The MIDAS display system

Thomas E. Ferrin, Conrad C. Huang, Laurie E. Jarvis and Robert Langridge

Computer Graphics Laboratory, Department of Pharmaceutical Chemistry, School of Pharmacy, University of California, San Francisco, California 94143-0446, USA

The Molecular Interactive Display and Simulation (MIDAS) system is designed to display and manipulate large macromolecules, such as proteins and nucleic acids. Several ancillary programs allow for such features as computing the surface of a molecule, selecting an active site region within a molecule, and computing electrostatic charge potentials. At the core of MIDAS is a hierarchical database system, designed specifically for macromolecules, that is both compact in its storage requirements and fast in its data access.

Keywords: molecular modeling, interactive computer graphics

Received 9 December 1987
Accepted 23 December 1987

HISTORICAL PERSPECTIVE

MIDAS is the most recent in a series of interactive molecular graphics systems whose direct lineage extends back to the first developments in interactive molecular graphics at Project MAC, MIT, in 1964.[1-3] National Institutes of Health (NIH) support began with the formation of the Computer Graphics Laboratory at Princeton University in 1969* and resulted in a number of pioneering developments, including CAAPS (Computer Aided Analysis of Protein Structure).[4]

In 1976 this NIH research resource moved to the University of California at San Francisco (UCSF). A new graphics package[5] was designed at UCSF to operate under the UNIX†[6] operating system, and a new molecular graphics system, Molecular Interactive Display System (MIDS), evolved that was able to accommodate the new developments in color displays.[7] The system was used by numerous visitors to the UCSF Computer Graphics Laboratory.

In 1980 we decided to redesign the system completely, making use of the lessons learned over the previous 15 years. The result of this effort, MIDAS, emphasizes highly interactive display and manipulation, with a data structure designed for very fast access to and manipulation of large and complex molecules, such as protein and nucleic acids.[8-10]

MIDAS was originally developed on the UNIX operating system for use with an Evans and Sutherland (E&S) Picture System (PS) 2 display. It was recently converted for use on other operating systems (VMS and UNIX System V) and other graphics display engines (E&S PS330 and PS350, MPS and Silicon Graphics IRIS).

DESIGN GOALS

MIDAS was designed primarily for modeling proteins and nucleic acids and their interactions with each other and with small molecules, such as drugs. The average amount of data associated with each atom is roughly 20 bytes (atom name, $\{x, y, z\}$ coordinates, some physical properties and graphics information). Thus, the memory requirement for studying an average-sized two-molecule interaction is roughly 50 kilobytes, just for the raw data. Since MIDAS was originally developed on a Digital Equipment (DEC) PDP-11/70 computer that had an inherent limited address space, an important aspect of our design goal was to build a modeling system that could process a large amount of data using little main memory.* Equally important, however, was that this aspect of the design did not affect performance; the system had to provide real-time interactive performance with minimal CPU use, and it could not use excessive amounts of disk space for storing the molecular databases.

MIDAS incorporates into one package the capabilities found through the years to contribute to interpretability of complex objects. These include real-time display interaction through various input devices and a coherent command syntax, stereoscopic image generation, the use of color and representation of molecular surfaces. Hardware capabilities such as depth cuing, perspective, clipping and real-time stereo are incorporated. Emphasis is placed on the interactive selection, manipulation and docking of drugs and receptors.

In short, the program had to be both efficient enough to operate within the space limitations of the available computer system and of sufficiently high performance to provide a truly interactive tool for scientist users.

*The Princeton University Computer Graphics Laboratory was established by a grant in 1970 from the Division of Research Resources of the National Institutes of Health (RR-578); funding also was provided by Princeton University.

†UNIX is a registered trademark of AT&T Bell Laboratories.

*In the late seventies the cost of main memory was about a thousand times greater per byte than the cost of magnetic disks. Most machines were used as centralized timesharing machines whose resources were typically shared among many users. Major technological advances in the past half-decade have influenced computing systems significantly. Microprocessors have become powerful enough to allow users to have their own personal workstations. In the past ten years there has been a fourfold decrease in the cost per byte of disk storage. In this same time period, the cost per byte of main memory has dropped by a factor of 100. These new technological influences would obviously have a major bearing on any design decisions for a new modeling system that was being developed today.

DATA STRUCTURES

In designing MIDAS, we first needed a data structure that could store the large amounts of data associated with a molecular model efficiently and provide fast access for a modeling program. The proteins and nucleic acids for which MIDAS is intended are built from small component molecules that are chained linearly into large structures. The redundancy in the structure of component residues provides an efficiency in nature that is useful in data storage as well. Residue structural data, such as atom names, bonding pattern and linkage atoms, need only be specified once and then applied to all like residues within a model. Only the actual coordinate data (3D position in a Cartesian coordinate system) is explicitly required for each atom.

The MIDAS database structure [see "The MIDAS database system" on page 2] thus consists of three binary files for each molecular model (see Figure 1). A data file contains the coordinate data, a template file contains the residue structural information, and an index file relates the two. The binary format of the database provides for fast access by the MIDAS display program, while the elimination of redundant structural information significantly reduces the MIDAS main memory requirement during program execution.

The utility program *midas.in* builds a MIDAS database (a set of three files) from crystallographic data; the user must provide the residue sequence and the atom coordinate data, as well as the more general residue structural data. The latter is termed a "template" since

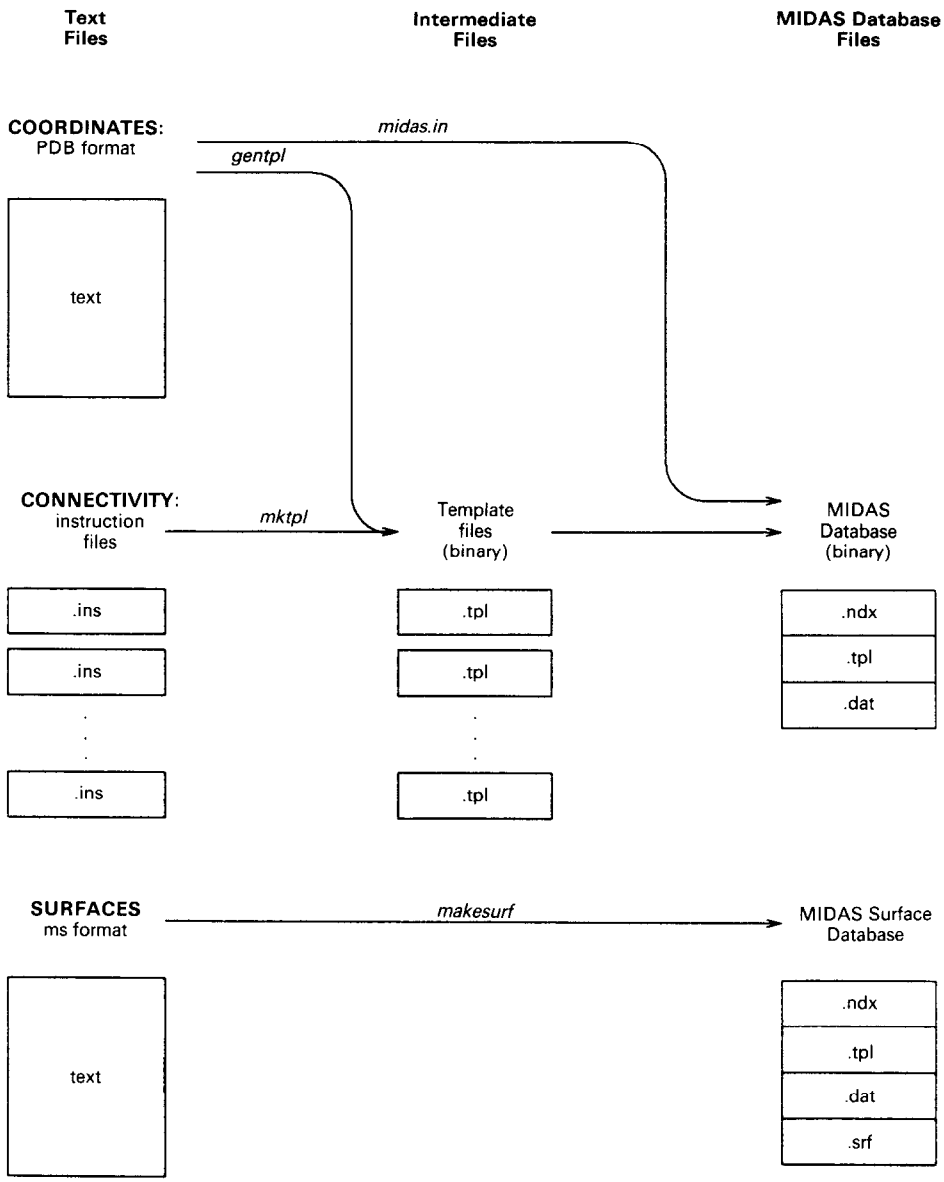

Figure 1. MIDAS database construction. Construction of a MIDAS database using the midas.in *utility program requires two types of input: (1) coordinate data in Protein Data Bank format and (2) connectivity data in the form of a binary template for each residue appearing in the database. These templates are generated from either connectivity instruction files using the* mktpl *program or from Protein Data Bank coordinate files using the* gentpl *utility. A MIDAS solvent-accessible surface database is converted from a ms format file using the* makesurf *utility*

it provides the connectivity information for all atoms in a residue.

Residue templates are generally prepared in advance. Since most models contain residues found commonly in nature, the system includes a library of residue templates. If the user's input data conforms to the established Brookhaven Protein Data Bank conventions,*[11] the corresponding MIDAS database is readily constructed in a few minutes.

If the input data contain residues not found in the library, the *midas.in* program will try to construct an appropriate template. This is done by calculating the distances between atoms of the residue and comparing these distances to a standard set of atom radii to determine the bonding pattern. Alternatively, users can construct templates by hand, using a series of DRAW and MOVE instructions that are used by another utility program, *maketpl*, to generate the template.

To summarize, the templates contain the name and connectivities of each atom in the residue, the residue name, and identification of the atoms that link the residue to the next residue and previous residue in the overall sequence of residues that form the molecule. The Protein Data Bank input data file provides the sequence of residues and the atomic coordinates for each atom in the residues. These files are processed by *midas.in* to create three binary files, which make up a MIDAS database that is accessed by the MIDAS display program.

*See the document "Protein Data Bank Atomic Coordinate Entry Description," distributed by the Brookhaven Protein Data Bank.

MIDAS DISPLAY PROGRAM AND DATA FLOW

The user can configure the MIDAS display program to display an arbitrary number of molecular models (databases) simultaneously, although in practice a dozen simultaneously displayed molecules provide sufficient capacity for studying nearly all problems of current interest while still maintaining a comprehensible display. Even with a dozen simultaneously displayed models, MIDAS provides real-time user interaction with the displayed objects (molecules).

MIDAS is divided into three program modules. One handles the real-time interaction with the user, including polling the input devices and sending instructions back to the display device. A second module modifies and maintains the disk-based MIDAS databases. This includes managing structural data as well as color and temperature factor data. A third module parses commands and manages the graphics objects. Objects are regenerated only when necessary. Thus, a newly generated object returned from the database editor is merged with the unchanged stored objects and then sent as a single display list to the display device. Figure 2 summarizes the data flow among the various modules.

MODEL DISPLAY AND MANIPULATION

The MIDAS display program represents molecules as wire models. Atoms are points that are joined by line segments to represent bonds, and these points can be selectively labeled for identification. The user sees on

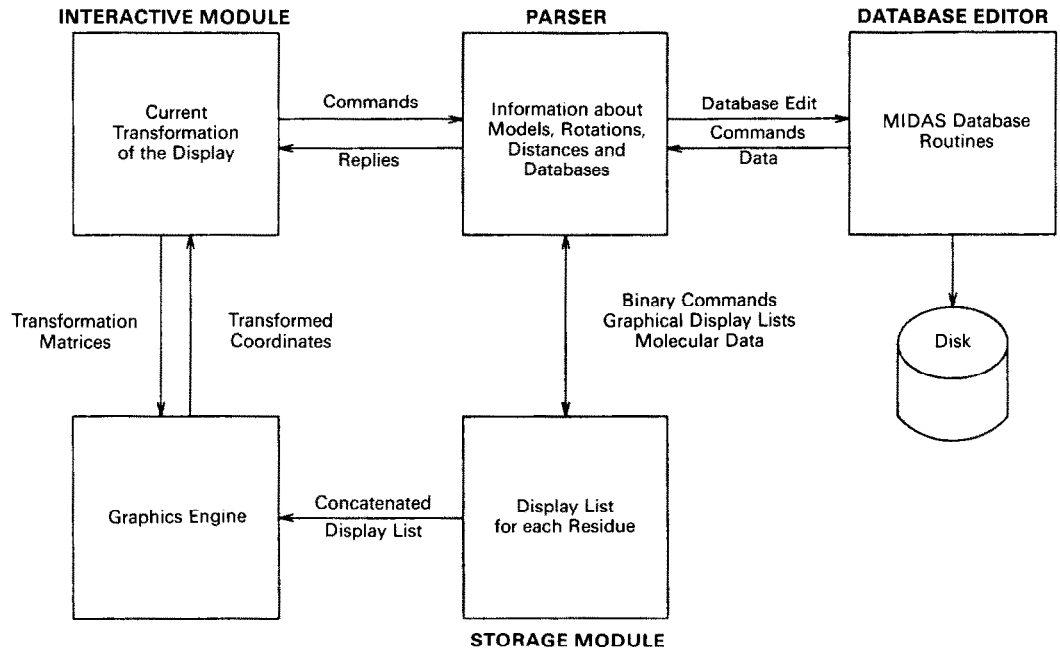

Figure 2. MIDAS display program overview. The interactive module *receives user input from the interactive devices (joysticks, knobs, keyboard, etc.). Interactive commands are processed by the interactive module and transformation matrices sent back to the graphics engine. Editor commands are sent on to the* parser. *If the command requires that the MIDAS database be edited, the command is sent to the* database editor, *which returns new object data for the modified objects. The parser then sends either the new object in the form of a binary display list or a binary command (if the database editor was not needed) to the* storage module. *The storage module sends a concatenated display list of all the objects to the graphics display engine. The parser and storage modules are actually a single process but have been distinguished here for clarity*

the display screen a two-dimensional (2D) representation of a three-dimensional (3D) object enhanced by perspective and depth cuing and, optionally, stereoscopic viewing.

The MIDAS display program provides the user with both global and local interactive object-manipulation capabilities.

The global object-manipulation capabilities include translating one or more objects in the X, Y and Z directions, rotating the objects about the X, Y, Z laboratory axes centered at the common center of mass and scaling the screen image. When more than one model is displayed, any subset of models can be selected for independent translation and rotation about the recalculated center of mass. The center of rotation can be set to any atom. Clipping planes in the x, y and z Cartesian coordinate system provide viewing access to otherwise obscured parts of the molecule.

The user controls all global manipulations in real time through knobs, joysticks, switches, and a 2D data tablet. The nearly immediate response of the display to user input via these devices provides the user with a strong sense of a "real" object. With the addition of stereoscopic viewing via a synchronized display and shutter device, the realism becomes so great that the user may be tempted to pick up the "molecule" and examine it in hand.

MIDAS emphasizes dynamic local object manipulation. The user selectively displays atomic bonds and labels. Isolating the area of interest in large biochemical systems is key to simplifying an otherwise unwieldy amount of graphical data.

Bond rotations about any nonring bond are performed interactively, usually by assigning the rotation to a knob. This bond-rotation capability, coupled with the interactive ability to monitor interatomic distances and angles, provides a powerful tool for quantitatively modeling molecular interactions. Selective display of molecular surfaces and partial molecular surfaces further enhances study of molecular interaction. Both bonds and surfaces can be colored independently, and surfaces are rotated and translated with the associated structure. The specifics of these modeling techniques are discussed in the following section with the associated MIDAS command.

MIDAS COMMAND LANGUAGE

In addition to device input, MIDAS also accepts command text.* The MIDAS command language is based on an LALR grammar[12] and is formally described in Appendix 2.

Key to the MIDAS command language is the syntax used to address molecular objects. The molecular models displayed by MIDAS are referenced by a hierarchical syntax designed to allow users to select only the portions of the model that interest them. The hierarchical scheme is as follows:

- Models: Models are the most general (highest) level of the hierarchy. As currently configured, up to 12 molecular models can be displayed at a time (although there is no fundamental limit to the number), and each model has both an associated name and a number by which it is referenced. A model is usually a single molecule, but it is not so restricted. For example, the four amino acid chains and heme group in hemoglobin could be included in a single model.
- Residues: Within each model, component residue(s) are named and uniquely numbered. Residue numbers can be negative or positive (to conform with biochemical literature) and are usually sequential within a model.
- Atoms: Within each residue, individual atoms are referenced by a unique atom name.

The syntax developed for accessing these components of molecular models uses the symbols "#," ":" and "@" as identifiers for models, residues and atoms, respectively. These identifiers, followed by the associated component name or number, are grouped into expressions that are mapped to selected portions of models. Conventions for the syntax are as follows:

(1) Identifier/components are written in order from model to residue to atom and evaluated so that a given atom identifier refers to the most recent (i.e., previous) residue and model identifier in the string. When an identifier of a higher hierarchical level follows one of lower level, intervening identifiers of lower value are ignored and the most recent higher-level identifier is associated with the named component(s). For example, #0:3@ca:8@na refers to the CA atom of residue 3 on model 0 and the NA atom of residue 8 on model 0.

(2) A residue or model identifier lacking a lower-level specification implies "all." For example #0 implies all atoms of all residues in model 0.

(3) An identifier lacking a higher-level specification implies "all." For example, #0@CA indicates all alpha carbon atoms in model 0 (all residues implied), and @CA alone indicates all alpha carbon atoms in all residues in all displayed models.

(4) Components of the same hierarchical level can be grouped under the same identifier using the "-" character to specify a range and "," to delimit an unordered list. For example, :21-24,35 indicates residues 21 through 24, inclusive, and residue 35. MIDAS does not guarantee the order of evaluation in such cases.

(5) Additional special characters, including wild card matching characters, are listed in Table 1. Expression examples are given in Table 2.

The hierarchical syntax described above provides flexible and explicit access to component parts of molecular objects, allowing the user to be very general or very specific in a minimal number of key strokes. In molecular models the "area of interest" is usually not a contiguous set of residues; in proteins, for example, the active site usually includes contributions from nonsequential residues in the folded amino acid chain. In this respect, the syntax, while explicit, may be quite tedious. Thus, MIDAS provides an aliasing facility for frequently typed strings; a utility program, *irs*, for active site atom selection; and the notion of "sessions," where a work session

*A complete description of the MIDAS Command Language is given in MIDAS User's Manual.[13] Early in the development of MIDAS the advantages of a command language versus a menu-based system were considered, and a command language based on typed commands was chosen. While a typed command language provides greater flexibility and speed, and can be implemented more easily, a menu-based system is currently being developed.

Table 1. MIDAS special characters for atom selection

Symbol	Function	Usage
#	Model number	# *model_number* where *model_number* is an integer
:	Residue	: *residue* where *residue* is a residue name, residue sequence number or range of residues
@	Atom name	@ *atom_name* where *atom_name* is an atom name or range of atoms
-	Range	Specifies a range of atoms such as @CB–* (beta carbon to the last atom), a range of residues such as :35–66 (residues 35 through 66) or a range of colors such as red-blue (shades of red, magenta and blue)
*	Wildcard match	Matches whole atom or residues names. For example, #0:*@CA selects the alpha carbon atoms of all residues in model ∅
?	Single character wildcard	Used for atom and residue *names* only. For example, :G?? selects all three letter residue names beginning with "G"
%	every *nth* residue or atom	For example, :%5 selects every fifth residue in the sequence
b> b<	Temperature factor	b> *temp_ factor* selects all atoms with temperature factors greater than *temp_ factor*. b< *temp_ factor* selects all atoms with temperature factors less than *temp_ factor*. For example, b>20 b<25 selects all atoms with temperature factors greater than 20 and less than 25
e> e<	Electrostatic potential	e> *potential* selects all atoms with electrostatic potentials greater than *potential*. e< *potential* selects all atoms with electrostatic potentials less than *potential*. For example, e>10 e<20 selects all atoms with electrostatic potentials between 10 and 20 kcal/mol

Table 2. Sample specifications for model components

Expression	Meaning
#0	Model 0 (all atoms, all residues implied)
#0:4–10,15	Residues 4 through 10 inclusive and 15 of model 0 (all atoms implied)
@CA	All carbon alpha atoms on all models, all residues (implied)
#1:12–20@CA:14@N	Alpha carbon atoms in residues 12 through 20 and the nitrogen atom on residue 14; all apply to model 1

can be saved so that the selection process need not be repeated when the MIDAS program is later restarted. As an example of the aliasing facility, consider the command *alias AS#0:145:248:270*, which is a typical specification for the active site within a model (in this case, carboxypeptidase). The string "*AS*" can now be used in place of the explicit atom identifier expression in any other MIDAS command.

Realizing that having to type atom specifications is a disadvantage for poor typists, we included in MIDAS a system of "atom picking" using the data tablet and cursor puck to ease the tedium of typing in strings of residue and atom names. Atom picking lets users point to the atom of interest, and MIDAS returns the appropriate character string specification. The user types a MIDAS command, substituting a "+" where the atom specification is to appear. This activates the data tablet and cursor such that a cross hair appearing on the display can be moved to the desired atom. Pressing the button on the cursor puck signals the selection, and MIDAS substitutes the full hierarchical atom name for the first occurring "+" in the typed command. Multiple "+" symbols may appear in the command line; each is sequentially assigned a complete atom specification. The command can be edited after substitution using the normal MIDAS keyboard editing characters (e.g., Control-W to erase a word). As a degenerate case, a "+" by itself without an accompanying MIDAS command still activates the atom picking facility and thus allows a user to temporarily identify a particular residue or atom, but then to ignore this information simply by typing Control-U to erase the line.

MIDAS commands are divided into two categories: *interactive commands*, which change the view of the displayed objects, and *editor commands*, which modify the displayed objects. These two command types correspond to the primary program module in which they are carried out. Editor commands are typically molecule specific (e.g., *addaa* for adding a new amino acid onto the end of a molecule), while the more general interactive com-

mands could just as easily be applied to any computer-generated object shown on the display screen. Editor commands require invoking the MIDAS database editor module to modify the database and are thus slower to execute than interactive commands, which require only that an instruction be sent to the display system. The editor and interactive commands are summarized in Tables 3 and 4, respectively.

By placing a tilde character "~" before the command, the user can reverse the action of most MIDAS commands. This is essentially an "undo" for the commands.

The command language duplicates all the capabilities of the input devices with interactive keyboard commands: *move* translates objects, *turn* rotates objects, *scale* applies a numerical scaling factor, *clip* adjusts the clipping planes, and *select* mimics the switches for activating models. These and other interactive commands essentially change the "view" of the model(s) on the screen. The duplication in the MIDAS command language of the functionality of interactive devices is especially advantageous for constructing command "scripts." Sequences of commands are prepared as part of an ordinary text file using a standard text editor and can then be "read" into the program with the *source* command, just as if the commands had been interactively typed on the graphics display keyboard. This feature is often used during filming sessions and for giving demonstrations to visitors. The command sequence in Appendix 1 and the accompanying color plates illustrate this capability.

A molecular model displayed in its entirety contains more information than the user either needs or wants. It is important to be able to simplify the picture to emphasize the areas of interest or generalize the struc-

Table 3. MIDAS interactive commands

Command	Function
alias	Set command alias
align	Align two atoms along the *z* axis
angle	Calculate the angle between three atoms
assign	Assign knobs and joysticks
cd	Change current working directory
charsz	Change text character size
clear	Equivalent to unset
clip	Move clipping planes
cofr	Change center of rotation
copy	Send image to electrostatic plotter
cpk	Produce a space-filling model on raster terminal
fix	Make bond rotations permanent
freeze	Reset knobs (avoids picture drift)
help	Get information on MIDAS commands
intensity	Set intensity of hither and yon clipping planes
match	Superimpose two models
move	Translate selected models
push/pop	Push and pop images on the picture stack
record	Record all executed MIDAS commands in a file
repeat	Fetch previous command line
reset	Reset all models to original orientations
rock	Rock a structure about the *x*, *y* or *z* axis
roll	Roll a structure or bond rotation about the *x*, *y* or *z* axis
run	Execute a shell command and send output to MIDAS
save	Save a MIDAS session
savepos	Save model orientation
scale	Apply scaling factor to all models
section	Change sectioning of the display
select	Select models for move, rock, roll, turn commands
set/unset	Set options
sleep	Temporarily suspend all activity
source	Read and execute a command file
speed	Set the control speed of knobs and joysticks
stop	Terminate the current MIDAS session
system	Execute a UNIX (VMS) shell command
thickness	Change thickness of the displayed section
vmstat	Collect performance information
transform	Apply a transformation matrix to each selected molecule
translate	Translate molecules in molecular coordinate system
turn	Turn a structure about the *x*, *y* or *z* axis
wait	Interrupt processing until model has stopped moving
window	Display the entire molecule on the screen

Table 4. MIDAS editor commands

Command	Function
addaa	Add amino acid to the end of molecule
addgrp	Add new group to a residue
brotat	"Backwards" bond rotation
chain	Chain specified atoms together
color	Color bonds, labels and surfaces
delete	Delete a group from a residue
display	Display specified molecules, residues, atoms
distance	Display atom distances
getcrd	Return x, y, z coordinates for an atom
label	Label atoms and residues
link	Join two residue chains
menu	Display menu of available molecular structures
open	Open a MIDAS database for display
read	Read a command file containing only editor commands
reverse	Reverse the direction of a rotation
rlabel	Enable residue labeling
(f)rotat	Activate a bond rotation
setcom	Set parameters for picture system objects
show	Display specified atoms deleting all others
surface	Display model surface
swapaa	Exchange an amino acid for another
swapna	Exchange a nucleotide for another
vdw	Display van der Waals surface
vdwopt	Set van der Waals surface options

ture. The *display* command lets users selectively display atoms, residues and models. Thus, a user can simplify a large protein by displaying only the backbone (i.e., eliminating the side chains). Such a simplified picture clarifies the structure, allowing the viewer to easily distinguish gross secondary and tertiary structural features, such as beta pleated sheets and alpha helices. The command *chain* "chains" together only the displayed atoms, ignoring intervening atoms. In this case, the lines connecting the atoms are "virtual" bonds that serve to distinguish gross structural features. For example, the alpha carbon atoms of a protein can be chained (ignoring amino nitrogen and carboxyl carbon) to produce a generalized view of the protein.

Having simplified a model structure, it is useful to carefully select details of the molecular structure for display. The portion of interest is often the active site of a protein. As more detail is added back to the model, color becomes indispensable for enabling the eye to differentiate among the various displayed structures. Protein and substrate can be colored differently, and residues involved in molecular interactions can be highlighted by color. Bonds, labels and surfaces (described in the next section) can be colored independently. This is of particular use for docking two surfaced models where the surfaces are very close to one another and possibly touch. The ability for the eye to distinguish between the two surfaces is crucial. Color also provides an alternative mechanism for atom identification: Bonds can be color-coded according to atom type, for example.

The command sequence in Appendix 1 and accompanying color plates demonstrate the capabilities described above. The sequence begins with the display of a DNA molecule and the cro regulator protein[14] (Color Plate 1). In Color Plate 2, the protein molecule is simplified by chaining together only the alpha carbon atoms of the backbone; the DNA and alpha carbon chain of the protein is docked. Color Plate 3 demonstrates selective display of the arginine 38 residue on the alpha carbon backbone and suggests its potential interaction with a guanine residue of the DNA. In Color Plate 4, the molecules are scaled to a large size, and appropriate clipping planes are applied to examine this interaction. The atoms are colored according to atom type and labeled. Proposed hydrogen bonding is demonstrated with MIDAS distance calculations with quantative bond length information shown in the upper right corner of the plate. Colour Plate 5 demonstrates the use of van der Waals surfaces to examine the proximity of the atoms involved in hydrogen bonding. In Color Plate 6, residue 462 of the DNA molecule has been replaced with an adenine by use of the *swapna* command. Bond rotations are applied to the arginine 38 side chain such that the side chain approaches the adjacent DNA residue, number 463. Distance calculations and van der Waals surfaces are displayed to demonstrate possible hydrogen bonding in this configuration.

MODEL SURFACES

A key feature of MIDAS is the ability to display molecular surfaces. These are represented as dots about the wire model and characterize contours, pockets and internal cavities. Studying molecule interactions depends heavily on accurately characterizing the molecular surfaces in the area of contact.

Two types of molecular surfaces are supported under MIDAS: van der Waals surfaces and solvent-accessible surfaces. Van der Waals surfaces[15] are fast to compute and provide "interactive" surfaces that accommodate

bond rotations readily. MIDAS assigns a van der Waals radius to each atom according to its atom type, creates the surface and clips the areas of overlap to create a Corey-Pauling-Koltun (CPK) type model. Note that the surface of *each* atom can be represented, not just those atoms on the exterior surface of the molecule. These surfaces can be selected for display on an atom-by-atom basis. Surfaces for atoms internal to the exterior molecular surface of the model can be eliminated from the display using the aforementioned *irs* utility program (Interior atom Removal and site Selection). The fifth color plate illustrates the use of van der Waals radii to characterize the surface interactions of two molecules.

Solvent-accessible surfaces are calculated off line using an algorithm first proposed by Lee and Richards,[16,17] which was implemented on a static 2D plotter by Greer and Bush.[18] Implementation at UCSF was first done by Jones in 1978[19] and later by Connolly.[7,20,21] The surface is calculated by rolling a theoretical water "probe" around the van der Waals surface of the molecule and using the contact reentry points to determine the surface. The resulting surface is smooth and free of "seams" between atoms. The program output, an *ms* format* file, is preprocessed prior to display to create a MIDAS database surface file that is accessed by the MIDAS display program. This is a molecular surface and cannot be selectively displayed on an atom-by-atom basis except at the level of the initial calculation. A major distinction between the van der Waals and solvent-accessible surfaces is the "tearing" of the latter when a bond rotation is performed.

Surfaces are useful for visualizing contact points of intermolecular interactions. A utility program, *esp*, which calculates the electrostatic potential of a solvent-accessible surface and stores the information in a MIDAS surface database, allows the MIDAS display program to selectively color surface regions by electrostatic charge. Visualizing the electrostatic pattern at the molecular surface provides additional information for docking.

STRUCTURAL MODIFICATIONS

In general, MIDAS is not designed for making major structural changes to displayed models. For small changes, however, MIDAS has the following capabilities:

(1) Amino acid residues can be added using the *addaa* command and nucleotides added using the *addna* command.
(2) One residue can be substituted for another using the commands *swapaa* (for an amino acid) and *swapna* (for a nucleotide). Color Plate 6 illustrates a swapped nucleotide.
(3) The *addgrp* command enables the user to add a small chemical group from an on-line library of groups to an existing residue.

COORDINATE RECOVERY

Coordinate recovery is particularly important in docking problems so that visually docked models can be subjected to further computational analysis, such as energy minimization calculations. MIDAS provides coordinate recovery via (1) a MIDAS command, *getcrd*, which returns the coordinates of a named atom to the display screen and (2) writing out the entire MIDAS session (i.e., all databases and surfaces) with the MIDAS *save* command and converting binary atom coordinate data to a text file format using the *midas.out* utility program. The latter method requires that bond rotations be "fixed" in the new positions using the MIDAS command *fix*. A saved MIDAS session can be restarted, thus making it convenient for users to break long modeling sessions into several shorter time periods.

CONCLUSION

MIDAS is a powerful molecular modeling system for characterizing the interactions of large biomolecules. The MIDAS program provides real-time display of molecules with manipulation through both input devices and a coherent command language with command syntax designed for hierarchical access to component model parts. The dynamic use of color, clipping planes, depth cueing, perspective and stereoscopic viewing provides utility and flexibility in 3D display. The MIDAS database design provides for fast access to data by the display program and eliminates redundant information. Surface generation using MIDAS utility programs is particularly useful for characterizing the docking of molecules and visualizing the electrostatic potential at the surface.

REFERENCES

1 Langridge, R., and MacEwan, A. W., in *Proceedings, IBM Scientific Computing Symposium on Computer Aided Experimentation.* IBM, Yorktown Heights, NY, 1965, 305
2 Levinthal, C., in *Proceedings, IBM Scientific Computing Symposium on Computer Aided Experimentation.* IBM, Yorktown Heights, NY, 1965, 315
3 Levinthal, C. Molecular model-building by computer. *Sci. Am.*, 1966, **214,** 42–52
4 Langridge, R. Interactive three-dimensional computer graphics in molecular biology. *Computers in Life Science Research,* William Siler and Donald A. B. Lindberg, Eds., 1975, 53–59
5 Ferrin, T. E., and Langridge, R. Interactive computer graphics with the UNIX time-sharing system. *Comp. Graph.*, 1980, **13,** 320–331
6 Ritchie, D. M., and Thompson, K. The UNIX time-sharing system. *Commun. ACM*, 1974, **17,** 7, 365–375
7 Langridge, R., *et al.* Real-time color graphics in studies of molecular interactions. *Science*, 1981, **211,** 661–666
8 Ferrin, T. E., *et al.* Molecular interactive display and simulation: MIDAS. *J. Mol. Graph.*, 1984, **2,** 55
9 Langridge, R., and Ferrin, T. E. The future of molecular graphics. *J. Mol. Graph.*, 1984, **2,** 56
10 Pattabiraman, N., *et al.* Real time energy calculation and minimization in interactive three dimensional computer graphics. *J. Mol. Graph.*, 1984, **2,** 59
11 Bernstein, F. C. The protein data bank: a computer archive. *J. Mol. Biol.*, 1977, **112,** 535–542

*The *ms* program was written at UCSF by Michael Connolly. The program is available from the Quantum Chemistry Program Exchange (QCPE) at Indiana University.

12 Aho, A. V., and Ullman, J. D. *Principles of Compiler Design.* Addison-Wesley Publishing Co., Reading, MA, 1979
13 Jarvis, L., *et al.* UCSF MIDAS user's manual. Computer Graphics Laboratory, University of California, San Francisco, November 1985
14 Anderson, W. F., *et al.* Structure of the CRO repressor from bacterophage lambda and its interaction with DNA. *Nature,* 1981, **290,** 754
15 Bash, P. A., *et al.* Van der Waals surfaces in molecular modeling: implementation with real-time computer graphics. *Science,* 1983, **222,** 1325
16 Lee, B., and Richards, F. M. *J. Mol. Biol.,* 1971, **55,** 379–400
17 Richards, F. M. *Annu. Rev. Biophys. Bioeng.,* 1977, **6,** 151–176
18 Greer, J., *et al.* Macromolecular shape, surface maps by solvent exclusion. *P.N.A.S.,* 1978, **75,** 303
19 Jones, O. J. PIG — protein interactive graphics. Computer Graphics Laboratory, University of California, San Francisco, November 1978
20 Connolly, M. L. Protein surfaces and interiors, Ph.D. Dissertation, University of California, 1981
21 Connolly, M. L. Solvent-accessible surfaces of proteins and nucleic acids. *Science,* 1983, **221,** 709
22 Johnson, S. C. YACC — yet another compiler compiler. Computing Science Technical Report #32, Bell Laboratories, Murray Hill, NJ, 1975
23 Freeman, P., and Wasserman, A. I., Eds. Tutorial on software design techniques. IEEE Computer Society, Long Beach, CA, 1977
24 Wasserman, A. I., and Shewmake, D. T. Rapid prototyping of interactive information systems. *Software Engineering Notes,* 1982, **7,** 5, 171–180

APPENDIX 1: MIDAS COMMAND SEQUENCES USED TO GENERATE COLOR PLATES

The following commands were extracted from a MIDAS command script used for illustrating the interactions of DNA and cro protein regulator. The commands were used for making a film sequence. Lines that begin with the character "#" indicate comments and hence are ignored by the MIDAS command processor.

```
# In this command sequence, CRO protein is model 0 and DNA is model 1.
# Color the two chains of the protein separately and display the DNA model (model #1).
# Refer to Color Plate 1 of the accompanying color plates
set half bond
color red #0:1–66; color green #0:201–266
clip h 1 15 ; clip y −1 15 ; wait
display #1

# Chain together the alpha carbons of the protein backbone.
# The side chains are not displayed.
chain #0@ca

# Select the protein model for movement relative to the DNA molecule.
select #0 ; ~select #1
move x −1 24 ; wait

# Select both models for movement and visually "dock" the models.
# See Color Plate 2 for the result.
select #1 ; move y −0.22 18 ; move x 1 18; wait

# Display protein residue arg 38 (backbone and side chain) and color it yellow.
# See Color Plate 3.
display #0:38
color yellow #0:38

# Position and scale the models for viewing residue 38 and its interactions with DNA.
move y −1 12 ; move x 0.5 6; wait
scale −1 130 ; cl h −0.1 140 ; cl y 0.1 110 ; wait

# Color atoms according to atom type.
# (hydrogen = white, carbon = green, oxygen = red, nitrogen = blue)
color white #0:38@h?? ; wait
color green #0:38@c?,c?? ; wait
color 14 #0:38@n,n?,n?? ; wait
color 32 #0:38@o ; wait

# Label atoms in DNA residue 462.
label #0:38 ; label #1:462 ; charsz 4

# Change center of rotation to an atom approximately in the center of the current view
cofr #0:38@cz ; turn x 1 30 ; wait
```

```
# Color DNA residue 462 according to atom type.
color green #1:462@c?,c?? ; wait
color 32#1:462@o?,o?? ; wait
color yellow #1:462@p ; wait

# Calculate interatomic distances which suggest possible hydrogen bonding.
# Color Plate 4 shows detail of the region of interest.
distance 0 #0 : 38 @ nh2 # 1 : 462 @ o6
distance 1 #0 : 38 @ nh1 # 1 : 462 @ n7

# Display van der Waals surface for specified atoms.
vdw # 1 : 462 @ o6,n7,c6,c8,c5,n1
vdwopt prot ; vdw # 0 : 38 @ cz-*

# Reset center of rotation; save current position as "orig."
cofr #0:38@cz ; savepos orig

# Rotate model for viewing overlap of van der Waals surfaces in hydrogen bonds.
# Color Plate 5 depicts this overlap.
turn y -1 25 ; wait ; turn x 1 45 ; wait

# Reset coordinates to the position previously saved as "orig."
reset orig

# Replace residue 462 in the DNA with an adenine.
# Color the atoms according to atom type and label them.
swapna #1 : 462
color green #1:462,463@c?,c?? ; wait
color 32 #1:462,463@o?,o?? ; wait
color 14 #1:462@n?,n?? ; wait
color yellow #1:462,463@p ; wait
move y -.1 10 ; move x -.2 10 ; wait
label #1:462@n7:463@o4

# Calculate interatomic distances which suggest possible hydrogen bonding.
distance # 0 : 38 @ nh1 # 1 : 462 @ n7
distance # 0 : 38 @ nh2 # 1 : 463 @ o4

# Perform bond rotations on protein residue 38.
rot # 0 : 38 @ ca,cb ; rot # 0 : 38 @ cb,cg ; rot # 0 : 38 @ cg,cd ; rot # 0 : 38 @ cd,ne

# Monitor angles between atoms participating in possible hydrogen bonding.
angle # 0 : 38 @ h12 @ nh2 # 1 : 463 @ o4 ; angle # 0 : 38 @ h13 @ nh1 # 1 : 462 @ n7

# Display van der Waals surface for arginine 38 and DNA residues 462 and 463.
vdwopt prot ; vdw # 0 : 38 @ ca-*
turn 0 .1 77 ; turn 1 -.1 130 ; turn 2 1 110 ; turn 3 1 20 ; wait
~vdwopt prot ; vdw #1 : 463 @ o4,c4,c5,c6 : 462 @ n7,c5,c6,n6,n1,c8
cofr #1 : 462 @ n7

# Apply clipping planes for visibility of van der Waals overlap.
# See Color Plate 6 for the final result.
clip h .33 39 ; turn y 1 45 ; wait
```

APPENDIX 2: MIDAS COMMAND LANGUAGE GRAMMAR

The MIDAS command language forms a nearly context-free grammar* and hence can be described by a Backus-Naur Form (BNF) syntactic specification.[12] This approach has resulted in several advantages; most important was that at an early stage we could generate a prototype of the language interpreter by utilizing the UNIX compiler-compiler tool YACC.[22] This allowed formal analysis of the initial language in order to spot potential ambiguities and conflicts in the parser production rules. It also allowed us to get a "feel" of how

*The language is not strictly context-free because arguments to some MIDAS commands are UNIX filename specifications, which can themselves be arbitrary strings. However, because the MIDAS syntax analyzer is implemented by recursive descent techniques, this minor amount of context sensitivity can be localized to the lexical analysis portion of the language interpreter, and hence the language grammar can be precisely described by formal BNF notation.

the language was to be used without having to implement the complete parser.[23,24]

Because of the desirability to provide comprehensive expression error analysis and informative error messages, however, the final implementation of the MIDAS language parser relies not on YACC but on a recursive-descent (top-down) parsing strategy. This method allows for the detection of syntactic errors in the input command string as soon as it is possible to do so based on the left-to-right scan of the input, normally as soon as the lexical analyzer returns the next input token. Formally speaking, then, the MIDAS command language is an LL(1) grammar, since the parser scans its input left-to-right, constructing a leftmost derivation in reverse, and examining no more than one of the next input symbols on the command line.

The BNF specification of the language follows. Notational conventions are: [] indicates exactly zero or one occurrence, {} indicates zero or more occurrences, and {}+ indicates one or more occurrences. Capitalized words are terminal symbols (tokens), while lowercase words are nonterminal symbols (syntactic variables). The nonterminal symbol "command-list" is the start symbol.

```
command_list    :   {command eoc}+

eoc             :   ';'  /*end-of-command*/
                |   '\n'
                |   EOF
                ;

command         :   ALIAS [WORD REMAINDER]
                |   ASSIGN [assign_args]
                |   ROCK movement
                |   SET [set_args]
                |   UNIT [unit_args]
                |   COPY [copy_args]
                |   CPK [cpk_args]
                |   HELP [WORD]
                |   REDRAW [selectors] atoms
                |   VDWOPT vdwopt_args
                |   COLOR color [',' selectors] atoms
                |   OPEN [open_args]
                |   ADD_GROUP add_group_args atoms
                |   DELETE_GROUP delete_group_args atoms
                |   compute_op atoms
                |   file_op string_list
                |   WAIT [NUMBER[time_unit]]
                |   number_op1 [number_list]
                |   number_op2 [number_list]
                |   word_op word_list
                |   movement
                |   monitor_op NUMBER atoms
                |   single_op
                |   unix_op REMAINDER
                |   db_edit_op atoms
                |   add_residue add_residue_args atoms
                |   swap_residue residue_type atoms
                |   '~' tilde_command
                |   '!' REMAINDER
                ;

tilde_command   :   ALIAS word_list
                |   COFR
                |   ASSIGN physical_device
                |   SELECT number_list
                |   SAVEPOS word_list
                |   SET set_args
                |   SETCOM number_list
                |   OPEN number_list
                |   logical_device
                |   monitor_op number_list
                |   db_edit_op atoms
                ;
```

```
compute_op          :   ALIGN
                    |   COFR
                    |   MATCH
                    |   GETCRD
                    ;

word_list           :   {WORD}+
                    ;

number_list         :   NUMBER {',' NUMBER}
                    ;

string_list         :   {STRING}+
                    ;

assign_args         :   FKEY NUMBER TO SELECT number_list
                    |   physical_device TO logical_devices
                    ;

physical_device     :   device_type device_index device_subindex
                    ;

device_type         :   JOYSTICK
                    |   DIAL
                    |   TABLET
                    ;

device_index        :   NUMBER
                    ;

device_subindex     :   X
                    |   Y
                    |   Z
                    ;

logical_devices     :   {logical_device}+
                    ;

logical_device      :   XROT
                    |   YROT
                    |   ZROT
                    |   BONDROT
                    |   XTRAN
                    |   YTRAN
                    |   ZTRAN
                    |   SCALE
                    |   SECTION
                    |   THICKNESS
                    |   CLIP plane
                    ;

plane               :   LEFT
                    |   RIGHT
                    |   TOP
                    |   BOTTOM
                    |   HITHER
                    |   YON
                    ;

file_op             :   CD
                    |   RECORD
                    |   SAVE
                    |   SOURCE
                    |   READ
                    ;
```

```
number_op1      :   CHARSZ
                |   SELECT
                |   SPEED
                |   INTENSITY
                ;

number_op2      :   FIX
                |   REVERSE
                |   FIXREVERSE
                |   SETCOM
                ;

movement        :   logical_device [delta][duration][wait_period]
                ;

delta           :   NUMBER [spatial_unit]
                ;

duration        :   FOR NUMBER [time_unit]
                |   FOREVER
                ;

wait_period     :   AFTER NUMBER [time_unit]
                ;

word_op         :   SAVEPOS
                |   RESET
                ;

monitor_op      :   ANGLE
                |   DISTANCE
                |   ROTATE
                |   FROTATE
                |   BROTATE
                ;

single_op       :   FREEZE
                |   PUSH
                |   POP
                |   REPEAT
                |   WINDOW
                |   VMSTAT
                |   STOP
                ;

unix_op         :   RUN
                |   SYSTEM
                ;

unit_args       :   TIME [time_unit]
                |   DISTANCE [spatial_unit]
                |   ANGLE [spatial_unit]
                ;

set_args        :   {set_arg}
                ;

set_arg         :   COFG
                |   HALFBOND
                |   INDEPENDENT
                |   LABELS
                |   LIGHTS
                |   REASSIGN
                |   ORTHO
                |   PAIR
```

```
                    |   SINGLE
                    |   STEREO
                    |   TEXT
                    |   VERBOSE
                    ;

time_unit           :   SECOND
                    |   MINUTE
                    |   FRAME
                    ;

spatial_unit        |   DEGREE
                    |   RADIAN
                    |   ANGSTROM
                    |   PS
                    ;

atoms               :   {model residue atom tfrange esrange}
                    ;

model               :   ['#' word_ranges [count]]
                    ;

residue             :   [':' word_ranges [count]]
                    ;

atom                :   ['@' word_ranges [count]]
                    ;

word_ranges         :   WORD ['—' WORD]
                    ;

count               :   '%' NUMBER
                    ;

tfrange             :   {tfspec}
                    ;

tfspec              :   TF '>' NUMBER
                    |   TF '<' NUMBER
                    ;

esrange             :   {esspec}
                    ;

esspec              :   ESP '>' NUMBER
                    |   ESP '<' NUMBER
                    ;

db_edit_op          :   DISPLAY
                    |   CHAIN
                    |   LABEL
                    |   SHOW
                    |   RLABEL
                    |   VDW
                    |   SURFACE
                    |   LINK
                    ;

selectors           :   selector {',' selector}
                    ;

selector            :   BOND
                    |   LABEL
                    |   SURFACE
```

```
                    |   VDW
                    ;

vdwopt__args        :   {vdwopt__arg}
                    ;

vdwopt__arg         :   RADII STRING
                    |   DENSITY NUMBER
                    |   NUC
                    |   PROT
                    ;

color__selectors    :   color [',' selectors]
                    ;

color               :   NUMBER
                    |   RED
                    |   GREEN
                    |   BLUE
                    |   CYAN
                    |   MAGENTA
                    |   YELLOW
                    |   WHITE
                    ;

open__args          :   [selector] NUMBER STRING
                    ;

add__group__args    :   group__type ',' NUMBER ',' NUMBER [',' NUMBER [',' WORD]]
                    ;

group__type         :   WORD
                    ;

add__residue        :   ADDAA
                    |   ADDNA
                    ;

add__residue__args  :   residue__type ',' NUMBER ',' NUMBER [',' NUMBER]
                    ;

swap__residue       :   SWAPAA
                    |   SWAPNA
                    ;

residue__type       :   WORD
                    ;

copy__args          :   {copy__arg}
                    ;

copy__arg           :   BOX
                    |   DATE
                    ;

cpk__args           :   PLOT
                    ;

help__args          :   WORD
                    ;
```