be aligned on any byte boundary in memory.

In October 1987 Motorola introduced the 68030 processor. This processor chip consisted of 300 000 transistors and was housed in a 128-pin enclosure. It was implemented using 1.2-micron high-density complementary metal oxide (HCMOS) technology and therefore dissipated only about two watts of power. The MMU was integrated with the CPU on a single chip, although the FPU was still a separate coprocessor. The chip contained separate and simultaneously accessible instruction and data caches of 256 bytes each, as well as multiple internal databuses, which allowed parallel access to instructions and data. (This is known as "Harvard-style" architecture.) The 68030 was available with clock rates ranging from 16.7 to 50 MHz, with concomitant performance ranging from 4 to 12.5 MIPS. Using an MC68882 FPU coprocessor, floating point performance ranged from 0.4 to 1.0 MFLOPS.

The MC68040 was announced in January 1990 and contains an unprecedented 1.2 million transistors implemented with 0.8-micron HCMOS technology. This processor combines CPU, MMU and FPU on a single chip and uses two 4-KB caches for instructions and data. The chip is physically larger than any of the previous 68000 family and has 179 external connections. Both 25-MHz and 40-MHz versions have been announced, and performance is claimed to be 20 MIPS (3.5 MFLOPS) and 30 MIPS (6.0 MFLOPS), respectively. As of September 1990 the chip was available in small quantities only, although NeXT Computer Inc. had estimated that volume delivery of workstations based on the 68040 would begin by the end of 1990. The "NeXTstation" is rated at 15 MIPS (2.0 MFLOPS) and has a list price of $4 995.

Thus we have seen a 45-fold performance improvement in the $10\frac{1}{4}$ year time period between the introduction of the original 68000 and the newest member of the family, the 68040. The improvement in floating point performance has been even more impressive since floating point arithmetic was first simulated in software on the original 68000. It is worth noting that the first virtual-memory based minicomputer, the Digital Equipment VAX-11/780, was introduced in 1978, sold for approximately $400 000 and had a performance rating of approximately 1-MIPS. Personal workstations like the $5 000 NeXTstation have 15 times the performance and are available for 1/80*th* the cost of comparable performance machines of little more than a decade ago.

Unfortunately the revolution in computer hardware during the past decade has not been accompanied by a corresponding revolution in computer software. A decade ago the UNIX operating system was already nine years old and widely used within the computer science departments of major universities. Window systems were beginning to emerge[7–10] and early bit-mapped terminals like the Bell Labs "Jerq"[11] and the BB&N "bitgraph" were proving cost effective to build as the price of the semiconductor memory necessary for the video display dropped rapidly. Programming languages included C and FORTRAN. (The latter is now 34 years old.)

One can certainly argue that there have been incremental improvements in computer software in the last ten years: UNIX is much more mature and its use is nearly universal among today's workstations, bit-mapped terminals with window systems are the norm now, and object-oriented programming languages like Smalltalk, C++ and Objective-C[12] have begun to supplant the original C programming language. However these are evolutionary changes, not revolutionary. Recently significant advances in the ability to rapidly develop new application programs can be attributed to the combination of object-oriented programming languages with graphical user interface design tools like Interface Builder from NeXT, Guide for the X/OpenLook window system from Sun Microsystems, and X Build (Nixdorf Computer) or X Designer (Imperial Software Technology) for the X/Motif window system from Open Software Foundation. For example, NeXT's Interface Builder allows software developers to describe graphically the user interface of a new application before any code is ever written. The user interface can then be associated with semantic actions within the program through use of the Objective-C programming language, which encapsulates data objects and sends them as messages to and from the user interface as well as to other objects within the application. While it is

still too early to predict the full impact of graphically designed user interfaces and object-oriented programming, early experiences suggest that it may be possible to reduce application development time by 50%.[13]

The UCSF Computer Graphics Laboratory has recently been exploring the software development environment provided on the NeXT workstation for use in our molecular modeling application development. This involves the use of the "NeXTstep" user interface and the Interface Builder application for developing user interfaces. The former is noteworthy as it recently received the prestigious Software Publishers' Association 1990 Andrew Fluegelman Award for its "innovative contribution to the personal computer community" and has been adopted by IBM for use on its RS/6000 and PS/2 family of workstations. Our experiences have culminated in the conversion of our MidasPlus molecular modeling package[14] for use on the NeXT workstation.

SOFTWARE DEVELOPMENT ENVIRONMENT

The original NeXT personal computer was a workstation based on the Motorola MC68030 CPU running at 25-MHz. The minimal configuration consisted of the CPU board, 8MB of memory, a 256-MB removable media magneto-optical disk, an 1120 × 832 monochrome monitor with four gray levels, a keyboard, a two-button mouse and an ethernet interface. Our software development systems also have an optional 330-MB hard disk for better I/O throughput. The academic institution price for our original machine was $8 500 in January 1989. Newer systems based on the Motorola 68040 are available for $3 775 to academic institutions.

Standard software for the NeXT machine includes the Mach operating system, many user-level applications, and NeXTstep and Interface Builder. Mach is a multitasking operating system written at Carnegie Mellon University.[15] While the system internals are different, the user utilities are identical to those of the popular 4.3 Berkeley Software Distribution version of UNIX. User-level applications include a relational database server from Sybase, Allegro Common Lisp from Franz Inc., Mathematica from Wolfram Research Inc. and several utilities developed by NeXT, such as Digital Webster and Digital Librarian. NeXT also provides NeXTstep, a software development environment that includes an object oriented software library called the application toolkit (or "appkit"), a graphical user interface construction program called Interface Builder, and a window system that uses Display PostScript from Adobe Systems.[16] Software developers interact most with the first two components when writing new software or porting existing applications.

The appkit is written in Objective-C and is organized into a hierarchy of classes. Objects from these classes are used to manipulate windows, to draw graphics and to implement the NeXT-style user interface. Examples of object classes include "window," "view" and "slider." (See Color Plate 1.) These classes are supplemented by routines that implement commonly used functions, such as displaying an alert panel containing an error message. Both the appkit and custom graphics are programmed to use a C interface to Display PostScript, which is a superset of PostScript, a simple device-independent interpretive programming language with powerful graphics capabilities that is supported on many laser printers. Thus, the image that appears on the screen is easily duplicated on printers. The combination of the appkit with Display PostScript greatly simplifies the creation of graphical applications because it hides low-level complexities and presents an orderly interface to graphical functions.

Even with the appkit, the design of user interface layouts can be difficult. Frequently, creating a user interface layout involves positioning control elements (such as buttons and sliders) within a window relative to each other. With the appkit, as with most existing graphics toolkits, the programmer must textually specify properties, such as the coordinates and color, of these control elements in a program, then compile and execute the program to confirm that the specified properties are desired. To facilitate the creation and modification of layouts, NeXT designed "Interface Builder," a user interface editor that allows the programmer to interactively manipulate graphical entities, such as windows, sliders, and buttons, and thereby to describe quickly and accurately the "look and feel" of the graphical user interface for a particular application, even before the semantics of the individual user actions have been defined. To effectively link the graphical design with the appkit and user objects, Interface Builder supports a target-action paradigm for message passing. The programmer can specify that upon the activation of a control element (e.g., a pushed button), an Objective-C message representing the "action" be sent to a particular "target." The target-action paradigm frees the programmer from worrying about control flow. Objects receive messages when certain user actions occur. As long as objects handle messages properly and are not greatly dependent on an external state, they will behave in the expected manner. The combination of the appkit with Interface Builder greatly accelerates user interface design, which is normally the most time-consuming task in implementing an application.

PORTING MidasPlus TO THE NeXT WORKSTATION

The molecular interactive display and simulation (MIDAS) system[14] is a collection of programs developed in the early 1980s by the Computer Graphics Laboratory at the University of California, San Francisco. The major component of the MIDAS system is an interactive graphics display program, MidasPlus, which was designed for the display and manipulation of macromolecules like proteins and nucleic acids. Several ancillary programs are also part of the system and allow for such features as computing the solvent accessible surface of a molecule, selecting an active site region within a molecule, computing electrostatic charge potentials, displaying space-filling images with shadows cast from multiple light sources, and displaying "ribbon" drawings depicting secondary protein structure. MidasPlus also supports interactive communications with other programs, such as the DOCK protein docking program.[17,18] MidasPlus was written during the summer of 1989 and designed to work exclusively on modern computer graphics worksta-

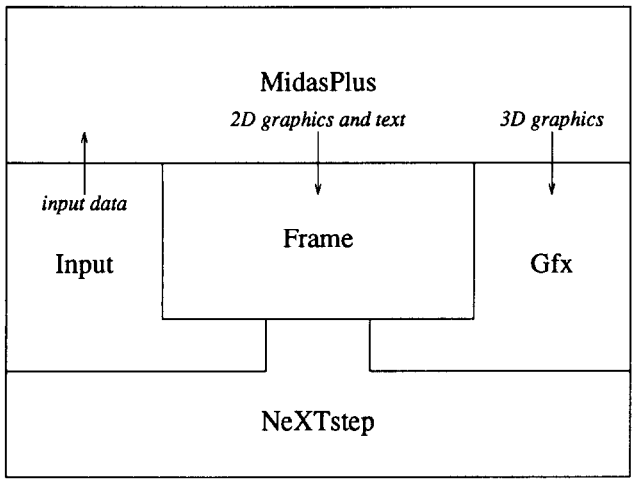

Figure 1. Software interface layering in MidasPlus. Input, frame and gfx modules comprise the device-dependent portion of MidasPlus, with the rest of the source code forming a device-independent module that does not change across workstation platforms

tions; it is a complete rewrite of the original MIDAS display program, which ran on several older generation graphics display systems. The entire MIDAS system represents approximately 50 000 lines of source code written in the C programming language. The MidasPlus display program contains 35 000 lines of C code.

MidasPlus was designed to be easily ported to a variety of three-dimensional graphics standards and interfaces, while keeping the same basic "look and feel" to the user interface. Internally, MidasPlus is divided into several functional units separated by well defined software interfaces. Three key modules—input, frame and gfx—separate the device-dependent and device-independent portions of the program. (See Figure 1.) The first two modules provide the basic building blocks for the user interface (windows, two-dimensional lines, text and device input); these primitives are then combined to implement sliders, buttons, command line input, direct manipulation, and scrolling replies. The gfx module provides three-dimensional graphics routines, such as dots, lines, a transformation matrix stack, display lists and picking. The three modules together form a device-dependent software layer that provides a hardware-independent interface to the rest of MidasPlus. Calls to the input, frame, and gfx modules are translated into procedure calls to the underlying graphics primitives library supplied by the vendor of the particular hardware platform being used. For example, on the Silicon Graphics IRIS, dots are transformed and drawn in three dimensions by the hardware via procedure calls to the "GL" graphics library,[19] while on the NeXT workstation dots are transformed and clipped in software and drawn with Display PostScript. This software design isolates the main application program from the graphics library dependencies of a particular workstation. Consequently, to port MidasPlus to a new hardware platform all that is necessary is to write new input, frame and gfx modules that use the software interface available on that platform. Because these three modules typically represent only 15% of the total lines of source code with MidasPlus, the task of porting to a new hardware platform is simplified and isolated to a specific portion of the overall program code.

On the NeXT workstation, the NeXTstep user interface environment provides nearly all of the functionality of the frame module so that procedure calls in this module result in roughly a 1:1 correspondence to procedure calls in NeXTstep. In addition, NeXTstep provides the building blocks for the input and gfx modules, so that these modules are straightforward to implement. Changes in MidasPlus necessary to take advantage of NeXTstep, therefore, are minor, with the result that the input, frame and gfx modules comprise only 8% of the overall source code in the NeXT implementation of MidasPlus. The port to the NeXT machine was done over an eight month period by a programmer with several other responsibilities. We estimate that the project involved four months of fulltime equivalent effort, including learning the details of NeXTstep. The same version of MidasPlus works equally well on the monochrome and color versions of the NeXT workstation. In fact, the program did not need to be recompiled when we first tested it on a color version of the NeXT. Color Plates 3–5 show sample images taken from the "NeXTdimension" color workstation.

The construction of the user interface was straightforward with NeXTstep's Interface Builder. All of the source code normally required to layout the user interface and to provide buttons, text input and sliders was "programmed" visually rather than through a classical procedurally oriented programming language. As this was our first port of a complex existing application to the NeXT, the most difficult step turned out to be learning how to program the NeXT appkit when the Interface Builder didn't provide a visual interface for implementing a particular form of an interaction. For instance, it took some detective work to discover how to forward keyboard events in the main window to the MidasPlus input command line. Our task was made considerably more difficult by the fact that the chapter entitled "Putting Together a NeXT Application" in the NeXT system reference manual was unavailable at the time. On the other hand, using NeXTstep also gave us extra functionality for free. For example, mouse-based input editing of text worked automatically and all the example commands that are part of the MidasPlus on-line help system can be selected, copied and pasted into the normal MidasPlus command input window. This cut-and-paste support is built into the NeXTstep text drawing interface and hence is available in all applications.

An important technique for interacting with images in MidasPlus is the "virtual trackball," a method for performing three-dimensional (3D) rotations using only a two-degree of freedom mouse for user input.[20] Color Plate 2 shows the circular boundary representing the projection of a sphere on the graphics display screen. When the mouse is inside the circle, dragging the mouse is interpreted as a rotation of both the sphere and associated molecule about the center of the sphere, allowing any orientation to be reached. Dragging the mouse outside of the circle is interpreted as pure rotation about the *Z*-axis. This provides a convenient way to do rotations about any axis and is essentially equivalent to a three-degree of freedom joystick, with the advantage

that a mouse comes as standard equipment on virtually every computer workstation. Because the source code needed to implement the virtual trackball interface is freely available,[21,22] this interaction technique could easily be incorporated into all molecular modeling software.

Because the NeXT workstation currently does not include true 3D hardware capability, it was not possible to include all of the functionality available on other hardware platforms in the NeXT implementation of MidasPlus. For example, the "ribbon" command[23] depends on a hardware Z buffer to render the image, and the NeXT workstation currently lacks this hardware feature. The lack of three-dimensional transformation hardware also drastically affects the performance of MidasPlus on the NeXT workstation, because display functions that would normally be implemented in hardware on a graphics workstation must instead be simulated in software. While interactive performance is adequate for modeling drugs and small proteins like glucagon (246 atoms), larger structures cannot be manipulated in real time. Nevertheless, the ability to display and manipulate even small molecular structures using the same personal computer that potentially is present on every worker's desktop is a significant advance over the shared equipment of just a few years ago.

CONCLUSIONS

Fast and affordable personal computers like the $5 000 NeXTstation ($8 000 for a color version) now make it possible to provide personal interactive molecular modeling systems capable of manipulating drugs and small protein molecules in real time. Furthermore, recent advances in graphical user interface design tools and object-oriented programming environments are making it possible to quickly develop reliable application software on these computers. We have shown that it is realistic to move a large interactive molecular modeling application program originally designed for high-performance 3D graphics workstations onto a personal computer and still provide adequate real-time performance. Key to the ease of getting our application working was the environment provided by the UNIX operating system, Display PostScript, the NeXTstep user interface, and the Interface Builder design tool. The high performance of today's microprocessors allows software to perform functions that previously required special purpose graphics hardware.

ACKNOWLEDGEMENTS

This work was supported by the National Center for Research Resources of the National Institutes of Health under grant RR-01081 and by the Defense Advanced Research Projects Agency under contract N00014-86-K-0757, administered by the Office of Naval Research. We also thank NeXT Computer for giving us access to an early version of their newly announced color workstation to generate the color figures.

REFERENCES

1 Morin, R. Is Sun Playing Fair with Processors? *Sun Expert* 1990, **1** (6) 40–3

2 Oh, Boy! 040. *Semiconductor Data Update* 1990, **10** (1) 2–3

3 Ritchie, D.M. and Thompson, K. The UNIX Time-Sharing System. *Comm. ACM* 1974, **17** (7) 365–375. UNIX is a registered trademark of AT&T Bell Laboratories

4 Scheifler, R.W. and Gettys, J. The X Window System. *ACM Trans. Graphics* 1986, **5** (2) 79–109

5 Mosley, D. (Motorola Applications Engineer) personal communication, 1990

6 Bechtolsheim, A. and Baskett, F. The SUN Workstation. Technical report, Stanford University computer science department, 1981

7 Lantz, K.A. and Rashid, R.F. Virtual Terminal Management in a Multiple Process Environment. in *Proceedings of the 7th Symposium on Operating Systems Principles*. ACM, New York (1979), 86–97

8 Weinreb, D. and Moon, D. *Introduction to Using the Window System*. Symbolics, Inc., Cambridge (1981)

9 Tesler, L. The Smalltalk Environment. *Byte* 1981, **6** (8) 90–147

10 Meyrowitz, N. and Moser, M. in *Proceedings of the 8th Symposium on Operating Systems Principles*. ACM, New York (1981) **15**, 180–9

11 Pike, R. Graphics in Overlapping Bitmap Layers. *Computer Graphics* 1983, **17** (3) 331–56

12 Cox, B.J. *Object Oriented Programming—An Evolutionary Approach*. Addison-Wesley, Reading (1986). Objective-C is a registered trademark of Stepstone Corp.

13 Manzi, J. (President and CEO, Lotus Development Corp.) personal communication, 1990

14 Ferrin, T.E., Huang, C.C., Jarvis, L.E., and Langridge, R. The MIDAS display system. *J. Mol. Graphics* 1988, **6** (1) 13–27. The MIDAS software package is licensed by the Regents of the University of California and can be obtained by writing to our laboratory

15 Accetta, M., Baron, R., Bolosky, W., Golub, D., Rashid, R., Tevanian, A., and Young, M. Mach: A new kernel foundation for UNIX development. in *Proceedings of the Usenix Technical Conference and Exhibition*. Usenix Association, Berkeley (1986), 93–112

16 Adobe Systems Inc. *PostScript Language Reference Manual*. Addison-Wesley, Reading (1985)

17 DesJarlais, R.L., Sheridan, R.P., Seibel, G.L., Dixon, J.S., Kuntz, I.D., and Venkataragahaven, R. Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure. *J. Med. Chem.* 1988, **31** (4) 722–9

18 Lewis, R., Roe, D., Huang, C.C., Ferrin, T.E., and Kuntz, I.D. *J. Mol. Graphics*, submitted

19 *Graphics Library Reference Manual*. Silicon Graphics, Inc., Mountain View, CA, 1988

20 Chen, M., Mountford, S.J., and Sellen, A. A study in interactive 3-D rotation using 2-D control devices. *Computer Graphics* 1988, **22** (4) 121–9

21 Hultquist, J. A Virtual Trackball. in *Graphics Gems*. (A.S. Glassner, Ed.) Academic Press, New York (1990) 462–3

22 The source code and documentation for implementing the virtual trackball technique of display interaction is available from the UCSF Computer Graphics Laboratory

via anonymous file transfer protocol. Hostname is cgl.ucsf.edu, file pub/vsphere.shar
23 *UCSF MidasPlus User's Manual.* Technical report, computer graphics laboratory, University of California, San Francisco (1989)
24 Huang, C.C., Pettersen, E.F., Klein, T.E., Ferrin, T.E. and Langridge, R. Conic—A fast renderer for space-filling molecules with shadows. *J. Mol. Graphics,* submitted
25 Wlodawer, A., Miller, M., Jaskolski, M., Sathyanarayana, B.K., Baldwin, E., Weber, I.T., Selk, L., Clawson, L., Schneider, J. and Kent, S.B. Conserved folding in retroviral proteases: Crystal structure of a synthetic HIV-1 Protease. *Science* 1989, **245** (4918) 616–21
26 Miller, M., Schneider, J., Sathyanarayana, B.K., Toth, M.V., Marshall, G.R., Clawson, L., Selk, L., Kent, S.B. and Wlodawer, A. Structure of complex of synthetic HIV-1 protease with a substrate-based inhibitor at 2.3-Å resolution. *Science* 1989, **246** (4934) 1149–52