Menu Cascade End
These are used as demarcations by Grasp. When a history file is read in they indicate a menu sequence to be stored in the programs internal history list. Note that these are NOT needed in macro files as one does not want the history list updated by menu cascades produced by a macro.

The entry in the history list for a menu cascade is made up of the selections chosen from each menu, up to a maximum of sixty characters, and preceded by the string "MenuCascade|". So, for example, if the user had performed the alteration in atom display mode describe above the history entry (enter "history" as a textport command) would be:
n) Menu Cascade|Display>Alter>Atom Type>Flat Circles
where "n" is the index of the entry within the list.

Since all this information is stored from within the program it is as straightforward to rerun the menu cascade as it is to "rerun" a command line entry, i.e. typing "!n" will rerun the above menu cascade.

The actual process by which this accomplished is to make a "temporary" macro, i.e. one which deletes itself when it is completed. As such the macro can be forced into "macro menu" form (see above) like any other macro, i.e. if the user types "!n" to the textport, and has the left or right control button depressed when "return" is hit, then the user will be given a menu:
Macro_Menu: History_Command #n
menu: Grasp-Display
menu: Display Action-Alter
menu: STRUCTURE-Atoms
menu: Color Atoms by-Atom Type
menu: Atom Drawing Mode- Flat Circles
Complete Macro
Selecting any particular entry works as described above for macro_menus, i.e. if the user selects "menu: Atom Drawing Mode- Flat Circles" they will have the chance to alter this menu selection.

This procedure of recalling menu cascades as menus and selecting a starting point can be particularly useful when the cascade is long and the variable one really wants to alter is in the last menu in the sequence.

Because the user may most often want to "replay" the previous menu cascade there is a special button for this, namely the function button F1. Hitting F1 automatically produces the previous menu cascade as a macro_menu.

Note a current limitation of history list replays is that when a history command is issued the history list is NOT updated. Therefore if one were to do a menu cascade to change some value, and then recall it as a macro_menu, the new value entered is not retained in the listing. Also any commands which occur inside of a "do loop" construction are not added to the history file. Finally, the session record file should be able to reconstruct all macro_menus. However this is an additional layer of complication on the replay and might not be as well debugged as it should be!

"Display: Alter: Atoms: Atom Type: Flat Circles" which alters the display mode of atoms. The user might wish the last menu selection "Flat Circles" be altered to "Full Spheres" instead. To do so the user simply makes sure that when the macro is launched either control button is being held down. If this is the case each step in the macro is transformed into a menu entry in a menu entitled "Macro_Menu: Macro_name", where "Macro_name" is the name of the macro being executed. This menu is then presented to the user. If the user clicks away the macro is canceled. If the user selects the last menu entry, which is always "Complete Macro" then the macro is executed to completion. If however the user selects a step within the macro, for instance in the example above the next to last entry which would read: "Atom Drawing Mode: Flat Circles", then the macro will execute as usual until this step when it will instead give the "Atom Drawing Mode" menu to the user, i.e. it will be come "foreground" rather than "background". Note that only menu selections and command line entries are included in a "macro_menu". Note also that this method is also useful in seeing what actions constitute a particular macro.

(This "jump" to the foreground can also be captured in a macro via the command:
foreground
i.e. after this command is processed in a macro all actions are expected to be from the user rather than from the stored macro commands.)

There is one final macro-specific command which can be inserted into a macro file. It is:
input
or
input: file=filename
The first can be used when either a menu selection or a command line entry is expected and causes the program to accept user input instead (but just for this one entry, not all entries further along in the macro as with the "foreground" command). The second, when in the place of a text entry (i.e. either a request by a menu selection or a command line entry) will instead read it from the indicated file. If the file is not already opened it is opened and the first line read. If already open the next line is read. If the last line in the file is reached the file is closed, reopened and the first line read. This functionality can be quite powerful when used with the loop construction detailed above, i.e. at each pass through a loop a different entry can be read in from an appropriate prepared file.

Menu Cascades and the History List

Grasp v1.0 maintained a list of all commands entered via the textport in a history list similar to that provided by UNIX (see Grasp Addendum #1.3: "History and '!' Commands"). As Grasp now has to write all menu sequences to file it was straightforward to include such sequences, or cascades, into the history list. If the user looks at a history file (session record) from v1.1 they will see that each such menu sequence is preceded by the line:
Menu Cascade Begin
and followed by the line:

-n : do not execute any graphics commands
to turn Grasp into a non-graphical program. This also implies that the user does not have to be at a graphics terminal to run Grasp if what is needed is just a numerical procedure. For instance Grasp contains an efficient accessible area algorithm. Suppose a macro is constructed to calculate the accessible area of all residues and output this information on a residue by residue basis. Let this macro be put in a file "area.mac", then the command:
grasp -n -rx area.mac 6lyz.pdb
will read the macro file and execute it on the file 6lyz.pdb. Note that such a macro is contained in the example file "accessible_areas.mac" where it is called "Calculate_and_Export_Residue_Area". (Note that a reason that the symbol "_" is usually to be preferred to simply a space is that then it can be used with the "-r" option described above).

On-the-fly Macro Creation

Sometimes the user might not wish to have to edit a history file to produce a macro, but would rather produce it from within the program. Grasp has this facility. All the user has to do is to hit Control I and the program will begin to record all menu selections, all command line entries and all control keys hit until the user hits Control I again. At this point the program will suggest a "name" for this macro, based upon how many macros are present, for instance "Macro11" if ten macros are already resident, or asks the user to provide one of their own devising, or gives the user the chance to trash the macro just created. This macro name is then added to the list produced by the "Macros" entry in the main menu. The user can save the macro to file via the menu sequence "Write: Macro to File".
IMPORTANT NOTE!
On-the-fly Macro creation is one of the very few operations NOT properly interpreted by history files, i.e. history files will not run properly if a Control I is encountered. (Work-around: copy all actions between Control I's to a macro file to be read at startup and then delete these lines and the Control I's.

All the embellishments (e.g. "pause", "wait", "write" etc.) mentioned above for history files can be inserted into macro files. For instance the user can insert a system command to remove a certain file before it is recreated later in a macro. The user can also run a macro in "Teach Mode", i.e. have the program prompt the user for the correct input during macro execution. Macros can also be embedded within each other i.e. Macro1 can call Macro2 (but don't let Macro2 then call Macro1!). If macros are "nested" be sure that they are present in the macro list to begin with. Macros can also be escaped from by hitting <Esc>.

Finally it is possible to "enter" a macro rather than simply executing it. By this is meant that if the macro consists, for instance, of a series of menu selections normally these are executed sequentially to completion. Suppose, however, that the user would like to change the macro during execution, i.e. by having the process becoming "foreground" at some point, rather than processed in the "background". For instance a simple example might be the menu sequence

Macro Name: abc
Macro End
i.e. the macro must start with the first line and end with the second.
Here "abc" is any name the user would like to associate with the macro. Upon reading in a file thus created ("Read: Grasp Macro File"), the name will appear in the macro listing menu produced when the user selects "Macros" from the main menu. Selecting the macros name from this list will then cause it to execute.

If the user includes the line:
Define: Alt X
after the line with the macro name then the macro will also be associated with the key combination "Alt" "X". X can of course be any letter. This then provides a "hot key" way to initiate a macro. A record of which macros are associated with which Alt keys can be found via the menu selection: "Help: User Defined Alt Keys". This will give an alphabetic listing and also acts as an alternative method of initiating a macro, i.e. if the user selects from the list that macro is executed.

In the current Grasp data directory is a file "default_macros" which contains several macros a user might find useful. Two of them are assigned Alt keys, i.e. Alt B to build a surface, and Alt S to build a surface, read in "full.crg", do a Poisson Boltzmann calculation and interpolate the potentials onto the surface. The automatic installation of these macros, or any others, via the init_grasp file is described below.

Default Macros and Initialization with Macros:

If the user has a set of macros they would like available each time Grasp is run this can be accomplished by a line in their "init_grasp" file:
macrofile= filename
For instance the init_grasp file provided in the current grasp data directory has
macrofile= default_macros
The program will look in the users local directory and the Grasp data directory for any such file to be read at startup.

It is also possible to have the program execute a macro it has already read at startup. The init_grasp command:
execute= macro_name
will execute the macro "macro_name" before relinquishing control to the user. This then can be used as a way to set parameters at start up, for instance to apply a certain color scheme, turn of the cross hairs etc. Although many parameters can be set via the init_grasp file directly this provides almost complete flexibility in setting up Grasp.

Reading, Executing, and Reading and Executing Macros via Grasp command line options, and the "no graphics" option.

There are three new command line options which relate to macros. They are
-r macro_file : read the file "macro_file".
-x macro_name : execute the macro "macro_name" at startup.
-rx macro_file : read the file "macro_file" and execute all macros
therein at startup.
These options can be combined with the option

display for each loop pass. Note also that the nographics/graphics commands are useful here in controlling the "refresh" of the display. An example of such a history file (animation_ex.his) is included in the "example_scripts_and_macros" directory users can now download with the program.

Teach Mode

As well as "demonstrating" Grasp the replay facility can also "teach". By hitting Control T the program is put into "teach mode". While this mode is active any history file or any internal macro (see below) will be replayed a step at a time such that the appropriate action (e.g. menu click, command line entry) is both indicated to the user and requested of the user. Hence a Grasp "novice" can see just what menu and command line procedures are necessary to achieve certain goals. (Note that not all actions within a replay are required of the user, for instance rotations and translations are not.)

History File Limitations and Macros

Although history file replays are useful they also have drawbacks, some conceptual and some by design. Of the latter is a certain "fragility". By this meant that as such files were constructed with the idea of simply reconstructing a session they "expect" a certain starting state, e.g. perhaps no structure loaded. In fact, if a "problem" occurs during the replay of a history file Grasp has few mechanisms with which to recover. An example of a problem which can arise is if a new external file was created during the original session. When Grasp writes a file it always checks to see if the file is present, and if it is it will ask the user if it should overwrite this file. Hence if the original session created a file for the first time there would be no prompt, whereas Grasp will want to produce one during the replay since the file now exists, and since it "expects" all input to be in the history file it will get "out of sequence" with the replay. One simple way around this is just to delete all such files before rerunning a session. Hence sometimes the user may have to put some thought into recreating a session.

An example of a conceptual problem with history files are that they tend to be "too" complete, i.e. may often include too much detail. One might instead want to "extract" parts of a history file into self-contained, "modular", replay units, or "macros". Since these smaller units might reasonably be expected to be run in different situations there is more provision for exception handling. For instance if the program expects a certain menu but the macro provides the selection for a different menu the program can "look-ahead" for the correct selection, or can query the user as to the correct action. Hence "macros" are less "fragile" than history replays. (Hopefully as the program improves the so will the "tensile strength" of both).

Hence, one way to produce a macro is to edit a history file, cutting out the pieces that the user thinks would be useful to save. (Note: The user should, however, ideally remove lines which start "Menu Cascade Begin" or "Menu Cascade End". See below for a description of the function of these lines). To produce a macro file from these pieces all that is needed then is to "wrap" this "code" with the following two lines:

wait: 5.0
Grasp will idle for 5.0 seconds. This can be useful in letting a user read text printed to the textport by...

write: now calculate curvature
Grasp will write "Comment: now calculate curvature" to the textport.

nographics
Continue to process but do not actually draw anything until there is a line which says..
graphics
These two commands allow one to get to the end of a history file very quickly as the predominant time taken in most sessions is graphical, i.e. that necessary to draw to the screen. The user simply places a "nographics" before the first processable line (but not as the first line of the file as Grasp needs to read the header information) and a "graphics" after the last and the script will execute without the screen view changing.

system: rm test.dat
Grasp writes "system command rm test.dat" to the textport and executes the UNIX command "rm test.dat". Having access to system commands is quite useful when dealing with external data files.

pause
Insert a replay "escape" into the replay sequence, i.e. a "pause" menu is produced as described above which allows the user to continue, continue after the next <Esc>, or quit the replay.

do i=1,10
Go through the lines below until reaching a line which reads
end do
at which point return to the first line after the "do i=1,10" line is reached. Repeat ten times! Furthermore if the construction $(i) is encountered anywhere within the "loop" it is substituted with the loop count, i.e. the string "1", "2", "3" etc. up to "10". For instance one can read or write a different sequentially named file each time through the loop.
Note that loops can be nested as long as the loop counters are different, i.e.
do j=1,10
do i=1,10
command #1
.
command #n
end do
end do
will execute the commands one to n 100 times.

Animation is thus possible in a limited respect by having the program alter the

to the "uncovering" of several "hard-to-reproduce" bugs within the program.

To run a session record simply use the menu sequence "Read: History/Script (+execute): Enter Script File Name". Note that the last menu here will automatically include any files it can find with extensions ".gs" or ".his" (for "history") in your present directory, and any ".gs" files in your Grasp data directory. It will also look for a file "session_record" and include it in the list if found.

Note ".gs" files are equivalent to ".s" files previously. The extension was changed so as not to be confused with UNIX ".s" files, i.e. UNIX scripts, and consists of a series of line commands (only) to be executed. For instance the default data directory contains a file "color_only_backbone.gs" which uncolors all but bonds along a peptide backbone.

Finally the user should be aware that history files do not currently record any command line options used when grasp is invoked, e.g. "grasp 6lyz.pdb" will start Grasp and read in the file 6lyz.pdb but there will be no record of this "read" in the history file.

Replay Escape

Whenever Grasp is involved in a replay it is possible to "escape" by pressing the escape ("Esc") key. On doing so Grasp produces a menu entitled "Pause From File Replay", with three options:

Continue Replay
Continue on Next <ESC>
Quit Replay

The first and third options are obvious. The second allows the user to take "time-out" from the replay. Control returns to the user who can then restart the history file at anytime simply by hitting the escape button. For instance the user could switch to fullscreen mode, change an atom display etc. The user should however refrain from making any macros (see later) or alter any characteristic whose value the replay may later depend upon. (Note also that at present any actions taken during the time out are not recorded in the session record of the replay.)

This feature may also be used to halt "runaway" replays, i.e. where an error has occurred and the replay has gone haywire! (Keep hitting "Esc", it will eventually take hold!).

Demonstration Files

Another use of history files is to provide "demonstration" files for others. For instance the user might want to show certain aspects to a co-worker. Since sessions are rarely perfect, i.e. the record has all the users mistakes as well as their successes, editing becomes very useful. In addition, Grasp recognizes several constructions which are not part of the normal recording process. These are listed, along with examples, below.

! this is a comment line

All lines beginning with an exclamation mark are ignored.

Playback Facilities

This section details the ability to "automate" Grasp operations via external files or internal data structures. As detailed below the facilities grew out of functionality to "restore" a Grasp session, and which led to the following additional features:

i) Crash recovery (and debugging).
ii) Demonstration files (including animation).
iii) "Teach" files.
iv) "Macro" files (encapsulation of commonly performed operations).
v) Automatic execution of macros upon start up (e.g. complete Grasp customization).
vi) A "No graphics" Grasp.
vii) On-the-fly macro creation within the program.
viii) Replay of menu "cascades" from the Grasp "history" list (plus, being able to start at any level within the cascade).

Grasp Replay

Many users had asked if it was possible to "save" during Grasp, i.e. to store the position of a session for easy access at a later date. I considered ways of doing this and eventually decided that an outright "save" might not be such a good idea. For one thing Grasp uses large data spaces and hence might require a lot of disk space for a comprehensive save. As a alternative, it occurred to me that a "replay" facility would be almost as useful, i.e. allow a user to resume at a particular point, and, since only the users input need be stored, the "save" file would not be large. On the downside is that the time it would take the program has actually to retrace all its steps might be considerable for a very long session. (Not, though, that this can be alleviated dramatically with the graphics/nographics commands described below).

And so Grasp v1.1 writes out all menu clicks, all command line entries, all control keys, all rotations and translations, all mouse "picks" that the user executes to file. The file is usually named "session_record". Since most session records are not wanted the program will DELETE this file when you quit Grasp, unless a flag has been set in the init_grasp file (see later) or by the user. The latter is effected by hitting control D, which gives the user the option of saving or deleting the session_record file upon program termination. (Note there is also an option to have the program generate a "unique" name for each session record based upon the current date. See notes on the init_grasp file for details).

If the program should crash then there will automatically be a session record "saved" since the program will not have had "time" to delete it! This then is one of the "side" benefits of the approach taken. The user can "avoid" the crash a second time, assuming they have some idea of what caused it, by editing the file and executing it from within the program as a script (see below). (Alternatively they can send it to me with an angry note attached!) This facility has already led

is to allow the user to move the molecule simply by moving the mouse, i.e. without any buttons depressed. To escape this novel control of orientation or position just reclick with either the left or middle mouse button.

Deleting: Molecules, Surfaces, Contours and Objects.

Grasp now allows the user a menu driven mechanism to delete certain data sets. The menu option resides in the main Grasp menu after "Build". For atoms one has the choice of all atoms or a single molecule. Note that one can NOT delete arbitrary subsets of atoms at present. For surfaces the situation is similar, i.e. one can either delete all surfaces or any single constructed surface, i.e. a set of vertices which was created or read in together.

The contour delete will prompt the user for which type of contour, if there is more than one type present, and then display all current contour values of the chosen type. Upon removing one contour, the user is prompted with the same menu but with the value just chosen removed. Hence the user can successively delete several, even all, contours. Simply click away from the "value" menu to refrain from further deletions.

Finally one can delete all current worms by selecting "Objects" from the delete submenu. The delete options will be expanded in future versions.

position. The second is a set of vertices, similarly producing a center as the average position. The third is to enter a box Z coordinate (ONE coordinate). The fourth is to enter a coordinate in angstroms in the molecules frame of reference. The novel feature of these options are that options one, two and four are all updated as the molecule is moved! Thus if a certain residue is entered to define the center then that center of that residue will always be within the slice plane no matter where the molecule is moved. The third option, wherein a single box coordinate is given, has a center fixed at this position which does not change as the molecule is moved around. Finally, after choosing the center, the width is required of the user in box units.

The other options allow the user to alter the center without altering the width of the Z-slice planes and vice versa, plus the option to turn clipping off. If one has the Z-slice tool active then the values within the tool are updated if altered by one of the other methods, i.e. the two should coexist peacefully.

Rotating without even moving the mouse

Moving the molecule with the mouse has never been very straightforward with Grasp, especially when the display refresh rate is slow, which is often true with surfaces. An attempt to give the user a finer degree of control has now been implemented. If the user holds down a mouse button or buttons as if to move the molecule, but instead keeps the mouse motionless for more than half a second, the molecule will rotate or translate as if the user where moving the mouse away from the center of the window. Furthermore, the rate of movement is (quadratically) proportional to the distance of the cursor from the center of the window, i.e. rotations or translations are very slow when the mouse is close to the center of the window, much faster further away. If the user moves the mouse after this automatic motion has commenced, then the motion will stop and the molecule will respond as usual to a moving mouse. (Note that Grasp will sense ANY movement of the mouse and so the user has to hold it quite still to get motion). Picking at an atom or surface is unaffected as long as the mouse button is released within 0.5 seconds, which is usually the case from personal experience.

Continual Rotations (whirling)

The facility just described, i.e. moving the molecule by positioning and holding down mouse buttons, can be extended to a new feature, "whirling", i.e. spinning about an axis without any action by the user. To affect this in grasp simply have the molecule rotating about some axis by the method described above, hold down the control key (either side) and release the mouse button being used to rotate the molecule. The molecule will continue to move and once it has the user can release the control key. Note that this can also be used for translations! To quit whirling simply click once with the left or middle buttons.

Note that if the user tries the same control key trick while moving the molecule in the traditional "drag" technique, i.e. releases the mouse button while either control key is depressed, then the program does not realize that mouse buttons have been released. This is, of course, the way "whirling" works. Its effect here

"focused" the reflection is. This value may vary from 1.0 to 128.0, (which is actually an "exponent" value), and the system default is 10.0. 1.0 is very diffuse and 128.0 is very localized. Specularity is how "bright" the highlights are. There are three components to specularity, one for how strong the red component is, one for the blue and one for the green. These vary from one to zero and are normally set to 0.5 each. Making the components unequal tends to give the surface a "tint" of the strongest color. This can be quite pleasing if it offsets the surfaces discrete color. Note that these effects can only be seen on surfaces which are both lit and displaying discrete colors. Examples would be a molecular surface in "Lit and Rendered" and "Discrete Color" mode, atoms in "Atom Type" and "Full Sphere" mode, or a backbone worm in "Discrete Color" mode.

Changing the center of world rotations (Control J)

It is possible (and very useful) to change the center about which the world rotations are performed. This is achieved via Control J which gives the user the option of centering about a set of atoms, a set of vertices, a position in box coordinates or some absolute coordinates in angstroms relative to the molecule.

Repositioning to the center of the box (Control G)

In addition to centering the axis of rotation at some new position it is often useful to position these axis so that their center is in the center of the screen. Control G will affect this via an appropriate XY translation, while leaving the Z, or depth, coordinate untouched. This is especially useful if one "loses" the molecule, for instance if one deletes all atoms and reads in another molecule with a substantially different center such that the new molecule is not within view.

Rescaling (Control K)

A third utility, absent for a long time from Grasp, and which compliments changing the center (Control J) and screen repositioning (Control G), is that of changing the scale of the molecule. Grasp implements this by changing the size of the Grasp box, so molecules "appear" larger or smaller. Hence Control K will prompt the user for a new box size, a larger box meaning a smaller molecule and vice versa.

The Slice plane: relative and absolute positioning

Version v1.0 used a simple window tool to control slice planes in Grasp. This is still present, but other options are now available via "Miscellaneous: Z-Slice Options". These options are:

Z-Slice Tool
Enter Center and Width
Enter Center
Enter Width
End Clipping

If the user chooses "Enter Center and Width" they are presented with different options by which to define the midpoint, i.e. center, between slice planes. The first is to enter a set of atoms, in which case the center is the average atomic

types which Grasp will bond has been externalized and now resides in the Grasp data directory where it is named "bonding_distances". This file then allows the user to add or modify "bondables". It also allows the definition of a minimum and a maximum number of bonds for an atom type. If these limits are exceeded, upper or lower, then a report is made to the file "bonding.dat" (Grasp always tells if this file has been created when a molecule has been read). Users who are interested in this facility should look at the file (wherein explanatory notes lie). An example of how one can substitute this file for one in which only bonds between carbon alphas closer than 4.2 Angstroms is also provided, again for interested users. Note that if this file is not found Grasp will use an internal default version similar to that in v1.0.

Furthermore, on the subject of bonding, Grasp will also read CONECT records from pdb files. If such records specify a bond which already exists then it is ignored. (CONECT records list at least two atom numbers per line. Every atom after the first is bonded to the first.) Note that this is ONE (and only) time in Grasp where Grasp uses the atom numbers as provided in the pdb file rather than assigning atom numbers based upon the order in which atoms are read in.

The Parse Charge and Radii Set

The question of what radii and charges to use in electrostatic calculations has been a hard question to answer with assurance. The quantity one usually wants to extract from solving the Poisson-Boltzmann equation is the reaction field energy (the electrostatic "solvation" energy). This can be quite dependent on the choice of charges and especially radii. And yet most charge sets available have not been parameterized with this property in mind. Doree Sitkoff has developed the Parse ("Parse" stands for "PARameterized by Solvation Energy") charge set by parameterizing to solvation energies (to be published) . Although the calculations done with the Poisson-Boltzmann equation in Grasp are of a qualitative nature, the Parse charge set has none-the-less been included as a standard data set for Grasp v1.1.

Changing material properties of surface representations.

The response of rendered surfaces to incident light is described by the "material properties" of the surface, for example the surface "shininess". Grasp uses two "types" of surfaces, one for continuously colored surface , i.e. when the surface is color-coding some property, and one where the surface has a discrete color. The difference in the surfaces is that the latter includes surface reflectivity, where as the former does not. The distinction is made because reflectivity can wash out color distinctions and so is not recommended when trying to purvey information via continuous color coding.

The material properties of discretely colored surfaces can be changed via Control H. This produces a menu of which only the first two options are usefully utilized, i.e. "Shininess" and "Specularity", while "Emmisivity" just adds a certain surface brightness. "Ambiance" and "Diffusivity" and "Alpha" do not as yet do anything.

Shininess is the degree to which reflected light forms "highlights", i.e. how

The final option will output the surface points such that each is grouped with others of the same connectivity index (equal to the residue number).

Note that all positions are in the molecule's frame of reference.

Smoothing Properties : Surface, Worm and Interpolation Plane

Sometimes it is useful to locally average properties to smooth out high frequency effects. Examples are properties on the molecular surface, the backbone worm and the potentials calculated for the interpolation plane. The smoothing for each is calculated by averaging each value with the average of its local neighbors. For the interpolation plane these are points one higher and one lower, and to the left and to the right of each point. For surface points these are the vertices connected to each vertex by a triangle edge. (Note that in the latter case the number may vary from 4 to as many as 12!). For the backbone worm the average is of the values of the segments immediately before and after each segment. Note that for the interpolation plane and worm there will be points which do not have the normal complement of neighbors, i.e. points at the boundary of the interpolation plane and at the termini of worms. These do not have their values smoothed.

Property smoothing is accessed via "Miscellaneous: Smooth Property". If the user then chooses "Interpolation Plane" they are prompted for the smoothing level. This is just the number of times the smoothing operation is applied each time new potentials are calculated. If "Surface Property" is chosen then the user must choose which surface property to smooth. They also get an opportunity to save the original property distribution as another property (for comparison, or later reversal of smoothing). Worm property smoothing options are similar to surface properties.

Smoothing is particularly effective with local surface curvature. This can be seen visually, and also from the regularity of surface contours calculated from smoothed as opposed to unsmoothed curvature. It its also generally useful for worm properties since these are often mapped from the underlying atoms and therefore several successive segments will usually have same value.

Resetting World Rotations

Grasp provides a reset of all world rotations and translations via: "Miscellaneous: Reset World Rotations". If in stereo mode it will reset both sets of world rotations and translations (left and right). By "reset" here is meant "remove" in the sense that the original view is restored.

Bond Distance Tables, CONECT records.

Grasp normally uses atom-atom distances and a set of minimum distances for different possible atom type pairs to decide what is bonded to what. While this approach lacks a lot of chemical information it is surprising good. In well resolved structures there are seldom any incorrect bonding patterns. However Grasp has not encountered all structures and hence its lack of knowledge can sometimes be a drawback. To compensate for this the "matrix" of atom-atom

check for compatibility before changing screen mode and let the user know if they are unlucky enough to be able to take advantage of this wonderful technology.

Surface Volumes: Non-closed surfaces

Calculating the volume of a closed surface, given a surface triangulation, is straightforward. However an open surface does not have a well defined volume. And yet one often would like to know the "volume" of a pocket or groove on a surface. Grasp will now calculate a volume for open surfaces by applying a "closure" to the surface. To do this Grasp finds all triangles at the edge of the surface and calculates the average position of such. The edges are then "joined" to this position to close the surface and a volume calculated. At present this trick is only performed when the "Build: Cavities/ Connectivities" is called, i.e. to calculate the volume of pocket the user should make that the only surface visible and calculate "cavities". The volume is printed out automatically.

Saving Pictures as RGB files

Grasp will now write out an ".rgb" file for viewing with the SGI function "ipaste". The size of the output is taken to be the size of the current window. The user is prompted for the name of the file. It is recommended that the user check the resultant file with ipaste before moving on. (The image can be corrupted if the textport does not move out of the way fast enough). SGI provide routines to go from rgb files to most other common formats, for instance "tops" converts to postscript. Even black and white Grasp output can often be quite good (for those of us not lucky enough to own high resolution color printers).

Writing Surface Files as PDB files

There are now various options to output surface points not in the usual Grasp format but instead as a "PDB" file. Each atom is identified as an oxygen with a residue name "H20". This makes the surface information more directly accessible (because it is an ASCII file), and also gives the user the ability to read the surface back into the program as a set of atoms with all the attendant facilities.

The three options available are:
Cavity/Water Position File
General Surface Positions
..Ordered by Patch Number.

The first option actually outputs not the surface position but the accessible surface position that corresponds to it, i.e. the center of a water molecule in contact with that surface point. This is only performed on the surface points which are designated as "cavities", i.e. closed concave surfaces. The reason for this is simply to give the user possible positions of internal waters, i.e. waters which would fit inside a cavity. Note that the record for each cavity is written out with a header "Cavity N", where N is the connectivity index of the cavity (the residue number is also set equal to this index).

The second option will output any surface subset. The residue number is one, the atom numbers are sequential, starting at one.

Interpolation Plane Contours

There is now the facility to augment the interpolation plane display with line contours based upon the potentials within this plane. These contours are updated as the interpolation plane potentials are updated, i.e. as the molecule (or plane) is moved. Contours are constructed in exactly the same manner as three dimensional contours, i.e. one enters a set of values and then a set of colors. Contours can be deleted in the same way (although see the section on "Delete" below for a much better method). The thickness of the line contours are adjustable via "Display: Alter: Contour". The default thickness is "line thickness=2". Note there are now three types of contour, 3D, interpolation and surface. Note also that the interpolation plane (Control Y) does not have to be visible for the interpolation plane contours to show.

Interpolation Plane Field Bars

Along with line contours there is another facility for enhancing the interpolation plane. It consists of calculating the field within the plane at every vertex which makes up the plane (which is 65x65 points). The field direction is then indicated by a "bar" at that point. Note the inspiration for this display is the use of iron filings to visualize field lines of a magnet. As such there is no information on the magnitude of the field, nor on its component out of the plane. The "bars" are activated by Control E (E being the universal electrostatic symbol for fields). The color of the bars may be altered via "Set Parameters: System Parameters: Field Bars Color". The user may find it useful to use a darker color when the interpolation plane is visible and a lighter color when not. The default color is yellow.

CrystalEyes Stereo viewing

Grasp supports the stereo view glasses known as "CrystalEyes" by StereoGraphics. For this functionality to work Grasp automatically goes into full screen mode. Menus and the textbox will appear elongated in the Y direction but everything else should be normal. The user should be able to pick on atoms and surfaces and color bars, rotate with the mouse etc. Surface scribing is also possible, though the display of the scribed triangles (blue and green) do not update correctly until the entire scene is redrawn.

There are no exceptions to what can be viewed within Grasp with this hardware but there are a few caveats. Firstly, if something does not appear in stereo it is probably because the user built that something while in stereo mode, i.e. the stereo pair has not been automatically generated upon construction. Simply dropping out and back in to CrystalEyes Viewer mode should force the display into stereo. Secondly, I found the best twist angle to be both smaller and opposite in sign to my own preferred stereo twist angle. The user should probably find their own best value. Mine was -3.5 degrees.

If the program crashes during execution of CrystalEyes then the screen will be left in peculiar state. The way to get out of this is to restart Grasp which will reset the display.

Not all monitors are able to work with the CrystalEyes system. Grasp will

magnitude of the direction vector. Also if two axis are of equal length then it is redundant to give both lengths. Grasp will attempt to build an ellipsoid from what ever information is present, e.g. if there is no axis length assigned it will take it from the direction "magnitude". If only two axis lengths are specified then the third is set equal to the second. If only one is specified, then all are made equal, i.e. a sphere. If the directions specified are not orthogonal then Grasp will attempt to fix this as well.

One can also describe an ellipsoid based upon a center and extrema, i.e. the points at the end of the axes. The formal for this is a line which reads:
Extrema = x y z
This position, relative to the given center (which must be defined first!) defines a direction and a magnitude.

Finally there are two display forms for ellipsoids, namely with a solid surface and with a mesh surface. To set this use: "Display: Alter: Objects: Ellipsoids". There is also a "mixed-mode" display wherein all ellipsoids with an even color index are displayed as in mesh-mode and all others as solids.

Surface Contours

Grasp now gives the user the option of creating line contours on molecular surfaces. These are akin to the interpolation plane line contours described later in this section except in several important categories. Firstly, the contour lines are "permanent" in the same sense as the 3D isopotential contours are, unlike the interpolation plane contours which are updated as the molecule moves. Secondly, contours can be made from any surface property, not just potential. Thirdly, surface contours are real grasp objects, i.e. they have "inheritance" and "intrinsic" properties.

The intrinsic properties presently consist of only the contours color, i.e. one can alter the color of a contour via the contours color index, thus:
lc=2,lcd=7 will change the color of all yellow contours to red.
Improvements will made be to add contour value, property type, line length and area enclosed as properties.

The inheritance properties are more developed. For instance, the line contours inherit formal subset names and so can be moved independently if created from different formal subset surfaces. Presently inheritance is implemented on a "per complete line" basis, not on a "per line segment" basis. This means that each complete line (i.e. all line segments which are connected to each other) has but one pointer, i.e. to the surface vertex first used in its creation. This leaves selection based upon surface properties fairly blunt and this will be improved.

Finally line contour widths can also be altered as per interpolation plane contours, and deleted as per other contours.

Note that an example of the use of line contours is to illustrate "holes" or "knobs" on the surface by contouring local curvature. It can also be useful in augmenting display of surface distances.

subset designation allows the user to move sets of ellipsoids relative to each other if they are created from different formal subsets . For instance a user could build ellipsoids to represent the significant features of both surfaces at a protein-protein interface and be able to manipulate both sets independently.

Other ellipsoid properties are its current color index (default=1) and its creation number, i.e. which describes the order in which ellipsoids where created or read from file (see below).

The command specifiers available to the user are:

ec=n, (specs) : i.e. color all ellipsoids with color n if the specification (specs) include the atom or vertex "belonging" to this ellipsoid

e.g. ec=7,r=lys will color all ellipsoids yellow which where created by lysines.

ecd=n, : select only those ellipsoids with color n

e.g. ec=7,ecd=2 will color yellow all ellipsoids colored red.

en= n, en=>n, en= <n, en= (n,m) : select those ellipsoids whose "creation" number is equal to n, greater than n etc.

e.g. ec=7, en=1 will color the first ellipsoid created yellow.

The user can also save ellipsoid information to file (Menus "Write: Objects File"). The information stored is the center of the each ellipsoid (in the original frame of the molecule), the direction of each axis, the length of each axis (in angstroms), the color assigned to the ellipsoid, the atom number or surface vertex number assigned to the ellipsoid, and, if applicable, the name of a formal subset. Grasp writes this information in a structured form, as outlined in the example below:

```
Object Definition: Ellipsoid
Center=          1.240    6.716   11.995
Direction=      -0.763    0.381   -0.522
Direction=       0.126    0.880    0.458
Direction=       0.633    0.284   -0.720
Axis Length=     5.786
Axis Length=     3.881
Axis Length=     3.190
Color=           7
Atom Index=      1
End Definition
```

The information is presented in the form of a definition. Note that the user can easily modify, or produce their own "record. This format will be extended to other geometric shapes in future versions (examples: cubes, cylinders, cones, torii, lines etc).

There are other possible ways to define an ellipsoid. For instance, instead of giving directions and axis lengths one could just encode the axis length in the

New Features

Ellipsoidal Objects

One of the goals of Grasp is to provide tools by which a user may "objectify" sets of data, for instance atoms, or to produce objects "de novo" to form a true molecular "model". A first step in this direction is being able to construct "minimal" ellipsoids.

The word "minimal" is more hopeful than accurate here. The problem "Given a set of points, what it the ellipsoid of minimal volume which encloses all points" is simple to state yet has proved hard to solve. Grasp currently uses a method which was modified from one suggested by Chris Cortez (Chemistry department, Columbia University). It does a reasonable job of finding the major axis direction, from where the lengths and directions of the two minor axes are found by trial and error. The method works well when the axial ratios are not too far from one. However for some cases it clearly fails to give a very "minimal" ellipsoid (also some sets of points are clearly not well represented by an ellipsoid).

The options available for this algorithm are to build ellipsoids from:
ALL residues
Atom Subset (1 Ellipsoid)
Atom Subset (Many Ellipsoids)
Surface Subset (1 Ellipsoid)
Surface Subset (Many Ellipsoids)

The first option will generate ellipsoids for all atoms non-backbone atoms within each residue. Note that for atoms the van der Waals surfaces are used NOT the atom centers, i.e. the ellipsoid will try and enclose the atomic volumes.

The second option will build just one ellipsoid about any set of atoms the user selects.

The third option will build ellipsoids as in the first option, but for a user selected set of atoms as in the second option, but with the additional set of constraints that each set of atoms selected within each residue is made into a separate ellipsoid (e.g. the user could select all charged side-chains).

The fourth option is equivalent to the second except that surface vertices replace atom spheres. The user could, for instance, select an active site surface to be modeled by an ellipsoid.

The fifth is equivalent to the third except again surface vertices replace atom spheres and connectivity index replaces residue name and number, i.e. all the vertices selected with the same connectivity number are modeled by a separate ellipsoid.

Note that a very important property of ellipsoidal objects is that they inherit both an index to either the set of atoms or the set of vertices which created them, AND any formal subset designation. The index into the atom or vertex list is always the first such atom or vertex which is used in the ellipsoids creation. It allows the user to color ellipsoids based upon the properties of the underlying atoms or vertices (N.B. there are no other operations currently available for ellipsoids from the command line other than coloring). The inherited formal

hp=Hydrophobic/ Polar
aa=Acid/ Acid
bb=Base/ Base

One final note on HINT files is that they describe interactions from atoms to atoms, not from residues to residues as is normally the case. As such, it is not possible to "refine" the position from which the strand begins or ends, since this is already precisely defined.

Potential Maps: Biosym Format

Grasp will now support potential maps made with Biosym's version of Delphi IF they are made of grids which are 65 cubed in size. The position and size of these maps should be correctly inferred.

calculate a map for the first molecule as if isolated, i.e. one needs to remove the dielectric volume and charges of the second molecule from the calculation, and then restore these for the second molecule and remove them for the first and generate a second map. Armed with these potential maps one now needs to interpolate to one surface or set of atoms from one map and to the other surface or set of atoms from the other. Previously, because the interpolation was to all atoms and all vertices automatically, one had to resort to more complicated machinations to achieve the same goal of the independent electrostatics of each molecule.

Interaction Strands: Interaction Type plus HINT Support and More

Standard (Format = 1) interaction strand files now support chain definition. The old fortran format of:
(a3, 3x, i3, 10x, a3, 3x, i3, 10x, 12g)
is now
(a3, 3x, i3, 3x, a, 6x, a3, 3x, i3, 3x, a, 6x, 12g)
i.e. the 10x of blank spaces now allows for a chain specifier. If this field is blank there is no discrimination based on chain.

Grasp can now read "HINT" type files and attempt to interpret the interactions therein for display as interaction strands. HINT is an analysis program developed by Glen Kellogg at the Medical College of Virginia which assigns interactions based upon group types within macromolecules and ligands. The different types of groups are hydrophobic, polar, charged and hydrogen bonding, which leads to six different possible types of interactions, namely:
Hydrophobic
Hydrogen Bond
Acid/ Base
Hydrophobic/ Polar
Acid/ Acid
Base/ Base

Note that the first three are favorable, the last three unfavorable.

The concept of a "type" of interaction was not utilized in Grasp previously, but has now been implemented so as to make use of the HINT classification. Each strand can be assigned a type index from 1 to 6, where the order corresponds to that listed above. These are interpreted for HINT files from the last column of each interaction line. They can also be read from the standard file as an integer between 1 and 6 in the ten spaces to the right of the interaction value, or from format number two in columns 13 to 15.

Once interpreted the strand "type" can be used in selecting interaction strands via the command line format
it=n
where n is the interaction type index. There is also a two letter code which can be used instead of the index number, i.e.:
hd=Hydrophobic
hb=Hydrogen Bond
ab=Acid/ Base

DNAObjects

There are additional representations for DNA bases, namely "ladder rungs" and "transparent boxes". There are also more representations for the DNA axis. The program will also recognize uracil for RNA molecules. The color bar for the color coded "twisted foil" representation is now functional, also in stereo.

Light Source Position

There was a bug in the way the lighting position was handled in v1.0, which sometimes could lead to it being undefined. In addition, the method used to change the light position was not reversible. Both of these problems have been fixed. When the user hits control L the cursor should jump to a screen position representative of the light source.

The mapping is such that the X coordinate of the cursor, relative to the center of the window, is the X-coordinate of the position of the light source if it where on a sphere at the center of the Grasp box of radius 1.0 box units. The Y coordinate of the cursor maps to the YZ circle defined by the intersection of the sphere and the plane x=X. The Y coordinate maps linearly to the angle of the light source on this circle from Y=0, Z=1.0 position.

Moving the mouse from this initial position resets the lighting source via this scheme, and since there is now a direct mapping there is no longer a problem resetting the light position. The light position can also be set in the init_grasp file.

Accessible Area Calculation

The performance of the accessible area algorithm has been improved by about a factor of two. On an R4000 machine at a "dots per sphere" level of 162 (about 1.25 per square angstrom) one should be able to calculate the area of roughly four thousand atoms per second.

Excluded Accessible Area Calculation

When selecting the "type" of area to be calculated for a molecule there now exists an option to choose not accessible or van der Waals area but "excluded" area. This option requires two sets of atoms and proceeds to calculate the area of each set first independently, and then together, storing the difference per atom for each atom in each set, and finally reporting the total difference. For example if the atoms read into Grasp are a complex of two molecules this procedure will report the area of each molecule if it where not in the presence of the other, and the area of each when in the complex, and also the total loss of accessible area. The absolute value of this difference area is stored in the accessible area array. Hence after such a calculation the command: "c=0,S=0" will hide all atoms except those which bury accessible area within the complex, i.e. this is one way to define "interfacial" atoms.

Potential Map Interpolation

When the user interpolates to find the potentials at atoms and vertices from a potential map they are now prompted with menus asking which atoms and which vertices. This can be very useful when the user is investigating the electrostatics of an interface between two molecules. To do this correctly one needs to first

but since several users had told me that they had found the ASCII version of the manual useful I decided to also make it easier to get to. For this function to work there must be an ASCII copy of the manual called "Grasp_Manual" in the Grasp data directory. All Grasp does is call the Iris editor "jot" with a few specifications set. The editor includes a simple "find" feature for those not familiar with it. One can also, of course, add ones own annotations to the manual in this way.

Surface Files:

When surface files are written out they now automatically save the discrete color of each vertex if it has been changed from the default value (of 91). This can be very useful in keeping track of previous work since discrete colors are often used to define temporary subsets of interest.

Color Names Recognized

Grasp now recognizes the color names white, red, green, blue, yellow, magenta, and cyan. It recognizes the first letter of each of these colors and maps these to the colors with index numbers 91-97, i.e. the constant Grasp primary and secondary colors. (Note to get blue you need to specify at least "bl" not just "b").

Shorthand Residue Type Names

For ease of use Grasp now recognizes several residue "group" names. For instance "r=pol" means all polar protein residues. The current names and definitions are listed below.

r=hyd: Hydrophobic residues = ala, val, phe, pro, met, ile, leu
r=pol: Polar residues = ser, thr, tyr, his, cys, asn, gln, trp
r=crg: Charged residue= asp, glu, arg, glu, oxt
r=aro: Aromatic residues = phe, tyr, his, trp
r=ali: Aliphatic residues = ala, val, ile, leu
r=sul: Sulfur containing residues = cys, met
r=hyx: Hydroxyl containing residues = ser, thr, tyr
r=dna: DNA bases = ade, cyt, gua, thy, a, c, g, t
r=wat: possible water names = wat, h2o, hoh, tip

Bond Colors: Transferring, Adopting, Restoring and Undoing

Four command syntax's have been added to aid in the setting of colors of bonds and atoms. They are:

bc=restore : exactly the same as the restore for atoms and vertices
bc=undo : exactly the same as the undo for atoms and vertices
bc=a : transfer the colors of each atom to the bonds belonging to it
c=b : transfer the colors of the first bond of each atom to that atom. (Note that there is no way to know which is the "first" bond for an atom, but since bonds from one particular atom nearly always have the same color this is unlikely to be problematic.)

"HETATM". The file header information is either:
GRASP VECTOR FILE
FORMAT NUMBER= 1
or
GRASP DIPOLE FILE
FORMAT NUMBER= 1

When Grasp reads in such a file it uses the header information to determine where the vector information should go, i.e. into the dipole vector array or into the electric vector array. Note that the header overrides the vector type information present on each line encoding a vector length and direction. Thus a file header which says VECTOR but has each line starting with "DIPOLE" will be read as a electric vector field and vice versa. If header is missing then Grasp will rely on the information in each line. Hence to read in two electric vector fields such that one is manipulatable as electric dipoles is simply a matter of editing the header of either file.

In future releases the blank space between the vector type and the position coordinates will be used to "assign" the vector to an atom so that vectors may more fully conform to the Grasp object specifications of inheritance. In addition, operations on and between vectors will be included.

Box Color

It is now possible to alter the color of the Grasp "box", i.e. the cube with the front face removed which comes up when Control O is hit. The menu cascade is: "Set Parameters: System Miscellaneous: Box/Background Color". The color entered must be a color index, i.e. 1-99. This color index can also be set in the init_grasp file.

Help Menu

The help menu, never a strong point in Grasp, has been slightly improved. The user now has the options:
Control Keys
User Defined Alt Keys
Command Line Syntax
The Manual

The first option lists the functions associated with each "Control" - "Letter" key combination. All key combinations now have functions assigned. If the user selects an entry then they are prompted as to whether they want this function enacted or not.

The second lists the macros which have been assigned to specific "Alt" - "Letter" key combinations. See the section on Grasp macros for further information. Again the user is prompted as to whether to enact a function if it is selected.

The third menu choice simply lists some examples of the Grasp command line syntax. The selection has been updated and is more extensive.

The final menu option will bring up a separate window with the text of the Grasp manual within. As yet there are no further aids in extracting information,

Measurement by Mouse: Surfaces

Grasp v1.0 allowed the user to assign a mouse button to certain measurement functions, specifically the distance between a pair of atoms, the angle between a triplet of atoms and the torsion angle for a set of four atoms. The sets of atoms are recorded by successive picks with the mouse. Grasp v1.1 allows the same facilities for surface vertices, i.e. distance, angle and torsion. These options appear appended to the menu for the equivalent atom functions.

Vectors and Dipoles

There have been many changes in the calculation and display of vectors. Grasp now allows the user to calculate dipole vectors from a subset of atoms, and to calculate more than one such. These vectors are stored internally in an array containing vector length, color, position etc. There is another array for an identical set of information for vectors calculated from electrostatic calculations, i.e. from a potential map. Thus we have a "dual" vector set in Grasp. Each is treated identically except one set is referred to as "electric dipole" vectors and the other as "electric field" vectors. At present the possible operations on a set or between sets is limited, but will be developed. (Note that for convenience field lines are listed in the same menu as dipole and field vectors but are in no other way connected).

Upon calculating either type of vector the user now has three choices as to how to determine the length of the vector or vectors. The first is to assign a fixed length in angstroms to all vectors. The second is to scale everything to the maximum vector magnitude, to which the user assigns a length in angstroms. The final method is to assign an angstrom per vector length, i.e. a scaling factor. Here there is no maximum vector length. Note that the scaling choice applies equally to all vectors of its type, i.e. to all dipole vectors, or all field vectors. One can alter the choice after vector creation by going through the menu cascade "display:alter:vectors:electric dipole/ electric field vectors: size scaling". Note that one can also change the color of a vector type with this sequence of menu choices. Finally this menu path also allows one to alter the "display form" of the vector type. Grasp now supports a simple line vector, which may be useful when graphics power is limited.

Having two vector types may not seem very useful when the methods of calculating them are so different. For instance, suppose the user wants to show two sets of electric field vectors, each a different color. There is a way to do this in Grasp, though not as readily as a user might wish! It is at least possible because Grasp now allows the user to save vector sets to file. The format of these files is two header lines followed by a list of positions and directions for each vector (the vector magnitude is "encoded" in the length of the direction components). The fortran format for each such line is:
(a6,x18,f8.3,x,f8.3,x,f8.3,x,f8.3,x,f8.3,x,f8.3)
The first six characters of the line are either "VECTOR" or "DIPOLE" depending on whether the vector type is electric field or molecular dipole. Note that this is similar to a PDB file, i.e. where each atom position line starts with "ATOM" or

do with "improper" rotations. Alessandro Monge (Chemistry department , Columbia university) , upon whose original code this subroutine is based, brought the issue to my attention that some time the Kabsch algorithm proposes a transformation that is actually both a translation/rotation PLUS an inversion, i.e. one ends up with the mirror image of the original molecule. This can be avoided by ensuring that the eigenvectors calculated with the procedure form a right-handed system. This check has been added and seems to work on examples provided to me (John Gunn, Chemistry department, Columbia).

The other alteration made is that the program now offers you the choice as to whether you want to save the individual atom r.m.s. values that go into calculating the total r.m.s. value. Saving these in some atom property can be quite useful. For instance one use would be to project this value onto a worm representation of the
structures after they are superimposed. This can help illustrate the divergence of the structures quite dramatically. Another would be to calculate partial r.m.s differences for substructures.

Worm

There are several new worm representations to choose from in addition to the original "tube" drawing. The simplest is just a line representation which is useful for viewing really big molecules as it is fast to draw. All the other new representations are "ribbons", i.e. there is a directionality other than that implied in the spline itself. This second direction is chosen so as to follow the peptide plane, modulo 180 degrees. This last caveat simply means that if the peptide plane twists by 180 (as in a beta sheet) then the representation stays flat, not twisted. Another way to look at this is that the direction is such as to stay within the peptide plane and have the maximum projection onto the previous direction. The different "ribbons" available are a flat but rendered ribbon, a ribbon with thickness one tenth the width, wireframes of these two, and finally the more traditional "chorded" representation.

There are also now worm "properties". There are four "intrinsic" properties of twist, stretch, shear and tilt, plus two generic properties. The "intrinsic" properties are not fully developed as yet. For instance "twist" is zero everywhere. This is because the "tube" worm is constructed so that each circular cross-section have no "dihedral" twist relative to the previous section. The property "stretch", i.e. how much each section is along the tube from the previous one tends to be low in helices, high in loops and beta sheets. The last two, "tilt" and "shear" are contrary to "stretch". The generic properties can be used (as can the intrinsics) to reflect underlying atom properties (see "mapping properties onto worms" in the section on improvements of the simple math utility).

Stereo/SplitScreen Viewing

The menu for setting stereo parameters has been altered so that one can also enter the separation of left and right explicitly. One can also store the preferred stereo twist value in the init_grasp file of grasp initialization parameters. Finally Grasp also supports the CrystalEyes stereo viewing system details of which can be found in the section on new facilities.

Mapping Properties to a Worm:

This selection allows the user to propagate atom values from some atom property array to a worm property array. For instance one might want to project atom potentials to a backbone worm. Note that only the atoms used in the creation of the worm are used in this projection since only this information is "inherited" by the worm segments. Other useful possibilities are B-values, distances (especially the distance to the surface) and r.m.s difference from the superimpose functionality (see later). This facility, of representing properties on a worm, has the useful quality of acting as a "volume" sampler for proteins, i.e. illustrates volume patterns, as a counterpoint to surface patterns.

Interaction Matrix:

This new math feature allows the user to map interaction energies FROM the interaction pairs TO their atoms or residues. The options are the average, sum, maximum or minimum for each interaction strand energy originating or ending at a residue or atom. An example of use would be to add up all the interactions from a particular residue. This can be particularly useful in illustrating which sites are "centers" or "hubs" of interaction.

Scribing

Scribing has been modified in several ways. Firstly, it is now a lot safer! The program had a nasty habit of crashing if one scribed off the surface. Now the worst that happens is that the molecule moves a bit.

Secondly, there are two new menu options for the menu "Scribing Action". The first option allows the user to initiate the "fill", i.e. the spreading green wave, from the menu instead of the usual mouse driven method of double clicking upon the same point. The second option allows the user to change all the (blue) border region to (green) fill, rather than, as before, just those in contact with the fill.

An example of the use of these new entries is a method to easily scribe a long surface groove. First make a blue stripe along the groove axis, then use the second new menu entry to turn these green, and then using the "initiate" fill option to start the spreading from this green "line", i.e. rather than from a single triangle. This then gives a linear band of fill rather than a circular fill.

Finally there are two "fill" speeds, slow and fast. When the user double clicks on the same triangle with the left button the speed is "fast", i.e. as fast as the program can calculate and draw (and note that this is a lot faster than the fill rate in v1.0). If the middle button is used the spread speed is one round of connectivity calculations per half second. This is quite slow enough to stop at any desired state. N.B. when fill is initiated from the menu option the fill speed is "slow".

Superimpose

Superimpose has had both options added and a bug fixed. The latter has to

The third option, "f(map) =(map)", contains the newest facilities. The next menu in the cascade gives the options of "point by point function", which means a function is applied to each value at each position on the grid (the result being stored at the same location), "field calculation", by which field quantities can be calculated, and "divergence", meaning the difference between the potential at each site and the average of its six nearest neighbors. The possible functions one can apply via the first option are the same as the those on the "special function" list for atom and surface properties. The field calculations give one the option of calculating the total field magnitude at each grid point or the field component in a particular direction. The options for direction are the absolute x, y or z directions in the molecules reference frame, and the relative, or "screen" X, Y or Z directions, i.e. left to right, up and down and to and fro as the viewer sees the screen. Note that only a field magnitude is calculated, not a vector map which would have 3*N values, where N is the number of grid positions. Note also that values are not calculated for the grid points on the boundary, or sides of the box, since to calculate a field one needs values at all surrounding grid points and these points do not have complete neighborhoods. The divergence of a potential map is proportional to the net source term at each grid point, i.e. the charge or induced charge on the grid. It can also be used to "smooth" potential maps. (If after a divergence calculation the original map is multiplied by 2.0, subtracted from the divergence and the resultant map multiplied by -1.0. A classic case for a Grasp macro!)

Finally there is an option to swap maps, i.e. all values, the scale and midpoint of one map are swapped with those of the other map. Recall that the current map can be set via Control C to map one or two and that the central value, i.e. at site (33,33,33), for that map is printed to the screen when either is chosen. This can be useful in keeping track of which map is where!

Atoms and Surfaces:

As with potential map math the "type" of mathematical operation is now a menu option before one proceeds to the actual operation. The four types of operations are, 1) those where two arrays are operated on together to produce a third, 2) where a scalar quantity alters an array to produce a new array, 3) where an array maps into a new array via an operation and 4) where a scalar quantity is returned from an array.

The options for operating on two arrays are addition, subtraction, division and multiplication. The scalar options are multiplication by or addition of a constant. The operations on an array to produce an array are either to apply a function, i.e. from the special function list, or to "contract" values over each residue.

This last option will, for example, find the sum of an array over each residue and put that number in EVERY entry for that residue in the target array. The other options are average, maximum or minimum. Examples of usage would be to calculate the accessible area of each residue (using "average"), or to find the minimum distance for each residue after doing a "distance" calculation.

The final option, to produce a scalar has the same options as the contraction operation just described, i.e. average, sum, maximum or minimum.

closed or not. By this we mean simply does the surface have any edges? (Alternatively, can we travel across one side of the surface and reach the other side?) Normally when a surface is first created, barring surfacing errors, each surface is complete, i.e. no edges. But supposed we were to select a surface subset, say an active site, and write it to file. Reading this surface into the program as an independent surface gives us an "un-closed", or open, surface. Note that one other difference is that we can not define a unique volume for an open surface (but we CAN define A volume, see below).

This leads us to another possible definition. If we simply uncolor (i.e. vc=0) some triangles then there are "apparent" edges in the displayed surface. These "patches" are actually part of closed surfaces but could be classified as "apparently open". Extending the connectivity techniques to find "patches" becomes a powerful technique. Since the disconnected surfaces are ranked by area we can, for instance, uncolor all vertices of curvature greater than some threshold and find the largest concave portion of the molecular surface. This simple technique usually finds active sites and/or binding pockets.

Since there are actually two types of open surface, real and apparent, we distinguish between them. The real ones we call "open patches", the others are "closed patches". Finally we have extended the cavity display option to include patches, i.e. we can display all patches such that each has a color defined by: {color index = mod(V,9)+1}.

Property Listings

The atom information printed in response to a mouse click on an atom or the end of a bond is determined by the atom information list (pg. 30-31 of the manual). This has been extended to the output in response to a "list" command. For instance, after an atom information list command "ai=pq" the command "list, rn=56" will list all atoms in residue 56 along with the potential and charge of each atom.

Simple Math:

The "Simple Math" utility has been improved considerably. The changes can be subdivided into four types, those involving potential map math, those for surfaces, those for atoms, and new types for mapping atoms properties to worms and interaction matrix math,

Potential Maps:

The rather limited potential map math of v1.0 has been totally replaced. The user now is confronted with four possible types of action.

The first, "(map) op (map) = (map)", deals with mathematics between two maps, i.e. adding maps, subtracting maps, multiplying or dividing maps. The user is prompted for each map in the process.

The second option, "(scalar) op (map) = (map)" allows one to add or multiply maps by a given number. There is also a "Convex correction" option which functions as before.

Upgrades, Fixes, Improvements

Transparency

All surfaces can now be made partly transparent. For instance molecular surfaces, 3D isopotential contours, worms, DNA boxes and the interpolation plane. The degree of transparency can be set at one of three values, namely 0.25, 0.5 and 0.75. The transparency is achieved by bit patterning, i.e. by literally turning some pixels of a surface "off". The lower the transparency value the more holes. One limitation of this approach is that one can not see transparent surfaces within, i.e. behind, other transparent surfaces. However the method is transportable over all SGI machines. Some display objects can be set translucent via their usual display option menu (i.e. surfaces and DNA boxes), others can be set via the menus "set parameters: set transparency".

Solid/ Transparent or Mesh Mixed-Mode Surface (and Ellipsoids)

One can now mix the display mode of surfaces, i.e. have some surfaces or parts of surfaces translucent (or in mesh mode) while the rest is solid. (Note that the secondary mode is either transparent OR mesh). These options now appear in the selection menu for surface display. Grasp uses the discrete color index of each vertex to determine whether part of a surface should be in which mode, i.e. if a surface has an ODD color (i.e. if the color index is numerically odd) the display is solid, otherwise it is translucent/mesh. (Note, there are usually plenty of "spare" colors in Grasp, i.e. no one ever uses all 99 colors, and so in the situation where one wants a SOLID red surface in mixed mode (red=2, therefore is normally in see-through mode) one need only make an unused ODD color, e.g. 55, red via the color palette or editing the "defcol.dat" file, and use this red55 instead of red2)

Finally ellipsoidal objects can also appear in similar mixed mode i.e. solid/mesh, via the same odd/even color index method.

Cavity and Connectivities: Patch Definition and Display.

The original method of finding internal cavities within Grasp is described in the main manual (pg. 43). Briefly one found the connectivity of each surface. Surfaces not connected to the main surface are deemed cavities. This method had its problems if there is more that one "outer" surface! The current algorithm has been improved in several ways. Firstly each surface is checked as to whether it is a convex or concave surface so that all outer surfaces are correctly identified. For good measure each disconnected surface is sorted by area and both area and volume are written out along with the surface connectivity index, i.e. the surface of greatest area is given connectivity index one, and the surface "type" i.e. concave (cavity) or convex (main). (Note connectivity index symbol is V in Grasp and can be used in selecting surfaces and in the surface information list).

The algorithm also recognizes other surface "types". What other types could there be other than concave or convex? There is also whether the surface is

Grasp Version 1.1

Introduction

Grasp v1.1 is the first "official" update since the program was "officially" released in October of 1992. However, the program distributed since then has always been the latest "stable" version and so many users have been using a Grasp which already has many of the features of v1.1, if undocumented. The same is implied for bug fixes. None-the-less the current version has some features which will be novel to all user. Equally important, there is now an INDEX to the manual!

This update to the manual for Grasp v1.1 is both an attempt to bring everyone up to date and to introduce the most significant advance in the program, namely session records. Grasp will now record all actions taken by the user in a readable, editable file. This file can be used to rerun, or "playback" entire sessions or parts of sessions. Furthermore one can construct "macros", i.e. new functionality built by compounding simpler grasp functions. These can be made within the program or by editing session records, and can be run both from the graphical interface and as stand-alone "programs". This facility greatly expands Grasp's utility, examples of which will be described below. From the users point of view one of the immediate benefits will be to both recover from program crashes and to be able to reproduce non-obvious bugs so that I can fix them! Also production of "demonstrations" is now straightforward and I have put several on the net for consideration. (In fact one of the least painless ways to learn of new functionality is probably to run these scripts!).

The improvements in the program can be roughly divided into three types. Firstly, simple fixes and "upgrades", for example grasp never did a good job of recognizing color names (e.g. c=red) and if it did would sometimes give the wrong color! The second type is new functionality, for instance the ability to represent sets of atoms or surfaces vertices as ellipsoidal objects, or to smooth surface properties. Finally the "playback" facility has many ramifications for program use.

This update to the manual is just that, i.e. "to" the manual, i.e. "in addition". I have not attempted to rewrite the manual. Partly because this is too much work at present and secondly because most of the manual is still valid. The next version will no doubt require a more substantial rewrite, but that is, to paraphrase Scarlet O'Hara, "another version". And I think people who already know the program will find that a quick scan through this annex to the manual will bring them up to date on the parts which might affect their usage, where as a new manual would be somewhat more daunting. Finally, I recommend users run the example scripts that now come with the program (better yet, send me your own!).

This version has taken longer to finish than I had hoped. I guess that is inevitable with software. I think the delay has been worthwhile in that I personally find the program much more useful and robust than before. I hope users agree and will get back to me with their comments. On to version 1.2!

Anthony Nicholls, August 1993

References

Curvature
Anthony Nicholls, Kim A. Sharp and Barry Honig
Protein Folding and Association: Insights From the Interfacial and Thermodynamic Properties of Hydrocarbons
PROTEINS: Structure, Function and Genetics, Vol 11, number 4, 1991, pg.282 ff

Superposition
W.Kabsch, Acta Cryst. A32 (1976) 922.

Bonds as Sticks
Kuznetsov and Lim, Journal of Molecular Graphics, Vol. 10 pg.

DelPhi

Small molecule solvation:
Electrostatic Contributions to Solvation Energies: Comparisons of Free Energy Perturbation and Continuum Calculations", Arald Jean-Charles, Anthony Nicholls, Kim Sharp, Barry Honig, Anna Tempczyk, Thomas H. Hendrickson and W.Clark Still. JACS, 1991, vol.113, pg. 1454-1454.(1991)

pKa Shifts:
"On the Calculations of pKas in Proteins", An-Suei Yang, M. R. Gunner, Rosemary Sampogna, Kim Sharp, and Barry Honig, PROTEINS: vol15:252-265 (1993)

Site-Site interactions:
"The Electrostatic Contributions to DNA Base-Stacking Interactions", Richard A. Friedman and Barry Honig, Biopolymers, Vol.32, 145-159 (1992).

Redox Potentials:
"Electrostatic control of midpoint potentials in the cytochrome subunit of the *Rhodopseudomonas viridis* reaction center", M. R. Gunner and Barry Honig, PNAS, Vol. 88, pp. 9151-9155 (1991)

Potentials and Fields around a Protein:
"Computer Simulations of the Diffusion of a Substrate to an Active Site of an Enzyme", K. Sharp, R. Fine, and B. Honig, Science, vol. 236, pp 1460-1463, (1987)

The DelPhi Algorithm:
"A Rapid Finite Difference Algorithm, Utilizing Successive Over-Relaxation to Solve the Poisson-Boltzmann Equation", Anthony Nicholls and Barry Honig, J. Comp. Chem. Vol.12, No. 4, 435-445 (1991).

Grasp Addendum #1.3: History and "!" Commands

The Grasp command line now supports Unix-like history and "!" commands.

History commands print selections of the history list, i.e. a list of all commands entered via the text window. Typing,

history *n*

or

his *n*

where *n* is an integer will cause the last *n* commands to be written to the text window. The default value of *n* is ten, i.e. if n is not entered the last ten commands are written to the screen. History commands are themselves NOT added to the history list, i.e. do not appear in subsequent listings. Note also that only commands which are sent directly to the command interpreter appear on the history list, e.g. selections made during menu driven actions do not appear.

Exclamation commands allow the user to reexecute previous commands stored in the history list. Typing,

!!

or

!

causes the last command to be reexecuted. Note the first (!!) preforms the same function in Unix, but not the second (!) does not.

Typing,

!n

causes the n-th command to be executed, whereas typing

!-n

causes the nth previous command to be executed, e.g. "!-1" cause the previous command to be executed. All commands reexecuted are added to the current history list. An example of the use of such commands would be to incrementally rotate a molecule about an axis, i.e. issue a command such as,

xr=2.0

i.e. rotate about the x axis by two degrees, and then repeat as often as desired using the "!!" or "!" command.

Adding the characters ":p" to an exclamation command causes that command to be printed but not executed (or added to the history list). For instance,

!-1:p

will print the previous command but not enact it. Commands which use ":p" are also not added to the history list. Note however that, unlike the Unix equivalent of this comand, editing of previous commands is not yet implemented.

command a worm subsegment is acted upon if its assigned atom is chosen. The command to alter worm color is "wc" in analogy to "vc", "kc", "ic" etc. For instance,
wc=2,rn=(8,33)
will color red all worm segments belonging to residues 8 to 33.
wc=0, q=-(0) > r
will undisplay all segments which belong to residues which have at least one atom charged.

Grasp Addendum #1.2: Molecular Dipoles

Grasp now allows the user to select a color for the molecular dipole vector when calculated. This is implemented via a menu with color choices. The program also now allows the user to alter the display length of the dipole vector via the menu entry: "Set Parameters: System Miscellaneous: Dipole Length". The user is given the choice of entering a length in Grasp Box units, or in Angstroms. Note that the latter choice has the advantage that the user can also use the dipole as a molecular "ruler" or scale bar, i.e. to indicate local dimensions. The default length of the vector is 0.7 box units.

Grasp Addendum #1.1: Worms

The backbone worm option has been updated. The changes made are listed below.

When the command is made to build a worm, i.e. a tubular B-spline through a set of atoms, the user is given the default option of using all alpha carbons of the protein backbone or entering a different set. This latter option might be used to select only a subset of all alpha carbons, or to select a totally different set of atoms, e.g. amide nitrogens.

If the user has previously constructed one or more worms the user has the option of replacing all of these or adding to them. There may be up to one hundred disjoint splines constructed.

The program now deals with "breaks" in the spline. If two sequential atoms in the selection are more than a certain distance apart the spline is terminated and a new one begun. Hence, if constructing a backbone spline when the protein has more than one chain there will be one spline per chain. The distance used to determine spline breakage can be set by the user via the menus "Set Parameters: System Miscellaneous: Worm Parameters: Spline Gap". If set sufficiently large this will force the construction of a single spline through disjoint chains. It can also be useful in judging patterns of distances of a certain subclass of atoms, i.e. spline breakage contains local distance information.

A B-spline segment is constructed from each set of four consecutive atoms in the selection list. Each segment is made of four subsegments by default. This can also be adjusted by the user by the menus "Set Parameters: System Miscellaneous: Worm Parameters: Worm Segment Density". Note that this change in resolution only takes effect when a worm is built, i.e. does note alter the segment density in chains already constructed. Each subsegment is tube of polygonal cross-section. The default number of sides to this polygon is ten, and this too may be altered via the menu combination "Set Parameters: System Miscellaneous: Worm Parameters: Segment resolution". Finally, the user may alter the radius of the cross section of the worm, i.e. the worm thickness via the menus "Set Parameters: System Miscellaneous: Worm Parameters: Worm Thickness". Note that changes made to these two parameters are reflected immediately in any worm already built.

One limitation to the subsegment density as described above is that the number of subsegments per segment must be even. This arises from the method of assigning atoms to each subsegment. The procedure used is that the first half of the segment is assigned to the second atom in the defining four atom sequence and the second half to the third atom. Hence to be able to equally distribute subsegments there must be an even number of such per segment. Note that this method of subsegment distribution results in no segments assigned to the first and last atoms in a sequence, and only half segments to the second and penultimate atoms.

The purpose of atom assignments is so the user can define subsets of worms for the purpose of coloring or undisplaying. Namely in any worm related

The first file contains a set of default colors the user may wish to use, the second is a Grasp Script which will color atoms by a certain color scheme. The former will be read in upon program startup if it is in the current directory, the second has to be read in via the menu selections "Read: Grasp Script File". Both files can be edited to the users satisfaction.

Appendix C: Grasp Data Files

Grasp comes with certain data files which need to be installed in a known data directory as previously described.

There are two files which are "vital" i.e. Grasp will not function correctly without them. These files are:

default.siz
v3.dat

The first of these is a DelPhi control file which assigns radii when a pdb file is read by the program. It may be edited by the user to include more specific radii assignments (e.g. different radii to different carbons), or to alter the radius values therein, for example setting hydrogen radii to zero. The format of these specifications is described in Appendix A.

The second file is an unformatted file which contains information necessary to the program to perform marching cubes surface construction.

Among the non-essential files, which the user nevertheless should have installed for ease of use are DelPhi charge files:

full.crg
full+backbone.crg
back_no_h.crg
amber.crg
dnarna.crg

The first of these files, "full.crg" describes the charge assignments to each normally ionizable residue, i.e. aspartate, glutamate, lysine, arginine. (Histidines must be charged by the user.). This is a "default" charge set for most proteins.

The second of these adds charges to the atoms N, HN, CA, C and O of the backbone. These "partial" charges are taken from the CHARMM charge set. If no hydrogens are present then the user should use first "full.crg", then "back_no_h.crg", the latter of which compresses the hydrogen charge onto the nitrogen.

The file "amber.crg" contains assignments from the AMBER charge set developed by Kolmann, i.e. a complete partial charge set which will assign charges to nearly all atoms. The last file contains partial charges for DNA and RNA taken from Tung, Harvey and McCammon, Biopolymers vol 23, pp2173-2193, 1984.

Note that because of the variability of atom names in pdb files few guarantees can be made that the protein or DNA will correctly charge. The user should read the section on charging molecules for ways to check this procedure.

Remember that reading charge files does not necessarily wipe out old assignments, i.e. one should not attempt to change charge sets merely by reading in a different charge set without being sure that all previous assignments are overwritten. The user can set all charges to zero prior to reading a new charge file with the command "alt(q=0.0)".

The following files are also included,

defcol.dat
atom.col

Appendix B: .init_grasp Commands

As described in the section "Getting Started" one can set certain initial parameters for Grasp in a file named ".init_grasp". This file can be in the directory the user launches Grasp from, the users root directory, or the data directory pointed to in the environment variable "GRASP". Commands have precedence in the same order, i.e. local over root over data directory. The commands are not case or spacing sensitive but they must contain the exact specifier words as listed below to the left of an equals sign, and similarly for the words or values to the right of the equals sign. Any line beginning with a "!" or a "#" are ignored as comment lines. Lines which can not be interpreted as comments or commands are written to the textport as errors.

Command	Comment
INITIAL DISPLAY = BONDS	Set default molecule display to bonds
INITIAL DISPLAY = ATOMS	Set default molecule display to atoms
DEFAULT ATOM DISPLAY=DISCRETE	Color atoms by colors selected via the command line
DEFAULT ATOM DISPLAY=CHARGE DEFAULT ATOM DISPLAY=POTENTIAL DEFAULT ATOM DISPLAY=DISTANCE DEFAULT ATOM DISPLAY=GPROPERTY1 DEFAULT ATOM DISPLAY=GPROPERTY2	Color atoms by this property
DEFAULT SURFACE DISPLAY=LIT	Surface is lit by SGI calls
DEFAULT SURFACE DISPLAY=PSEUDO	Surface is lit by Grasp calls
DEFAULT SURFACE DISPLAY=MESH DEFAULT SURFACE DISPLAY=POINTS	
DEFAULT SURFACE DISPLAY=DISCRETE	Color surface by colors selected via the command line
DEFAULT SURFACE DISPLAY=CURVATURE DEFAULT SURFACE DISPLAY=POTENTIAL DEFAULT SURFACE DISPLAY=DISTANCE DEFAULT SURFACE DISPLAY=GPROPERTY1 DEFAULT SURFACE DISPLAY=GPROPERTY2	Color surface by this property

numerical values (in any format). On the same line as the above key words the user should have one of the following: "potential", "distance", "curvature", "gproperty1", "gproperty2", "accessible", or "charge", e.g.
surface=distance
will inform the program that the list of values should be placed in the distance array for surface vertices.

Grasp Script File.

An ascii file the user can store commands in to executed together. For instance a set of coloring commands for atoms. Any line beginning with an exclamation mark (!) are disregarded. All other lines are sent to the command interpreter to be acted upon.

Interaction Energy (Matrix) File.

These come in two flavors. Both should have as their first line the words,
GRASP RESIDUE INTERACTION
The next line should either be,
FORMAT= 1
or
FORMAT = 2

The first format is more complete and expects lines in the FORTRAN format, (a3,3x,i3,10x,a3,3x,i3,10x,12g). Here the a3 is the first residue name, i3 the residue number, a3 the next residue name, i3 the next residue number, finally 12g is format of the interaction energy.

The second format contains only the residue numbers and the interaction strength, i.e. (i5,x,i5,4x,g12). There can be at most 10,000 lines of such descriptions. There is no provision for comment lines. Reading in a file automatically replaces all previous interaction values.

The same difference in mode of assignment between DelPhi and Grasp occurs as is described above for charge files, i.e. the user should be careful to put more general assignments first if the same assignment is to be made by both programs.

Both charge and size files are supported by Grasp for a DelPhi users convenience. However the Grasp user might want to consider using neither and instead using the more general format of Grasp "alt" (see section on command line syntax) commands in a script file to assign radii and charges.

Protein Data Bank File

The complete specification for these files is quite complex. Grasp (and DelPhi) only use information on lines beginning with "ATOM" or "HETATM" or "TER" (the latter only Grasp). These lines are expected to have format,
(A6, 5X, A5, X, A3, X, A1, A4, 4X, 3F8.3)
in the first fifty four characters, where A6 is the"ATOM " or "HETATM" header, A5 is the atom name (only four of which are used in Grasp), A3 the residue name, A1 the chain name, A4 the residue number and 3f8.3 the atom coordinates in Angstroms.

In some files the field to the right of these (i.e. characters 55 to 80) are also used to store information. In standard PDB format columns 55-60 are used for the occupancy (F6.2), columns 61-67 for the B-value (F7.3). The Grasp variants are as follows.
If the first two lines of the file are:
GRASP PDB FILE
FORMAT NUMBER= 1
then these two fields will be read in as the radius and charge of the atom in that order. If the first two lines are:
GRASP PDB FILE
FORMAT NUMBER= 3
Then the values in these fields are read into general property arrays one and two respectively. If instead they are:
GRASP PDB FILE
FORMAT NUMBER= 2
then the entire field of 55-80 is read in free format as general properties one and two. This clearly allows the user to store values to much higher accuracy.

Note that in using any of these formats there MUST be two numbers in *some* format or an error may occur.

End of molecule statements, i.e. "TER" lines should occur after each molecule described in a PDB file for Grasp to treat them as independent molecules.

Grasp Property File

These ascii files contain a listing of a single property for either all atoms or all surface vertices. The user may write as many comment lines as desired before a line which has the key word "surface=" or "atoms=" (so be careful not to include these in comment lines). All lines after that are assumed by the program to be

DelPhi Charge File

This file can have any number of header comment lines as long as the first character is an exclamation point "!". After the last comment line should appear a line as below,

atom__resnumbc_charge_

Any comments after this line must be to the right of the first twenty second character of the line, since this part of the line is not interpreted as an assignment.

To make an assignment statement the user places the atom name specification under the four characters "atom" in the above line, the three characters of the residue name under "res", the number of the residue under "numb" and the chain designator under "c". Finally the charge value is placed under "charge". If a descriptor field left totally blank is treated as wild, i.e. if the chain specifier is blank the assignments spelt out by the other fields in that line are applied to all chains in the molecule. However, blank spaces *within* a field are treated as true blanks, i.e. a "c " in the atom field will not apply to a "ca " atom. This is different from how DelPhi radii files are dealt with as is described below.

Individual assignment statements are interpreted identically in both Grasp and DelPhi. However the method of eventual assignment differs. In DelPhi each such specifications are entered in a hash table. When *all* lines are read the charge assigned to a particular atom is that which is *most specifically* declared in the assignment statements. On the other hand the method Grasp uses to assign charges from a DelPhi charge file is that each line is interpreted individually before the next is read.

As an example of the different approaches, the lines:

```
atom__resnumbc_charge_
nz       lys 25   A   0.0
nz       lys             1.0
```

will charge all zeta nitrogens of the molecule for Grasp, and all BUT lysine 25 on chain A in DelPhi. If the line order where reversed however the charging would be identical since Grasp would change the charge on lysine 25 on chain A back to zero after first setting it to 1.0. Hence when using DelPhi charge files in Grasp be sure that the more general charge assignments precede the more specific.

DelPhi Size File

The radii assignment files in DelPhi are similar to, but simpler from, the charge assignment files. There may again be comment lines before a "key word" line, which in this case is,

atom__res_radius

i.e. size files only contain room for atom names and residue names.

There is one further difference between DelPhi size files and radii files, namely that in size files empty spaces *within* a descriptor field are treated as wild cards. Thus "c " under the "**atom**" header in a size file applies to all atoms whose names begin with "c".

Appendices

Appendix A : File Formats

The following file types are described below:

Grasp Surface File
DelPhi Potential Map
DelPhi Charge File
DelPhi Size File
PDB File (and Grasp variants)
Grasp Property File
Grasp Script File
Pair Wise Interaction File

Grasp Surface File

This unformatted file starts with five "lines" of eighty characters. The first line contains a format specifier, i.e. the words "format=1". (There are no other formats at this time.) The second and third lines contain key words for the information contained within, i.e. "vertices" for the vertex positions, "accessible" for the associated accessible surface point coordinates, "normals" for the normal vector (of length unity) for each vertex, "triangles" for the triangle index list. The latter is a list of integers such that entries i-2, i-1 and i, where mod(i,3)=0, give the which vertices make up triangle i/3. NOTE that the index integers for the triangles are INTEGER*2 ! Line three contains which variable are also written to this file, "potentials", "curvature", "distances", "gproperty1", "gproperty2" being the key words for the appropriate quantities. Line four contains the number of vertices, the number of triangles, the grid size of the lattice used to create it (i.e. the number of points along one edge of the cube, always =65), and the reciprocal lattice spacing. Line five contains the midpoint of the coordinate system from which the vertices were derived (i.e. the midpoint of the Grasp box). The data then follows in the order of the keywords. Note that all are REAL*4 except for the triangle indexes which are INTEGER*2.

DelPhi Potential Map

This is also an unformatted file. Its contents follow the format:

character*20 uplbl
character*10 nxtlbl, character*60 toplbl
real*4 phi(65,65,65)
character*16 botlbl
real*4 scale, mid(3)

Here *phi* contains the map information, *mid* the grid midpoint, *scale* the reciprocal grid spacing. The rest are just character strings containing non-Grasp information.

curvature). This is time consuming to calculate for large molecules, however once the calculations are performed for two such, the *differential* occlusion between them is easier to calculate. In short we hope to have an immediate estimate of the buried area as two molecules are moved relative to each other.

The final energy term, desolvation, is the hardest to calculate precisely. Typically it involves numerous DelPhi calculations each of which takes up to a minute on a low-end Iris. However, there are certain short cuts which can be taken when the problem is to calculate the relative change of solvation for the arrangements of two otherwise static molecules. These revolve around reformulating desolvation as a surface term akin to accessible area.

For both the hydrophobic and the desolvation penalty the approach that will be taken is to "prepare" certain sets of data for each molecule before attempting to "dock" them. This approach will also be taken to improve the speed of calculating the Van der Waals and Coulombic terms. This is especially important in the case of the Coulombic term because of the long range nature of the force precludes continuous direct (i.e. exact) calculation except where a small number of charges are involved (i.e. typically a hundred for each subset).

The resultant energies will only be "approximate" in that they will not exactly reproduce the numbers that would be obtained from a precise application of each theoretical model. But since we should remember that each theoretical model term is only a *model* of physical reality the more important question is as to whether these approximations will still accurately reflect enough of the underlying physics to be useful. For instance, providing a sufficiently large penalty for burying two oppositely charged groups together compared to the favorable Coulombic energy. This will probably only be truly tested by extensive use.

The energies calculated could be displayed both numerically and graphically as the molecules are arranged. In some cases where the contributions can be made local (e.g. Coulombic) these can be displayed as an atom or surface property.

Using reduced surface representations it might be possible to automatically dock molecules together based upon their shape and local properties. One such promising method is to use geometric hashing to check all such matches. Such approaches will be investigated.

Docking with Realistic Energies

The docking of molecules, either by hand or by some automatic procedure, is of particular interest because of the importance of shape (i.e. surface) complementarity. As has been described earlier Grasp allows the user to "invert" a surface. When this is used for interacting surfaces it allow the user to match "like-with-like" rather than "knobs-with-holes". However, the physical docking of surfaces or molecules is hard to do manually. One project for Grasp is to improve the visual aids to a user attempting to do just that, and also to provide numerical feedback, for instance distance calculations and energy analysis in as close to real time as possible.

Graphically many of the tools are already present, for instance the independent manipulations of objects and the split screen capability. However these can be greatly enhanced with some additional features. One example would be "proximity strands". Given a certain set of points on one subset one might want to know the nearest set of points on the subset being moved relative to it. This can then be graphically illustrated by drawing strands between the pairs of points. These could then be dynamically updated as the user moves the subsets relative to each other. The split screen functionality can be used to great affect here. If the subsets are arranged on the right in an orientation such that the interface regions of both are clearly visible, while the dials are attached to the left view, the strand positions could be *updated* based upon the apparent left view arrangement, but *displayed* on the right, were they may be more clearly seen. The clarity of view can be further enhanced beyond just a different relative arrangement on the right-hand view. If one is looking at a "deep" binding site, e.g. a small molecule into a deep cleft, then on the right-hand view the user could split the cleft into two parts along its axis and "peel" each part back, i.e. exposing the "interior". This then further avoids any loss of visual information due to local topology, i.e. the small molecule obscuring the cleft.

Another approach which would help in docking molecules is to allow rotations and translations to be preformed in a "local" coordinate frame, i.e. one linked to the molecular shape, rather than the XYZ system of the screen. For instance in docking of an elongated molecule into a deep pocket of another one might want to assign one axis pointing into the axis. Such local coordinate frames can be confusing in general and so a good compromise would be to allow the user to switch back and forth between a "local" and a "global" frame.

Speeding up distance calculations to the point where they can be done in real time would also have a useful graphical impact. At the moment such algorithms in Grasp are quite slow but by using certain grid techniques it ought to be possible to speed this approach greatly. Similar comments would then apply as for proximity strands to the uses of a split screen approach.

Energy analysis can be split into four classes of energies, namely Coulombic, Van der Waals, Hydrophobic and Desolvation. The first two terms are traditional in the sense that they are included in packages available elsewhere.

The hydrophobic contributions can be assessed, at least according to theories developed in this lab, as being proportional to the excluded accessible area, where elements of this area are weighted by their accessibility to water (aka

the ribbon can display directional properties of a residue and, along with the circular tube can represent at least one property by via its size. Each element has as intrinsic properties of direction, a rate of change of direction, a twist and a rate of change of twist (higher derivatives may or may not be meaningful) either from the form of the line or from the intrinsic residue direction. Each element also inherits all the properties of the underlying residue. Any such scalar variable can be represented as the width of the line structure or as a color as with surface or atom colors. Any vector property can be set to be the intrinsic residue direction.

Hence this approach will give each residue an object i.e. the line segment, which comes in several flavors and which can represent both vector and scalar properties as well as having intrinsic properties.

One interesting example of the intrinsic properties is the variation from ideality of a helix or sheet along its length. This is often a difficult property to visualize but, as occurred with surface curvature, the quantification and color coding of this property will probably be highly instructive.

Secondary Structure Display

Why another secondary structure program? After all there are some excellent representation programs available, e.g. RIBBONS by Mike Carson. Specifically because I think there is a lack of emphasis in such programs on secondary structure representation representing anything except structure. Grasps philosophy of representation is that as well as representing structure objects must also be able to represent properties of the underlying atoms or vertices, and may also have intrinsic properties. Hence elements of a secondary structure representation should also be able to have their shape and or color altered by underlying properties such as hydrophobicity, potential, B-values as well as intrinsic ones such as twist, curvature, strain etc. The following is the envisioned implementation of Grasps secondary structure package.

Each residue will have a position and a direction associated with it. The position, or point, might be the atom center of the alpha carbon, or the average backbone position, or the middle of the side chain. It might also be the the closest point on the axis of a "perfect" helix defined relative to this residue. The direction might be the along the C-C beta bond, i.e. pointing towards the side chain, or might be the direction to the next residue, or to the nearest non-neighboring residues. Both the point and direction definitions can be user implemented with a considerable degree of flexibility.

Given a set of points, i.e. those belonging to a set of residues along a chain, a smooth line can always be drawn. Popular "splines" are the B-spline because of its inherent smoothness, and "Cardinal" splines which pass through control points. Each residue can then be assigned a line "segment". (In essence this is a property of the difference in positions of the points belonging to sequential residues before and after each residue). This line segment has dimension one as opposed to that of the point for that residue which has dimension zero.

The residue line can be expanded into a sheet, or ribbon, of two dimensions given a direction. This direction could be that assigned to the residue, or could be derived from properties of the line itself, i.e. the second derivative of the line at each point.

Given a direction associated with each line segment one can also define a twist, i.e. the variation of this vector as a function of distance along the line. The variation of this quantity is an interesting property in its own right. It should be constant for perfect sheets and helices, and random for true random coil.

The residue line can also be expanded to three dimensions into a tube. This is the most common representation seen today. The tube can be circular in cross-section, i.e. having no directional character, or be ellipsoidal in which case there is directionality in the long axis. This latter can again be correlated with the intrinsic residue direction or with that from the line segment. Circular tubes have one quantity i.e. thickness, ellipsoidal tubes have two, i.e. the lengths of the minor and major axis.

As described above we have four logical values we can assign to each segment, namely it can be a line, a ribbon, a circular tube or an ellipsoidal tube. For instance we can assign lines to random coil segments, circular tubes to helices and ellipsoidal tubes to beta sheets. Furthermore the ellipsoidal tube and

9) Surface Area, Curved Surface Area, Van der Waals energy, Coulombic Energy.

This can be used individually on any molecule or subset of a molecule. Output can be contracted to give totals, or assigned as properties within Grasp.

Programs and Grasp Features

1) Facilities to make changing and applying charge sets easier. This is often the most time consuming part of DelPhi, as currently implemented, because of non-standard naming conventions for atoms.

2) De novo charge sets. For new structures, e.g. drug molecules, it will be important to have a charge set of *some* kind. Doree Sikoff has Scheraga's program implemented which will act as a first step in this direction.

3) Proton placement. Marilyn Gunner's program to assign positions of protons based upon geometric constraints.

4) PK analysis. Programs from Gunner and Yang to estimate effective pK's of ionizable residues.

5) Solids. Grasp facility to include geometric shapes in the dielectric calculation

2) Find the potentials at a given set of points given a certain charge set.
These points would usually, though not exclusively, be at atom coordinates. This charge set could be the charges of all atoms or a subset of atoms. Results can be incorporated into Grasp, or exported to external files. Potentials can be broken down into contributions from permanent charges, polarization and from mobile ions, or given as total electrostatic potentials.

3) Find the interaction between two sets of charges.
To some extent this is a subset of 2), i.e. given a set of charges find the potential at a second set of sites. The work of extracting the *interaction* energy from this type of calculation is done for the user. One of the uses of this would be to find the effect of one ionizable residue upon another. Another would be the electrostatic interaction between two molecules.

4) Total solvation energy.
By this is meant the total *electrostatic* energy involved in the creation of a dielectric discontinuity between molecule and solvent.
Because of the nature of the algorithms used it is often much less computationally expensive to find the *difference* in solvation energy for two or more molecules, or conformations of a single molecule, than to calculate the individual values for each. Hence the user will have the option to specify a set of structures and take advantage of the improved speed of these difference calculations.

5) Construct an Interaction Matrix.
Use function 3) repeatedly to find the interaction between all members of a set of residues. Then use 4) to find the energy of charging a residue on its own. The former produce the off-diagonal and latter the diagonal elements of an interaction matrix which can be used in ancillary algorithms to estimate effective PK's of residues.

6) Total transfer free energy.
The total electrostatic energy of transferring between solvents, plus a contribution from volume effects, plus a contribution due to hydrophobicity.

7) Total Binding Energy.
Total electrostatic energy between two sets of atoms plus a contribution from Van der Waals energy and hydrophobicity.

8) Helix / Membrane Affinity.
Membrane affinity based upon an approximate desolvation penalty, plus hydrophobicity, per residue in the context of the primary sequence as an alpha helix.

ID: Intelligent Delphi

Intelligent **D**elPhi (Id) will automate many of the tasks performed by DelPhi, taking unto itself the task of organizing multiple runs and collating relevant data. Parameters such as dielectrics, salt concentration etc. would be set in a "DelPhi Panel", independent of the particular use. Those uses are outlined below.

The question of accuracy, where appropriate, would be settled via an option for "high", "medium" or "low" precision. These range would refer to accuracy ranges of roughly 0.1 kt, 0.25 kt and 1.0 kt for the particular quantities required.

Obviously more accurate calculations would require more time. Id would indicate the expected duration of the calculation before running, provide regular (visual) updates on the progress of the calculation, as well as options to abort the calculation.

Delphi is not always necessary or sufficient for all functions. For example there is a requirement for algorithms for Van der Waals energies, surface area, and molecular volume. Hence Id is also a superset of DelPhi functionality.

All functions which produce single number answers, such as total solvation energy, will produce break downs of these numbers to include, for instance, the effect of salt, if present in the calculation, and any other subsets which go into a composite calculation.

The user will have the option of saving results at various levels of detail to files. Also log files and files used temporarily by ID will be available for the user to track if they wish, or in the unlikely case of a crash, to find out what went wrong.

Functions:

1) Find potentials in a cubic box .

This would be similar to what is done now in Grasp except that the user could specify the size, orientation and position of the box.

This box would be visualized in Grasp and its parameters altered by a widget. Alternatively these parameters could be entered explicitly.

The size could be via the molecules size (i.e. percent fill), or absolute size (e.g. 100 Ångstroms). The molecules size can be defined either by the maximum dimension in the X, Y or Z direction, by the maximum length of the principal axis of the molecule, or to the largest separation between any two atoms.

Orientation can be determined relative to the principle axes of the molecule, or determined by the direction of maximum separation of any two atoms, or by any two points the user cares to define.

The central position could be by atom set or surface set, or entered explicitly.

If the DelPhi box boundaries are too near the molecule focussing runs would be automatically queued.

All other DelPhi parameters, such as salt concentration, dielectrics, etc. would be entered explicitly, with defaults for all values.

chains.

S. Sridharan has written much more efficient algorithms for the calculation of surface areas and also the curvature of those area elements. These will be interfaced to Grasp. He also has algorithms for the prediction of ion binding sites based upon electrostatics. These will be improved with desolvation and hydrophobicity penalties and added to the program.

Generic objects will be introduced to represent sets of atoms, e.g. cylinder, spheres, regular ellipses. These may also be included as low dielectric volumes in Poisson Boltzmann calculations. Properties of underlying atoms and residues may then be inherited by these objects and displayed thereon. Map properties may also be interpolated at the surface of such objects. These objects can be thought of as tertiary structure representations in analogy with those for secondary structures.

Reduced representations of surfaces will be introduced. Atoms and groups of atoms can be represented as spheres, worm lines, backbone boxes etc. In keeping with the duality within Grasp of vertices and atoms similar simplifications will be made for groups of vertices.

Grasp originally had animation facility i.e. the ability to display successive frames of a simulation based upon Discover (molecular dynamics from Biosym Technologies) output. It was removed until a better interface for this feature may be developed. This may include animation of reduced representations in analogy with that for DNA structures.

The secondary structure type of each residue will be assessed, either by previously published criteria or by internally adjustable constraints (e.g. hydrogen bonding patterns, phi and psi angles etc).

Material surface properties will be adjustable. The number of properties accessible to .init_grasp will be greatly increased. A global rescale will be included. Previous commands will be accessible via a "history" command as in Unix. Text labelling will be added.

Bugs will be fixed.

There are three specific projects which will developed as "modules" for Grasp. These are an interface to DelPhi which will also act as an expert system for that program, a secondary structure program which will develop the "worm" representation, and a docking program which will allow for the rapid assessment of the energies involved in the rearrangement of the position of two molecules.

Future Developments

General

The following is a digression into what I hope is installed within the program in the next six months, even though software completion is unpredictable.

The drawing of surfaces in Grasp is done in a *per triangle* basis, i.e. one triangle at a time. While this approach simplifies the drawing of partial surfaces it is not as efficient as *per mesh* approach. This means that collections of connected triangles are drawn together, taking advantage of the SGI *mesh* subroutine calls. S. Sridharan has produced general meshing algorithms which increase drawing rates by a factor of 2.7 compared to a theoretical maximum of 3.0. This will speed the drawing of any complex surface, i.e. molecular, contour etc.

Rex Bharadwaj has been responsible for the DNA objects within Grasp. At the moment it relies upon external programs to calculate DNA parameters. It would be easier to generate some parameters internally, i.e. the ones most often used like base pair twist and tilt. There will also be support for other DNA parameter scales than those provided by CURVES. An interface will be constructed which allows the user to display the results of dynamical simulations in object form. Reference forms of DNA will be included for comparison purposes. The ability to deal with multiple structures, "pick" objects and return values plus access to Grasp tools such as the color scale will also be included.

Following the object philosophy of Grasp residues will be given their own object, namely an ellipsoid enclosing the Van der Waals surface of the residue. At the same time residues will be given a more independent status, i.e. similar to that atoms and surface vertices have at present, e.g. having properties distinct, though usually derivable, from the underlying atoms. Instances of such properties might be residue dipole, residue area, residue volume, distance to other residues, phi and psi angles etc. This will facilitate mapping of residue properties onto residue objects, i.e. color coding by property, and onto surfaces. It will also allow the user to write and read residue property files.

Local averaging and differencing of properties will be introduced for surfaces and atoms and maps. This will allow for smoothing of properties and also the calculation of new ones, e.g. field strength for maps. Cross-surface distances, i.e. as opposed to Cartesian distances, will be added as a property.

A "math" interpreter will be written which will allow for more complex mathematics on surface and atomic properties. This will allow for conditional operations on sets of data which are hard to perform within the program at present. A simple graphing utility may be added.

Hydrogen bond representation and calculation will be added. As an object these will have intrinsic properties, e.g. angular and distance differences from "ideal" structures, as well as inheriting the properties of the atoms forming the bond. As with matrix representations the user will be able visualize hydrogen bonding networks, i.e. hydrogen bonds which share atoms, or residues, or

Subsets: Make a Formal Subset: Atoms: Make All Molecules Subsets". The program will then prompt the user with its chosen names which will be "m1", "m2"..etc. The helices are now formal subsets. The dials are attached to helix (i.e. molecule) six, since this was the last subset created. Move this helix to the right until it is clear of the other helices. Then attach the dials to the world with menus "Formal Subsets:Fix Dials to World" and rotate the view about the Z axis by 60 degrees. This might be more easily accomplished by the command line rotation command,

zr=90, sub=m1

Then attach the dials the helix 5, via menus "Formal Subsets: Fix Dials to a Subset: m5", move it to the right until clear of the other helices, then fix the dials to the world, rotate 60 degrees about the Z axis, fix the dials to helix 4 etc. The end result should be all six helices are separated and roughly at the corners of a hexagon.

Next change to "Active" rotations by hitting Control A. This means the helix moved will be the one the cursor lies upon. Adjust each helix to a "comfortable" position relative to the other helices in this mode, remembering that the world view can be changed by starting with the cursor off of all molecules.

When the user is happy with the arrangements the apparent relative positions should be fixed so that surfacing can begin. Remember that the internal coordinates of the molecules have not been changed yet and it is these that most functions within the program, such as surfacing and the Poisson Boltzmann solver, use. To fix the coordinates, select "Formal Subsets: Fix a Subset Rotation: m1", then "Formal Subsets: Fix a Subset Rotation: m2" etc. until all are fixed. Note the subsets are still formal subsets. The user might also want to "remove" each subset after "fixing" it. Removing only affects the *name* of the subset, i.e. it merely rejoins that subset to the "world" object. This would prevent the user from spoiling a careful arrangement by accidentally moving a subset.

At this point the user can perform any Grasp operation as if the assembly had just been read into the program, for instance create a molecular surface, do an electrostatics calculation etc.

Finding and displaying all residues within 3 Angstroms of an active site.

The cleanest way of selecting an "active site" is to use the scribing function in Grasp, i.e. to literally draw on the surface what one considers the active site to be. The method of doing this is described under the section on menu use on "Mouse Functions". So we will presume the user has mastered this procedure and has the surface of the active site a bright green. The next step is to create a distance map for all the atoms. This one does via the menus, "Calculate: Distance Map: Atoms to Surface: VdW Surface to Surface: All atoms: Currently Scribed Surface". This will calculate the nearest surface vertex of the bright green subset to each atom, and subtract from this distance the atoms radius.

Once this calculation is complete we want all *residues* within three angstroms. Since we only have atom information we want to use the "projection" syntax described in the Command Line section of the menu, i.e. if any atom of a residue is within three angstroms we want to select it. One way to so this is a command such as:

c=2, d=<3.0 >r

i.e. all atoms of all residues fulfilling this criteria are colored red. Note the command c=0, d=> 3.0 > r

will "uncolor" all atoms of all residues which have at least one atom at a distance greater than three angstroms, which is NOT the compliment of the previous command.

Finding the common volume between two superimposed molecules.

This example requires the user to have loaded two fairly similar molecules. The section on using the superimpose function describes how to successfully superimpose two molecule. To find the common volume then select the menu entry "Build: Consensus Volume". This creates a grid map where a value of zero means outside of both molecules, one means inside of either molecule but not both, and two means inside of both. This map is stored in map two. We want to contour this map, so we need to ensure the current map is number two. Do this with the menus, "Miscellaneous: Change Current Map: Inputted Delphi Map (Map2)". Next choose "Build: Contour", and enter "2.0", then "4". This should produce a blue contour which represents the boundary of the "consensus" volume. The final step is calculate the volume of this contour which is done with the menu selection, "Calculate: Volume of a Surface/Molecule: Contour: Value= 2.0" which returns the volume of this contour in cubic angstroms.

Form a six helix bundle from a single helix and surface the result.

First select a helix. For instance, choose a helix of sufficient length from a larger molecule. Write these atoms to an external file, exit Grasp and restart it with this new file (as Grasp has no way to "delete" atoms). We want to reproduce this helix. The simplest way is to read it in five more times. This result in six identical helices all exactly on top of each other. Align these helices so that the long axis is along the Z axis, i.e. into the screen.

To manipulate the helices independently of each other we need to make formal subsets of each helix. A quick way to so this is via the menus "Formal

two dielectric values to when there is just one, i.e. it is the ratio of potentials at the site for a 2/80 Poisson-Boltzmann calculation to that from Coulombs Law for a dielectric of 2.

First we have to charge a single site. Instead of using the command line "alt" command we will illustrate the use of the "mouse command line" function. Select "Mouse Functions: Command Line Mouse", then enter "alt(q=1.0)". This connects the function of assigning the charge of plus one to the act of picking an atom. The user can now select an atom on the molecule and charge it instantly. To remove this function repeat the menu selection.

Next calculate a new potential map (see example one). Then store this map in map two by the menu selections, "Simple Property Math: Potential Maps: Swap Map1 and Map 2". Next set the external dielectric to the internal dielectric, i.e. after "Set Parameters: Electrostatic Parameters: Outer Dielectric" enter "2.0". Then recalculate the potential map. Now the uniform dielectric results are in map one and the two dielectric results in map two. We want to divide map one by map two, which we do with the menu combination, "Simple Property Math: Divide Map1 by Map2". The result is that map one now contains "effective" dielectric values for all its grid points. Selecting "Calculate: Pot. via Map at Surfaces/Atoms" then interpolates these values at every atom and vertex. These can then be displayed as usual.

Note that we are using a uniform dielectric grid calculation to estimate the Coulombic potential at every atom/vertex. Since there is only one charge this could also be done by calculating a distance map and then using the "simple math" utility a couple times, along with the fact that a charge in dielectric of two produces a potential of 280.5 kt one angstrom away. Considering the inaccuracy involved in the two dielectric calculation this is hardly worth it. The values calculated should definitely be treated with caution for this reason, especially close to the charge where grid effects will be largest. However this is a calculation worth doing at least once to get a feel of the effect of two dielectrics on the potential distribution, i.e. of the distance at which solvent screening takes effect.

Calculate the average surface area per hydrophobic and hydrophilic residue.

First calculate the total surface area of all atoms, i.e. Calculate: Area of a Surface/Molecule: Molecule: All atoms". This fills the internal atom array for accessible area. Next choose "Calculate: Simple Property Math: Atom Properties: Enter Selection". At this point enter "r=hyd" and hit return. Then from the next menus select, "Sum of Values: Accessible Area". The will result in the printing of the total area of all hydrophobic residues. Substitute "r=pol" for hydrophilic residues. Note that we do not want to choose "Average Value" here because the average is over atoms not residues. To find how many hydrophobic residues there are try coloring the alpha carbons of each such residue, i.e. enter "c=1, a=ca, r=hyd". The number of such atoms colored will be the number of such residues. Similarly for hydrophobic residues. Note that the residues included in the definition of "hydrophobic" or "hydrophilic" are described in the section on command line syntax for atoms.

surfaces with the menu sequence "Display: Alter: Molecular Surface: Distance: (Surface Display Mode)". To bring back parts of the surface that are hidden give the command "vc=1".)

Calculate the occluded accessible surface area between these parts.

Following on from the previous example, select first the following menu options, "Calculate: Area of a Surface/Molecule: Molecule: Set A" . Here "Set A" means select the same set as used to create the first surface as in the previous example. This will result in various text appearing in the textport. At the end of this will be a number, the total accessible area for that set of atoms. Make a note of this number, then select "Calculate: Area of a Surface/Molecule: Molecule: Set B". This will give the accessible area of the second set. Make a note of this number then select "Calculate: Area of a Surface/Molecule: Molecule: Set A and B". This will give the area of both sets in the context of each other, i.e. the difference between this area and the sum of areas for A and B separately is the occluded accessible area between A and B. Note that it may not be simple to select the set "A and B". One way to do this is to NOT everything except A and B. For this one can make use of color as a subsetting tool. For instance, suppose A consists of residues (20,34) and B residues (39, 45). Then give the following three commands, "c=1", "c=2, rn=(20,34)" then "c=3, rn=(39,45)". The combination of A and B can then be selected by the subsetting command "cd=-1".

Reading in atomic B-values from a PDB file, displaying them on the surface.

Edit a PDB file containing B-value information such that the first two lines read:

GRASP PDB FILE
FORMAT NUMBER= 3

This provides a hint to Grasp that the B-values in columns (61-67) should be read and interpreted. Upon reading this modified file the B-values will be stored in general property 2. To see the atoms displayed such that they are color coded by B-value select menus "Display: Alter: Atoms: General Property 2: (Atom Display Mode)" A color scale should appear which the user can use to improve the color coding, alter colors etc.

Now construct a molecular surface for the whole molecule or some interesting subset of it. Next select "Calculate: Simple Property Math: Map Atom Value to Surface: All Surfaces: All Atoms: Potentials". This will project the B values into the surface potential array of Grasp, which can then be colored and considered independently of the underlying atoms. (Note that if instead of "All Atoms" the user selects a subset which does not include all the atoms used in creating the surface some of the surface vertices will not be assigned a value, i.e. values are only transferred from atoms in "contact" with the surface.)

Calculating and displaying the effective dielectric from a single charge site.

The "effective" dielectric at a particular site is defined, for our purposes, as the ratio of the potential at that site due to a charge at a second site when there are

and left-most numbers are less than ten in absolute values. The color saturations should then be stronger.

Displaying surface potential and surface curvature side by side.

Having calculated the surface potential in the example before now calculate the surface curvature. Do this with the menu selection "Calculate: Surface Curvature (+Display)". This menu will ask you two questions, namely which surface and what atoms. Since there is only one surface this part is easy. As to what atoms should be chosen this depends upon the set of atoms used in the construction of the surface. As a good general rule use the same set of atoms in calculating curvature.

Upon completion of this calculation (note that this is NOT the fastest calculation in Grasp) the surface quantity should switch to curvature. This should be obvious since the default color scheme for curvature is green/grey not red/blue. Also the title of the color scale ought to change from "Potential" to "Curvature". Adjust the color coding to the desired intensities.

Next change to Split Screen mode with the menu selection "Display: Stereo/Split Screen: Dials to Both". There should now be two surfaces color coded by curvature. Move these apart a suitable distance if necessary (use the bottom right dial OR use the menu selections "Mouse Function: Z-Trans. Alternatives: Stereo split/twist"). Then use the menu selections "Display: Alter: Molecular Surface: Potential: (Surface Display Mode): Left" The last choice will direct the program to change the display of the left surface to be by potential. Note that two color scales should now be visible, one for potential linked to the left view, and one for curvature linked to the right view.

Surfacing two interacting parts of a molecule and select the interface.

Choose two parts of the molecule which are close to each other. Surface each individually, i.e. select "Build: Molecular Surface: Set A" then "Build: Molecular Surface: Set B: Add Surface" (the second surfacing will ask the user whether to overwrite the first, in this case the answer being no). Having built two surfaces we want to select, e.g. display, or color, only those parts of the two surfaces which are proximal to each other. To do this select "Calculate: Distance Map: Surface and Surface: A Constructed Surface: Surface 1: A Constructed Surface: Surface 2". This will calculate the minimum distances from each vertex of the first surface constructed to any vertex on the second surface. This calculation only results in values being calculated for the first surface and so the reverse calculation also has to be done, i.e. "Calculate: Distance Map: Surface and Surface: A Constructed Surface: Surface 2: A Constructed Surface: Surface 1" to find minimum distance values for the second surface. (Note that these calculations are the slowest in Grasp so be patient!).

Now that distance is a property for both surfaces one can remove all but the interfacial regions by giving the color command "vc=0, d=>x", where x is whatever the user desires. For instance if x=5.0 then only parts of the surfaces which are with five angstroms of the other surface will remain visible. (Note that the user can make distance the display property of the remaining portions of the

Worked Examples

In all the examples described below it is assumed a PDB file or other primary data file has been read in and is currently displayed.

Getting a molecular surface color coded by electrostatic potential:

First assign charges. When PDB files are read in the charge on each atom is set to zero, unless charge information and the correct file header were included in the file (see Appendix A). The default charge set for proteins is to be found in the file "full.crg". Read this in and assign charges with the menu sequence "Read : Radius/Charge File (+Assign) : Full.crg". The total charge assigned is reported in the textport. Note that histidines are uncharged. Atoms can be charged "by hand" by the "alt(q=x)" command (see "The Command Line: Altering Radii and Charges"). If a "partial" charge set is employed, i.e. where atoms other than ionizables are charged, then there is often a problem of incomplete charging due to inconsistent naming conventions of atoms. If a residue has a non-integer charge after a charge assignment from a file then this is reported to the textport and the file "charging_data" is written to the current directory with information on such residues. Remember that one can write pdb-"like" files with the charge and radii information for any subset of atoms.

Having charged the protein consider if the default values for an electrostatic calculation are as you want. These are changed by the menus "Set Parameters: Electrostatic Parameters" and the default values are to be found in the section on setting parameters with menus. Next proceed to launch the Poisson Boltzmann solver with the menu sequence "Calculate: A New Potential Map". This should take a few seconds to complete, upon which the user has filled map one with potential values.

Next construct a molecular surface by the menu sequence "Build: Molecular Surface". At the completion of this procedure a white molecular surface should appear in view. The surface will appear in the default display mode which is a fully lit surface unless otherwise set in the file ".init_grasp". Change the surface display mode from the menu produced by "Display: Alter: Molecular Surface: (Color Surface by): Surface Display Mode". (The ()'s indicating no choice made within this menu). Note that the atoms are still being displayed, although this may not be obvious if the surface is opaque. Since the display of the atoms will slow the draw speed the user might want to get rid of them by using the menu selection "Display: Hide: Atoms".

If the user now issues the menu commands "Calculate: Pot. via Map at Surfaces/Atoms" the program will calculate the potential values at all atoms and all surface vertices form map one. This should produce color (red and blue) on the white molecular surface, even if only faintly. Also the color scale should appear in the top part of the box. If it does not hit Control B. Use this to alter the color coding. The most useful action will probably be to place the cursor over the ">-<" part of the color bar and hold down the left most button until the right-most

clipping planes for the view. These clipping planes apply to all structures, including the display box. The tool appears as an independent window, i.e. it can be placed anywhere, resized at will and quit like any other Iris window. However the implementation is not perfect and at this time it is recommended that the tool is NOT resized and that the user be VERY careful not to quit the entire program when exiting a Z-Slice window (there are two options here, "close" and "quit", always choose "close").

The tool appears as having four colored squares surrounded by a white border. This border will change to red if the cursor is placed over the tool. Each square controls one aspect of the slice plane, namely either the width, the mid point, the front plane position or the back plane position. Each square has its function inscribed along with the current value. These values are in Angstroms. By clicking and releasing either the left or the middle mouse button in a square the value associated with that box is altered. Clicking to the left of the center of a box decreases the value, to the right increases it. The closer to the center a click the smaller the increment. The right most mouse button brings up the windows control menu. When the tool is first activated the clipping planes are changed +/- 1.0, i.e the front and back of the Grasp box. It is recommended that the user have this box displayed (Control O) while using the Z-Slice tool since this clarifies the clipping planes position (since it too is clipped).

The eighth menu entry will invert the surface normals of a surface selection, which as has been mentioned is a way to make a surface appear "inverted" or "inside out".

The ninth menu entry swops the pointer to the current internal map as detailed previously, and which can also be accessed via Control C.

The final menu entry resets the world rotations and translations, i.e. returns the view to that when the first structure was read into Grasp.

Help

I am the first to admit that the Help section in Grasp is not very helpful. It is divided into two sections, examples of Grasp syntax and Control keys. The former displays a menu whose entries are Grasp commands with a short description. The second lists each control key and its function. Selecting an entry in this second menu will cause that function to execute.

Quit

As in "It's Miller time".

This also ends this section describing the Grasp menuing system.

keys. The menu is,
Toggle cross hairs on and off
Toggle color bar on and off
Release/reconnect all dials
Fullscreen/normal toggle
Set and save colors
Alter depth shading
Z-Slice Tool
Invert Normals
Change Current Map
Reset World Rotations

Most entries are self-explanatory and so we shall describe only those which are not.

The sixth entry deals with depth shading. As previously described depth shading is a technique for adding apparent depth to a view by having parts of a view further away appear darker than parts closer. In theory one does not have to interpolate colors to black, but that is the automatic choice in Grasp. Colors are recalculated based upon a front distance, say "f" and a back distance, say "k" (f > k). If a vertex, for example, has Grasp Z-coordinate of "z" and a color of (r,g,b) its new color (r',g',b') is found from:

$r' = r * F$
$g' = g * F$
$b' = b * F$
where $F = (Z-k)/(f-k)$,
where $(f-k) = \max(0.1, f-k)$
and where ($F=1.0$ if $F> 1.0$) and ($F=0.0$ if $F<0.0$),

In Grasp f is taken from the front of the object and k is defined relative to it as a multiple of the distance from the front to the back of the object. The user actually enters a factor which is the inverse of this multiple. So if an object has a front to back distance of 0.8 box units and a depth factor of 0.5 the back of the object has its colors half interpolated to black. If the depth factor is 0.0 then the back distance is infinity, i.e. depth shading is turned off. A depth factor greater than unity will result in some of the object being colored completely black (Note that the element is not "uncolored".) As mentioned, the front and back of the object in question, whether a surface such as a contour or a molecule surface, or for bonds or atoms, is calculated dynamically, i.e. it changes as the molecule is moved. Note that the last condition specified above, that $(f-k)=\max(0.1, f-k)$ is included so that depth shading does not become too excessive for small molecules. In general the larger the molecule the more depth shading is needed, the smaller the molecule the less is needed. Another way of stating this is that the degree of depth shading should be proportional to the complexity of the object.

Upon selecting this menu entry the user get the option of entering a new value or turning the depth shading off for all structures or for any one of the following: atoms, bonds, molecular surfaces, and contours. Depth shading is *highly* recommended for surfaces drawn in a mesh representation.

The seventh entry invokes a special tool which allows the user to control the

The probe used in surfacing calculations for a molecular surface is also that used in the calculation of an accessible surface, i.e. defines how far out from the molecule such a surface is calculated.

In "Electrostatic Parameters" the user can alter those values which go into a Poisson Boltzmann calculation when the user launches a "New Potential Map" calculation. Those parameters are at present the internal and external relative dielectrics, the probe size used to estimate water inaccessibility, the probe size used to account for ion size, i.e. how close an ion may come to the surface of a molecule and the ionic or salt concentration. The default values for these parameters are:
interior dielectric: 2.0
exterior dielectric: 80.0
Water Probe Radius: 1.4
Ionic Radius: 2.0
Salt Concentration: 0.0

The final menu option "System Miscellaneous" has the following entries:
CPK level
Surface Area Probe Density
Bond Cylinder Diameter
Maximum Surface Resolution

The first allows the user to select the resolution used to draw the solid spheres used to represent atoms. The different levels represent hierarchical decimations of an octahedron.

The second entry controls the number of points on the probe sphere used to calculate surface area in Grasp implementation of Shrake and Rupley's algorithm (In short, points on a sphere of radius = Van der Waals of atom + radius of water are placed tested as to whether they lie within the similarly expanded radius of any other atom. The fraction of such points which do not is the fractional accessibility of that atom from which the accessible area is derived by multiplying by the area of that atoms expanded sphere). The point densities are based upon decimation of an icosahedron. On choosing a level the number of points in the test sphere is written to the textport.

The Bond Cylinder Diameter is the thickness of the cylindrical bond representation in Angstroms. The user can enter any value between 0.0 and 1.0. The default value for this parameter is 0.2 Angstroms.

The maximum surface resolution has been mentioned previously. It corresponds to the *minimum* lattice spacing allowed for the lattice used in creating surfaces. The smaller this number then the finer the grid spacing the higher the resolution allowed. This number is not allowed below one third of an Angstrom. Making this number larger forces a courser grid to be used which results in fewer triangles and hence surfaces which are easier to spatially manipulate in real time.

Miscellaneous

This menu is a "holding space" for functions which as yet do not quite fit anywhere else in the program. Some of the entries are also included as control

atom one of list one with atom one of list two etc. This r.ms. value is reported via the textport.

The user is then given the option of actually superimposing the two sets of atoms OR larger subsets of atoms based upon the same rotation and translation. For instance, suppose one is interested in the similarity of two helices with in a protein. First select equal number of residues within each helix. However, since the residues of each helix may not be exactly the same these selections can not be automatically compared or superimposed. So select just the backbone alpha carbons from each such set for the algorithm to work upon. An r.m.s is then calculated and reported for these two subsets.

The user then is presented with five options under a menu entitled "Apply rotation matrix?",

NO!
Apply Forward to 1st Selection
Apply Backward to 2nd Selection
Apply Forward to new Selection
Apply Backward to new Selection

The first option quits the subprogram. The second will rotate and translate the atoms of the first set onto the second via the optimal matrix calculated. The third option will do the reverse process of the second set onto the first. Obviously in the example above this is not a very useful operation in either direction, i.e. one wants to compare entire helices not just the alpha carbons. The fourth and fifth options address this limitation. By choosing a new selection the program will apply the calculated matrix to a new selection. So by choosing the whole of the first helix and the fourth option, or the second helix and the fifth will superimpose the complete helices.

Set Parameters

This set of menus gives the user access to many of the internal parameters of Grasp. The menu contains entries for parameters often altered and submenus for parameters which can be grouped together. It appears as,

Rotation Rate
Translation Rate
Probe Radii
Electrostatic Parameters
System Miscellaneous

The first two entries alter the rate at which the view is rotated and translated by the dials or the mouse. The user enters new values via the textport. To leave the value unchanged hit return without typing anything. Rotation rates are in angles per redraw, while translation rates are in Angstroms per redraw. The default values in Grasp are set high to allow for rapid initial manipulations, after which the values can be set smaller for fine control.

The third entry controls a submenu which accesses the radii of different probes used in Grasp, namely that used in calculating the surface area of atoms, that used in making a molecular surface, and finally that used in a Poisson Boltzmann calculation (the latter can also be set in the Electrostatics submenu).

view. To do this merely set the dials to the right-hand view and attach the dials to the desired formal subset. To switch to the same subset on the left one has to go through the "Display: Alter Stereo" menu options again (Note that being in the "Active" selection mode does not help here, i.e. the side which is manipulated is NOT decided by which side the cursor initially resides upon). Furthermore, not all features are fully implemented, for instance the "undo" features of the "Remove Subset Rotations" only apply to the left-hand subset. This limitation also applies to the final menu option "Fix Subset Rotations". It is unlikely this will present many problems to most users. The matter will be more completely resolved by the next release, especially if the last comment proves incorrect.

By "Fixing Subset Rotations" is meant that the subsets own transformation matrix, i.e. that matrix which represents the differential manipulations relative to the rotations and translations applied to the world view, is applied to the subsets internal coordinates. Its values are then reset to the identity matrix. Remember that moving an object via mouse or dials does not affect an objects internal coordinates, just the view, whereas this option allows the user to "fix" the apparent rotation, i.e. make the change of view apparent in the objects coordinates, and reset the view. This application of a transformation has been described earlier in writing a set of coordinates to an external file. (Note if the user chooses to "fix" the "world" then all coordinates are so recalculated, while the world transformation is set to its initial value, i.e. the identity matrix.).

Finally, one can "Delete" a formal subset. When this is done the atoms and or vertices formally associated with that subset are returned to the "world", i.e. its own transformation matrix is wiped clean, and that formal subset name is removed from the name list. Note that this may cause the subset to "jump" if this subset has been moved relative to the world. Note that if a surface subset has been made which does not consist of a complete constructed surface, for instance the user scribed a patch of a complete subset and made this a subset, then deleting a subset does not return the internal representation completely to its original state. This is because forming such a subset will cause the duplication of some vertices, specifically those at the border of the subset where vertices are originally shared with the rest of the surface. This *may* cause problems if the user later attempts to scribe across this region into another, or in any other Grasp function where vertex connectivity is important.

Programs

This menu entry is intended to allow the user to access auxiliary programs via Grasp. At present the only such program is "Superimpose", which uses the Kabsch algorithm to calculate the r.m.s between sets of atoms. (Actually this program has been subsumed in Grasp so is not really auxiliary!)

Superimpose requires two set of atoms as its input. The user can supply these in the usual manner, i.e. formal subset name, command line selection, complete molecule, etc. These two sets MUST have the same number of atoms. Once given, the algorithm calculates the rotation and translation which will minimize the root mean square difference between these two sets, the differences being calculated between each atom in each list sequentially, i.e.

surface two with all atoms within five angstroms would be "sm2:a".

This somewhat complicated naming system was designed to enable the user to keep track of somewhat complicated subsetting operations. If the user is not going to require such sophistication choosing an arbitrary name may be preferable. Note also that if the subset created contains elements of a previously created formal subset then those elements are reassigned to the *new* subset.

Once a formal subset is named, the dials, i.e. rotations and translations, are automatically assigned to that subset to the exclusion of all other atoms and/or vertices. If the subset just chosen was itself a subset of a formal subset which had been moved relative to the original structures, then the new subset will inherit the previous manipulations of this subset. In technical terms, it inherits its parents transformation matrix.

To change this "focus" there exist the first two menu options allow the user to attach dials either to the "world", or to any formal subset. By "world" here is meant the entire view, i.e. all atoms and vertices. If the user opts to connect the dials to another formal subset the user is presented with a list of such subsets so far created. If the "Active Formal Subset Rotation" mode is in effect the dials are still automatically attached to any new subset, however this is no longer significant, unless the user quits such a mode immediately, since manipulations are only applied to what the cursor first lies upon. Note that choosing a formal subset or the world for the dials turns off the "Active" mode.

One of the intentions of formal subsetting was to allow the user to rearrange the positions of two molecules, or parts of molecules. For instance to explore new configurations of an enzyme and an inhibitor. The menu options for removing subset rotations and for fixing subset rotations are designed to help this process. The former allow selective "undo" operations on rotations and translations applied to subsets. After selecting a subset the user has three choices, namely,

That Subset whole rotation
Since world last moved
Since another Subset moved

The first option will remove all the "differential" manipulations on the chosen subset, i.e. all those manipulations different to those performed on the world view. Hence the subset is moved to the position it would have had if it had never been selected as a formal subset. The second and third options remove *partial* rotations (and translations). The second removes all such operations applied to the subset since the world was last moved, the third all such since some other formal subset was moved. In terms of transformations the world has one transformation matrix associated with it, each formal subset has an *additional* unique such matrix. The total transformation matrix for a subset is then the product of these matrices. The first menu option resets the subsets own matrix to the identity matrix, where the second and third option resets that matrix to previously saved values, e.g. that matrix when the world was last moved.

If the user is in stereo, or split screen, mode subset manipulations can become very complicated. Grasp allows the user to manipulate the left and right views independently. This facility carries over to formal subsets, i.e. one can manipulate the right-hand view of a formal subset independently of the left-hand

hundred such formal subsets may be created.

Formal subsets are usually, but not exclusively, formed to enable independent rotations and translations, the other possible use is the ease of having a unique label for a certain set of atoms or vertices or both. Hence the functions below deal with either the naming or un-naming of formal subsets and their spatial manipulation. The possible menu options are:

Fix Dials to World
Fix Dials to a Subset
Make a Formal Subset
Remove a Formal Subset
Remove a Subset Rotation
Fix a Subset Rotation

We will consider first creating a formal subset.

If the user desires a subset of atoms, or surface vertices Grasp will prompt the user with the standard options in selecting a subset, e.g. a command line selection, a particular molecule, a scribed surface etc. These atoms or vertices then become the new formal subset. If the user want to make a mixed subset then the criteria used in associating surface with atoms or atoms with surface is distance. This can be distance in terms of atom center to vertex distance, atom center to vertex distance minus the Van der Waals radius of that atom, or "contact", which means that the surface and atom are related in that that atom was used in the creation of that bit of surface. Note that as with all distance calculations the user may have to wait momentarily while the calculation proceeds.

The names of formal subsets are derived from the constructed surface number of the surface, or the molecule number of the atoms, and any previously named subsets. For instance, if the user has selected atoms from the second molecule, and it is the first such subset, then the name suggested would be "m2:a", "m" for molecule, "2" for the number, "a" for the first such. If it was the second such subset from molecule two and at least one atom in this selection is not contained within the first subset "m2:a", then the name suggested is "m2:b". If it is totally contained within the subset "m2:a" then the name will be "m2:a:a", i.e. the naming become hierarchical. If the atoms make up exactly all of one molecule then everything from the colon to the right is dropped, i.e. molecule two as a subset is simply "m2". (Note that molecules are not automatically formal subsets, they have to be made so). If the atoms are from more than one molecule then the "m" number is set to zero, i.e. the first such subset would be "m0:a", the second, not included in m0:a, would be "m0:b", etc. The same rules apply to surfaces, where the construction number substitutes for molecule number. For mixed subsets the rules for atoms with associated surface are applied as if there where only atoms, except the "m" becomes an "ms", e.g. "ms1:a" for the first set of atoms from molecule number one with associated surface. For mixed subsets where the surface is the primary selection, and atoms associated, the "s" changes to "sm", e.g. the set of some vertices from

duplicate those atoms.

Grasp will write out either internal map in DelPhi form. This file will contain the values at each grid point, along with the grid spacing and coordinates of the central point. Such files can not be concatenated. A default file name of "fort.14" is provided because that is the default Fortran unit to which potential maps are written by DelPhi.

Grasp property files have been discussed above in the context of reading such. The user is given the choice of surface or atom property, and then the particular property to be written. (If the user wishes to add comment lines to this file this must at present be done "by hand", i.e. by editing the file externally). One should note that the user does not get to choose which vertices and which atoms have their properties exported as *all* such are exported in Grasp property files. The reasoning behind this is that since the files do not contain any atom or vertex information other than the sequential position in the file, so selecting a subset of, for instance, atoms, would not necessarily be sequential. Writing a property file for such a selection would give a file almost impossible to interpret, and which certainly could not be read back in. If a subset of properties is required the user should export a property in a PDB file , or list them to the textport using the "list" command and capture them using the "cut and paste" feature of the window manager.

The "History File" option will write all the command processed by the program via the textport to the file "history.dat". Note that only *commands* will be written, not selections passed to the program via the textport in the context of a menu request. The user can then edit this file as seen fit, adding comment lines by making the first character of that line an exclamation mark "!".

The last write option will write a list of *accessible* surface points which belong to cavity surfaces (assuming these have been calculated) as if each such point was an atom, i.e. in PDB file format. The file is always named "cav.pdb". One use of this file has been to add these pseudo-atoms to a pdb file and prevent this volume from being assumed high dielectric by DelPhi. (Note this is not assured of achieving this goal if the cavity could accommodate a sphere of diameter twice that of a water, but this rarely *ever* occurs).

Formal Subset Operations

Formal subsets may be a selection of atoms, or surface vertices, or both. In the latter case we refer to the subset as being of mixed type. Mixed type subsets come in two flavors, those were the atoms are selected on the basis of proximity to a set of surface vertices, and secondly those where the surface vertices were selected on the basis of proximity to a set of atoms. Upon construction each subset is given a unique name, the process by which this name is created being described below. Alternatively the user may provide their own name. Up to one

different molecules would from a properly constructed multiple molecule PDB file.

When writing a surface file the user is given the same choice of surface subsetting as in any other operation involving surfaces, i.e. one is not restricted to exporting whole surfaces. For instance the user may wish to save only the part of a surface involved in making an interface with another surface. Note that the user can immediately read a surface just written, which provides a mechanism to duplicate a surface subset.

For both surface and atom files the user has four choices for the frame of reference used to calculate the coordinates written to file. These appear in a menu entitled "Coordinate Type", which looks like,

Absolute Centering
..and Rotated
Box Centered
..and Rotated

The first menu option, which is the default, is to write out the coordinates unchanged, the other three change the coordinates from their original values. These three arise from the combination of two effects, firstly the centering Grasp does when it first reads in a primary data structure, and secondly the rotations and translations performed within the program.

As an example of the first effect, a molecule has center, i.e. average position, (4.203, 3.123, 5.011). If this molecule is the first file grasp reads in it will both set the scale of the Grasp box by it, and also make the center of that box at this exact same point. If the third option is chosen then 4.203 is subtracted from each X coordinate, 3.123 from each Y coordinate, and 5.011 from each Z coordinate, i.e. the average position of the molecule written out becomes (0.0, 0.0, 0.0).

The second effect is that the user can apply any translation and/or rotation performed to the written coordinates, i.e. if the molecule has been rotated by 90 degrees about the X axis then by choosing the second menu option the molecule coordinate is written out as if the molecule had been so rotated.

Combining these two effects as in the fourth menu entry causes the coordinates to be calculated *first* with any rotation and translation applied, and then the Grasp box center subtracted from the coordinates.

These various option exist in case the user wants to compare surfaces or molecules, or parts thereof, which have radically different coordinates and/or orientations. The option to remove the center from the coordinates can superimpose the centers of different objects, the application of rotations and translations can then give them similar orientations.

Atom files are PDB files but with the four different possible formats mentioned in the section on reading a PDB file, i.e. no extra data, radius and charge data in the occupancy fields, another pair of properties in these fields, or a pair of properties to higher precision. In the latter two cases the user is prompted as to which atom properties to export. As with surface files the user may select any subset of atoms to be exported rather than all atoms, and the user has the choice of four different coordinate frames, as described above. As with surface files the user can export a molecule or part of a molecule and then read it in as a way to

are located in the Grasp data directory, and the user may place their own there too if they wish them to appear explicitly in this menu. These files should have the correct file extension, i.e. size file are ".siz", charge files ".crg". Alternatively, the user can input the name of such a file, which will be processed according to the file extension. The correct format of these files is described in Appendix A. Upon reading a charge file the total assigned charge will be reported, and the maximum and minimum values found, which may alter the display if charge is a current display variable. Upon reading a radius file the program will report the number of distinct radii amongst all atoms. This value may not exceed one hundred.

Files containing information on DNA parameters at present must be in the same format as that put out by the program "Curves", (which must have extension ".lis"). Future versions of Grasp will support other DNA structure programs output.

Pair wise interaction energy files are ascii files which can come in two formats as described in Appendix A. The files contain information on which two residues each interaction is between and the strength of this interaction. If, based upon the supplied information the program can not assign the given value to a pair, i.e. it can not find one or both of the described residues, that line of the file is written to a file called "int.pair.not.assigned.dat". That there have been such failures is written to the textport after the file has been read in. There can be at most 10,000 such interactions in the program at any one time. Note that reading in a new file will, at present, overwrite old values.

Write

Grasp can export most of its internal data structures to external files. Some of these files will have been referred to in the previous section on the input of data. The format of these files are to be found in Appendix A. Files are written to the current directory (unless the file name specifies otherwise). If the file exists the user will be prompted as to whether to overwrite the file or not. In some case the user may have the option to append to a file of the same name.

The menu options are:

Grasp Surface File
Atom (PDB) File
Potential Map
GRASP Property File
History File
Cavity Surface Points (PDB)

Surface files contain information sufficient to display and manipulate a surface. Any property array which differs from zero in any of its entries is also written automatically to the same file. One can append surface files to other surface files when writing. When this is done the equivalent of a "TER" line is inserted so that upon subsequent input different surfaces remain distinct, just as

calculation. Grasp supports this type of file, i.e. will recognize the format when the file is read and assign the values in these fields to the radius and charge of each atom. Grasp also supports files where this information is actually meant for general atom properties one and two (for instance if the values really do represent occupancy and B values), and where these properties are written in higher precision. To alert the program to the correct interpretation of these fields the user should insert the correct headers to these files, as described in Appendix A. All these file types are written by Grasp, in which case the correct header is automatically inserted.

Upon reading a pdb file Grasp automatically constructs a bonding pattern for those atoms based upon proximity and atom type. This procedure is not always perfect but works well for high resolution structures. The exception to this is if the file contains alternate atoms, i.e. if some atoms are multiply represented in the file due to ambiguities in the original electron density map. The program will not at present take account of this in bond formation, i.e. inappropriate bonds will probably be made due to proximity considerations. Multiple structures can be entered within one file if they are separated by "TER" statements, i.e. by lines which begin with those three capitalized letters. Otherwise the same problem of inappropriate bonding patterns may occur. If Grasp detects what it thinks is such a pattern, e.g. a carbon bonded to five atoms, it writes out that atom's information field, plus the number of bonds being made, to a file called "bonding.dat". This can help the user check the validity of the assumed bonding by viewing this file after a structure has been read. Grasp will also write the number of atoms and the number of molecules read to the textport.

Radii are assigned from the data file "default.siz" unless the file type had embedded radius information. Charges are not assigned automatically, unless they are also embedded in the file. If charges are assigned within the file the maximum and minimum charge values are (re)calculated for use in color coding of this variable.

If no scale has been calculated then Grasp calculates the maximum X, Y or Z dimension and makes the Grasp box size 50% greater, i.e. the molecule fills at most two thirds of the box. (Note that Grasp includes the atom radii in this assessment so that small molecules do not appear too large).

Grasp script files are ascii files such that each line is sent to the command interpreter which deals with commands normally sent via the textport during program execution. As the program reads the file each line is written to the textport at the same time it is processed. Lines which begin with the symbol "!" are treated as comment lines and written to the screen but not acted upon. An example of the use of such a file is to enact a default color scheme for atoms the user favors, or as an alternative method of setting radii or charges from DelPhi control files. (Note that the user has the option of writing all commands entered, via the "Write" submenu, to an external file "history.dat" which is often a good starting point in producing a script file.)

The next menu entry, that for "Radius" and "Charge" files, deals with the input and processing of DelPhi style control files for these variables. Several such files

such operations as contouring. If no scale and midpoint has been set for the Grasp unit box it is set from the scale and midpoint of the phi map, such that the map exactly fits the Grasp box.

DelPhi *eps* maps (default extension .eps) are the final unformatted file type Grasp will read and contain the information on the dielectric map used in a DelPhi calculation (*eps* is short for epsilon, the symbol for the dielectric value at each point in space). The dielectric map is defined at the grid mid-points, i.e. the points half way between every grid point and its six nearest neighbors. There are 3*65**3 of these. Eps maps have a center and scale associated with them, though they can not be used as primary data files for Grasp. They also contain a list of which grid points were assigned as being in salt, which is referred to as the "Debye" map. When reading in an eps file the user can choose to read in just the X or Y or Z component of the eps map, since these are each individually 65**3 maps. The grid values are either set to one or zero, one if inside the low dielectric, zero if outside. The user can also opt to make a composite map of these three components, i.e. where the values of the eps map at each of the six midpoints associated with a grid point are added together and assigned to that grid point. Finally the user can select to read only the Debye map from the epsmap. Here the values assigned to each grid point are one if in salt and zero otherwise.

These maps can be used to check the actual surface used in a calculation, relative to the underlying atoms. They can also be useful in calculations. For instance, one can use the eps map to set all potentials inside the molecule to zero, or the debye map can be used to calculate excess salt concentration from a DelPhi phi map.

A Grasp Property file is an ascii file which contains a list of values to be assigned to either atoms or vertices. The format of the file is especially simple, namely a number of comment lines, then a line with the letters "surface=property" if the values belong on a surface, or "atoms=property", if they belong to atoms, where "property" can be "potential", "curvature" and so on, followed by a list, one value per line, of the values to be assigned to vertices or atoms in sequential order, i.e. value one goes to atom one or vertex one. When such a file is read in the maximum and minimum values for that property array are recalculated, the color coding altered if necessary if currently displayed. These files can be created by Grasp, or by the user via external programs.

Atom coordinate files, in the form of Protein Data Bank files, are the final primary data file for Grasp. The correct extension is ".pdb", and a list of all such files in the current directory is maintained in menu form. Grasp supports several variations on the format laid down by the Data Bank, in that the field after the coordinates can be used to contain Grasp atomic property information. The original such variant was the "modified" PDB file output by DelPhi. This substituted the occupancy and B-factor information with radius and charge for that atom, since this is information was necessary and sufficient for a DelPhi

latter being the option common to all. When a file name is entered Grasp it always checks first whether that file exists in the current directory. If it does not then it may, for instance in the case of PDB files, then proceeds to check all the directories in the users "$PATH". (This Unix environment variable is used whenever a command is entered on the Unix command line to determine the order of directories to search for that executable.) To set, alter, or check this variable, see the notes on the data directory variable "$GRASP" in the section "Basics". (Note Grasp only checks the "$PATH" variable at startup). The user can use this functionality, for instance, to point to a directory containing set of pdb files held in common for users of their system. If no file is found upon this path the fact will be reported to the user and the "Read" aborted.

Grasp Surface Files contain the information necessary to reconstruct a surface. It is an unformatted, or binary, file, i.e. can not be read as an ascii file. It contains the surface scale and midpoint, plus a list of vertex coordinates, vertex normals, accessible surface points and a list of triangles, i.e. vertex numbers of each triangle. Also, any surface variable which has been calculated or assigned is also written to this file, i.e. surface potential, distance, curvature, general properties one or two. Upon reading such a file the surface is automatically displayed in the default surface mode. The surface display quantity is usually set for potentials. Since there are memory limits on the number of vertices and triangles current in the program at any one time the program checks as to whether there is space for the new surface and the input is halted if there is not.

The program will write to the textport the number of vertices and triangles read, as well as which properties were also included in the file. As described below surface files may contain more than one distinct set of surface points, in analogy to a PDB file containing more than one molecule. Each surface read in is assigned a sequential integer, as if each had just been constructed anew, i.e. if the user has built two surfaces and two more are read in they are assigned "Constructed Surface Number" three and four. These numbers can be used in surface subsetting as described in the section on command line syntax for surfaces. These numbers also appear in menu lists of constructed surfaces.

Grasp data files should have the extension ".srf", for surface representation file. They are one of the three primary data files for Grasp, by which is meant that the program can set the view center and scale, and can display the structure upon input.

DelPhi potential maps, or *phi* maps, can be produced by DelPhi, or by Grasp. They are also unformatted files and one of the three primary data files for Grasp and should have the extension ".phi". The program keeps a menu list of all such files in the current directory, which is accessible to the user when choosing a file. As well as containing the potential at every point of a 65 cubed lattice, phi maps contain a scale, i.e. a lattice spacing, and a mid-point, i.e. the absolute coordinates of grid point (33,33,33). Extended phi maps may also contain a rotation matrix. A phi map is read into internal map two, overwriting any resident map. This map is set to be the current map, i.e the one automatically used for

clicking on the same place (=triangle) twice causes the "greening" to begin, so be careful not to accidentally start this process by starting at the same site twice in a row. Remember that other than a well formed blue border only a further mouse click will halt the expansion of the green triangles.

The fifth entry in the "Mouse Function" menu is "Command Line Mouse". What this means is that the user can fix a certain written command to be automatically enacted when an atom is "picked". These commands can only refer to atoms at present. For instance, if the user enters,

c=2

after selecting this item, then every atom picked after that will be colored red. This procedure works by finding the atom number of the atom picked and then appending this to the entered command, and sending it to the command interpreter. So that if the user clicks on atom 55 then the command actually sends the command,

c=2, an=55

to the command interpreter. (Note, then, that this format will not work with the projection syntax, i.e. c=2 >r,an=55 will not color the residue containing atom 55 red.) One use of this function is to assign charges to a picked atom, or to remove certain atoms, e.g. by using "c=0" or "alt(r=0)" from view. Rechoosing this menu item turns off the automatic command function.

The final item on the "Mouse Function" menu is that which affixes alternate functions to the Z displacement mouse function, i.e. holding the middle mouse button down while moving the mouse up or down. There are only two alternate such functions, one is to alter the position of the projection plane, i.e. that plane which is colored coded by the potential from the current map. The second is to alter the separation of a stereo pair. This latter function is normally taken care of via the dialbox, but it included since not all users will have such. Also in the latter case the stereo twist is fixed to the left right motion of the mouse. One also uses this menu to return the button to its usual functions. When Grasp has more reliable graphical tools the functions here will instead be assigned to such.

Read

This menu item deals with the input of data to Grasp, and the operations performed on it as a consequence. The formats for many of these files will be found in the Appendix A. The menu entries are,

GRASP Surface File
DelPhi Potential Map
DelPhi Epsmap
Grasp Property File
PDB File
Grasp Script File
Radius/ Charge File
Curves File
Pair Wise Interaction File

Upon choosing one of these options the user will up to get three choices, namely a default file name, a menu list, or the option to enter a file name, the

triangles. Note that to move the surface while in scribing mode one has to either use a dialbox, or to position the mouse off of any surface before attempting to move the view by mouse.

When the user has outlined an area of interest the inside of this area can be selected by double clicking on it. For instance once the user has turned the outer collar of an active site bright blue, double clicking on the surface of the site will cause the initial triangle under the cursor to turn first blue, then green, then the triangles adjacent to the green triangle are turned green, and then those adjacent to these turn green etc. Thus a green "wave" expands outward from this initial selection. However a "blue" triangle, i.e. one previously selected by the user, will not turn green even if a neighboring triangle is so. Hence if the blue triangles form a complete circle about the initially chosen triangle the green expansion will halt inside this blue "border". Alternatively, if the user clicks again while the wave is still expanding, with either the left or middle mouse button, it will immediately stop "growing".

Either process results in a surface selection, i.e. all those triangles colored green. This selection is referred to as the "Currently Scribed Surface" in all menus which prompt the user for a surface selection. For instance, the "Calculate:Surface Area:Molecular Surface" menu path will give the user a menu with this option, which will result in the calculation of the surface area of the selection. The user can make this selection permanent by making it a "formal" subset. It will then be assigned a name which can be used to refer to it at any future time. This is described in more detail later in this section.

There are further options available if the user selects the same menu entry as before, i.e. "Mouse Functions:Scribing". If scribing is already active a submenu appears as below,

Undo Fill
Add Border to Fill
Clear All Marks
Turn Scribing Off

The last of these options is self explanatory, and also results in the removal of all green and blue markings from the surface. The first entry will remove all green markings, i.e. the "fill", the second will change any blue triangles in contact with green triangles green, and the third will remove all markings without exiting scribing mode.

One should exercise a little care while scribing for two reasons. First, if the cursor starts on the surface when a button is depressed but ends off the surface before it is released (which should not really happen when scribing) one can get unexpected results (e.g. the mouse is "locked" into a rotational mode). Secondly if the surface is at a course scale, e.g. a large molecule has been surfaced, then one should not scribe too fast because occasionally the program may get confused as to where it is on such a surface. Furthermore if one is trying to trace a boundary on a complicated surface one may have to rotate the molecule to get over some surface features with the border intact. Note that scribing does not work if the surface is rotated underneath the cursor, i.e. rotating by dial while holding a mouse button down with the cursor on the surface. Remember that

There are three different ways information can be sent from the mouse to the program, namely holding a button down and releasing it without moving the mouse (referred to as "picking"), which usually returns information on the underlying object, holding a button, or buttons, down and moving the mouse, which typically causes rotations or translations, and moving the mouse without any buttons depressed, which usually in Grasp has no effect. These methods can be affected with the menu options within this menu.

The menu appears as,

Atom Information
Surface Information
Measure
Scribing
Command Line Mouse
Z-Trans. Alternatives

The first two options allow the user to choose whether the left or the middle button is associated with returning surface information or atom information when either an atom or vertex is "picked". The default arrangement is that the left button gives atom information, the middle button returns surface information. These entries are also useful in resetting functionality altered by other menu options.

For instance the third option allows the user to associate certain measurement functions with the picking mode of either left or right buttons. The different measures are distance, angle and torsion angle for atoms. These measurements require two, three and four atoms respectively as data. As an example, suppose the user links "distance" to the left most button. The user then "picks" an atom. The next atom picked by this button will cause the distance to be calculated between the first atom and this atom, and for the result to be written to the textport. The next atom chosen after that will cause the distance between this and the second atom to be calculated and presented. This "chaining" of distance calculations continues until the user either clicks on the last atom again (i.e. double clicks on it) or "clicks away" i.e. clicks on empty screen. This resets the "memory" of atoms i.e. the next atom picked does not cause a distance calculation but acts as the first atom chosen for a new chain of distances. The same procedure is followed for angles and for torsion angles except that there are the angle calculation occurs for the last three atoms picked, and torsions for the last four. Clearing the "memory" is done as before. Note that atom information is still written to the screen with each individual pick. One can turn off the measurement function for the chosen button by reassigning it to atom or surface information via the first two menu options.

The fourth option concerns the "scribing" function in Grasp. This is a way to select parts of a surface by drawing the outline of an area directly onto the surface. Selecting this option turns the mode on and disables other functions of either button while the cursor lies over a molecular or accessible surface. When either the left or the middle button is held down while the cursor is over such a surface, and the mouse moved, the triangles under the cursor that make up that surface should "respond" by changing color to bright blue. Upon releasing the cursor the "track" of the cursor should remain visible as a chain of bright blue

radius zero. This probe radius may be set by the user via the menus "Set Parameters: Probe Radii", the default being 1.4 angstroms.

When a subset of atoms is used the area calculated is as if other atoms do not exist, i.e. as with curvature calculations other atoms are not used to determine accessibility. All results from the calculation are placed in the accessible area array, which can be manipulated via the simple math options. Hence if one wants to know the total accessible area of all lysines in a protein one does not want to select just lysines at this step, rather choose all atoms, then go through "simple math" to calculate the sum of the accessible areas of that subset of atoms. On the other hand one might want to know the total accessible area of a subunit of a quaternary molecule if it were isolated from the total structure. Then it would be appropriate to select just this set of atoms.

Distance maps are actually *arrays*, not maps in the Grasp sense. They can be calculated for any subset of atoms or vertices to any other set of atoms or vertices. They represent the minimum distance for a "from" set to a "to" set. Since atoms have radii it is often useful to take account of this in the "distance" quantity, i.e. subtract it so that the distance calculated is the distance to or from the Van der Waals surface of the atom. Note that this can result in *negative* distances.

The results of the calculation enter the atom or surface array for distance *only* for the initial (i.e. the "from") subset selected, not for both. For instance if one wishes to calculate the distance map between two surfaces one should choose this function for the first surface against the second and then repeat it for the second surface against the first. Note the user should be careful that the second set does not contain some of the same points as the first set or the minimum distance will clearly be zero for these points (or negative if Van der Waals radii are being subtracted).

Distance calculations are quite slow if many points are being checked against many points. At this stage of the program these calculations have not been optimized. A rough estimate of the time taken for a calculation to complete is to multiply the number of points in each set (i.e. the "to" and "from" sets), divide by one million and multiply by two. This is then an estimate in seconds of CPU time. If one is calculating for a 10,000 vertex surface to a 15,000 vertex surface, sit back and wait!

(Note: if one wishes to remove the Van der Waals radius from the "to" set, i.e. the set of atoms not having distance calculated for, then strictly each distance comparison should involve a square root. Because this is very slow on the Iris, and because this is already a slow calculation, Grasp actually calculates the minimum center to center distance *first* and then subtracts the Van der Waals radius of the atom found closest. This is not strictly correct and may result in inaccuracy at small distances (i.e. of the order of a few radii). This limitation will be removed when more efficient distance algorithms are installed.)

Mouse Functions

magnitudes and directions are calculated from the current potential map. Each result is displayed as a field arrow similar to that for the molecular dipole, originating from each chosen point.

The user is given the option of giving each field vector either a constant length, which is then entered, in angstroms, by the user, or a variable length, where the user enters both the maximum vector length in angstroms and the maximum field strength to which this corresponds. In the latter case field vector of strength less than the maximum representable field strength are assigned a proportionally smaller vector and those greater than the maximum the maximum length. Note that these lengths can not be altered once set and neither can the color for which the user is prompted for each set of vectors. As with field lines, one can delete current vectors when making new ones or one can add them, with the same maximum of 100 total vectors.

One should note that as with field lines near charges the field vectors may be less than reliable. One should bear in mind the scale of the grid used in calculating the potential map. Fields are calculated by first estimating the fields at the eight corners of the grid cube that encloses the chosen point and then using trilinear interpolation on that set of vectors. The relative error of the field when a point is near a charge is greater than that of the potential because it involves quantities closer to the charge (i.e. higher order derivatives). For this reason one should be very careful in interpreting field *strengths* quantitatively from any such interpolation when near atomic charges.

The volume enclosed by a surface is relatively simple to calculate given a tessellation of the surface. One merely chooses a point, preferably close to the surface or molecule, and then calculates the volume of each pyramid formed by this point and the three points of each triangle in the tessellation, being careful to have the order of the points in the triangle base be consistent with the surface normal. One can calculate the volume of any closed surface, such as a contour, a cavity, a molecular surface or an accessible surface. The volume of a set of atoms may also be calculated this way, i.e. be tessellating the Van der Waals surface of that set of atoms, calculating the volume of that closed surface. Alternatively one can map the atoms onto a fine grid, find which grid points are outside and which inside, and form a lattice approximation that way. Since both are simple Grasp does both and answers are written to the textport.

Areas of surfaces are also simple given a tessellation, since it is just the sum of the areas of all triangles. The area may be of any subset of surface one can categorize. The area of a set of atoms is calculated using the surface area algorithm of Grasp which is an efficient version of "Shrake and Rupley", i.e. placing points on a test sphere about each atom and finding which are inside and which out. The density of points, and hence the accuracy used in the calculation can be altered by the user via the "Set Parameters: System Miscellaneous :Surface Area Probe Density" menu entry. The user also has the choice of whether to calculate the accessible area, i.e. as if the radius of each atom is increased by the diameter of a probe, or the Van der Waals surface, i.e. probe

selected by surface scribing. Once chosen then a number of points equal to the number of lines chosen are found at random over the selected surface as seed points.

Choosing "Atom Set" will cause the center of each atom selected to be initial seed points. If more than one field line is selected than that number of seeds will be formed within 0.5 angstroms of each atom center. The user should note that there is a maximum of one hundred field lines and should be careful not to exceed this limit or the process will not run to completion.

Choosing "Origin" causes the initial seed point to be at the center of the Grasp box. This number of seed points can be multiplied in the same random manner as described above.

Finally one can enter the absolute coordinates for the initial seed via the last menu entry. Again this can be multiplied by random scatter about this point.

The next menu deals with field line directionality. Clearly one has a choice of whether to use a positive or a negative "test" charge when plotting the field line. Traditional electrostatic lines begin on positive charges and end on negative charges (or infinity). In Grasp they can go either way depending on the sign of the test charge, or go both ways! Note that by choosing bidirectional lines one actually creates *two* field lines, one in each direction. If one has chosen to create the field line seeds from a surface then one has yet another choice, i.e. whether one would like to force the direction of the field line into or away from the molecular surface.

Next one can choose the number, or multiplicity, of field lines. As mentioned above this will multiply the initial seed positions by finding random positions in the vicinity of the initial seed point(s). The default value is one. As mentioned above, bidirectional lines count as two. Thus if one chooses the atom centers of 20 atoms, with a multiplicity of 2 and choose bidirectional lines one actually gets 80 field lines.

Finally the user gets a choice of colors for the field lines, the default being white. If one has previously created some field lines then one has the choice of adding to those already created, or replacing them (Note the 100 field line limit applies to the total number of field lines). Field lines can be drawn as lines or as cylinders. The latter display is more dramatic than the former but is very slow to draw.

Currently in Grasp the line segment length of the field line is constant. This means that sometimes the lines will exit the potential map i.e. for which there is no interpolated field. Lines will then terminate. Another problem is that around charges field lines will not actually terminate, rather they oscillate. This is due to the nature of the field around a charge mapped onto a grid. There is also no method of illustrating the field strength at any point along a field line. These shortcomings will be addressed in future releases.

Field vectors can be calculated at up to 100 sites. Sites of interest are chosen as either a set of atoms, i.e. each atom center, the center of a set of atoms, or at a random selection of points from a surface, or from a point whose coordinates are entered by the user. Once the points of interest are selected the

the property array for that vertex. Most typically the user will apply an atomic property universally, i.e. choose all atoms and all vertices, to the surface.

The flexibility of operations on atoms and vertices greatly exceeds that of maps, which is mainly because the uses so far imagined for the former greatly exceed those for the latter. In time it is hoped that Grasp will support a math interpreter rather than use the menu system which should more fully realize the potential for on the fly analysis it is hoped this section makes possible.

"Dipole moment" takes all the charges on all atoms and calculates a dipole and, if necessary, monopole. The latter is necessary if the total charge is not zero. The procedure is to find the sum and charge weighted average position of all positive charges and the similar quantities for the negative charges. If the two sums are not the same then there is a monopole equal to the sum of the sums acting at the charge weighted average position of the larger (in absolute terms) of the sums, plus a dipole of magnitude determined by the smaller (in absolute terms) sum multiplied by the distance between charge weighted centers. These values are printed to the screen, plus once calculated the user can display a dipole "arrow" centered at the average of the charge weighted centers, and of length 0.3 box units. This is displayed via the "Display:Show:Vectors" menus. There is as yet no options to display or calculate multiple dipole vectors, or to alter the default vector length.

Field lines are a difficult quantity to visualize. This is mainly because they are infinite in number and it is difficult to visualize an infinite number of lines. In a "correct" implementation the number of lines within a given volume is proportional to the average field strength. Hence one could imagine discretizing this concept so that there are a finite number of such lines. However this is NOT the approach followed in Grasp. Instead one chooses a set of starting points, or *seed* points, and projects the direction a positive or negative charge would take from each such point, i.e. the field direction, as found from the *current* potential map. A line segment is drawn a certain distance in this direction and then the field at the end of this segment calculated. This process is repeated up to a fixed number of times to give a field "line". Typically one hundred segments are calculated. If the length of each segment is small enough the line appears smooth.

Seed points are determined by the first submenu of this selection. The choices are,

Center of Atom Set
Surface
Atom Set
Origin
Enter Coordinate

The first choice will prompt the user for a set of atoms. The average of these coordinates will then be found. If the number of lines chosen in a later menu is greater than one then that number of random positions within half an angstrom of this position are found and used as seed positions for field lines.

In selecting a surface for the second menu entry one can obviously choose all surface vertices, but often a more useful selection is some patch of surface

multiply or divide map one by map two, putting the result in map one. There is one further operation which is to apply a "convex" correction. It was noticed that potential maps generated on the the Convex used for most of our DelPhi calculations where misread if transferred onto an iris. To be more precise all real numbers (but not integers) were exactly four times too large. This includes the potential map center and scale as well as the potential values. This option then will rescale all those values to their correct value. In addition, if the Grasp box scale had been derived from a potential map then the user is prompted as whether to reduce the box scale by one quarter as well.

One should note in the "map math" that no checking is done to see if the grids actually have the same center and scale, i.e. whether they refer to the same part of physical space. The numbers of corresponding grid points are just added, multiplied etc. The inclusion of a division operation was prompted by the desire to calculate effective dielectrics, i.e. the ratio of the potential calculated with Coulombs Law to that via the Poisson Boltzmann equation. If a zero value is found in the denominator of a division then the new grid value is set to zero.

Math on surface and atom properties are similar in that the same basic operations can be applied to either. There are four operations which require three properties, i.e. a first and second property and a third or target property for the result. These operations are adding, subtracting, multiplying and dividing two properties.

Then there are operations which require two properties, namely multiplying by a constant, adding a constant and special functions. The first property is that acted upon and the second is the target property for the result. Note that this set of operations gives the user a way to put one array into another, for instance by multiplying property one by 1.0 and storing the result in property two. Special functions causes a further menu to appear which lists functions which can be applied to the first array and the result stored in the second. These functions are square root, reciprocal, raise to a power, exponentiation, absolute value, natural log, plus cosine, sine and their hyperbolic equivalents. Illegal operations such as negative square roots are not performed.

Finally there are operations which take but one property and return a single value printed to the textport, namely minimum value, maximum value, average and sum. The first two of these also return the atom or vertex number to which the extrema value is associated.

The properties for each step of the math calculation is selected from a property list for either atoms or surfaces. The user is also prompted for a subset of all atoms or vertices to do the math upon, which is then common to each array. The user should take some care in using this simple math facility to ensure that the property arrays are entered in the correct sequence.

The final selection, the mapping of atom properties to the surface requires that the user enter a selection of atoms, a selection of surface, an atom property and finally a surface property. The connection between surface and atoms is via the original construction process, i.e. which atom is responsible for which part of the surface. If the responsible atom for a particular vertex in the surface selection is not included in the selection of atoms made then no value is set in

radius to zero in the calculation and turn off any assigned charges.

As with surface creation much information is written to the screen during the calculation, some of which is useful to the user in verifying the accuracy of the calculation, i.e. that it has converged, that the correct total charge has been assigned, that the scale of the final map is approximately correct etc. A typical calculation should take about five seconds on a Personal Iris.

The next menu item "Pot. via Map at Surfaces/Atoms" calculates the potential at all surface vertices and all atoms from the current map. If one has just calculated a new potential map this is map one. The algorithm uses trilinear interpolation from the eight grid points which make up the map grid cube an atom center or surface vertex falls within. If it lies outside the map a zero value is assigned. Note that this process will overwrite any previous potentials. If this is a problem the user should first store the previous array in one of the other variables, as described below in the "Simple Property Math" option.

The third menu item "Surface Curvature (+Display)" will cause the program to calculate the surface curvature, as defined in Nicholls et al., for a set of surface points and a set of atoms. As our definition of curvature is related to accessibility of water to a single water placed in contact with a particular surface point, the set of atoms chosen is crucial, since the hard sphere radii of these atoms will determine this accessibility (Note that a zero radius atom does not affect accessibility). Thus, for instance, if one want to compare the curvature of two surfaces which make up an interface one should choose surface one and the atoms which made surface one for the first calculation of surface curvature, then choose surface two and its atoms for a second calculation.

Curvature calculations can sometimes take a long time (i.e. over 10 seconds). This is due to the choice of test sphere used to determine accessibility, i.e. more points take a longer time. The choice of this density is made automatically relative to the scale employed in the surface creation. Because Grasp only a fixed number of test densities the time taken in calculation will vary considerably.

Upon completion of a curvature calculation the display should automatically switch to displaying this quantity. The values are scaled to +/- 100, such that 100 would imply that the surface point is completely accessible, which will never happen since this would imply that one could put a water molecule anywhere in contact with the original water molecule touching the surface. However -100 implies that the surface water is completely isolated from other water molecules, which is quite possible, for instance in a deep cleft or if the surface is enclosed inside the molecule, i.e. is part of a molecular cavity.

"Simple Property Math" is one the most useful options in Grasp. Its submenu gives the user four choices namely maps, surface properties, atom properties and atom to surface projections. The "maps" option allows the user to swap maps, i.e. put map one into map two and map two into map one, to add map two to map one, the result going to map one, to subtract map two from map one, again putting the result in map one, to multiply either map by a constant, and to

of the volume common to all molecules. At present there are no options to this facility and all atoms are included in the calculation. The map is stored as internal map two.

Calculate

The calculate option does much that is unique in Grasp. It allows the user to quickly calculate electrostatic quantities like maps, fields, site potentials, as well as surface curvatures, distances and volumes, plus manipulate fields of information previously calculated or imported to the program.

The Calculate menu is as follows:

New Potential Map
Pot. via Map at Surfaces/Atoms
Surface Curvature (+Display)
Simple Property Math
Dipole Moment
Field Lines
Field Vectors
Volume of a Surface/Molecule
Area of a Surface/Molecule
Distance Map

Calculating a New Potential Map causes the program to execute its internal Poisson Boltzmann solver. The size (in Angstroms) of the map produced is automatically determined. Only the *linearized* Poisson-Boltzmann equation is solved. Such parameters as the probe radius, ion exclusion radius, salt concentration and inner and outer dielectric values can all be set via the menus "Set Parameter :Electrostatic Parameters". The map produced is stored in internal map one. If no charges are assigned then the procedure will inform the user and then abort rather than calculate a null map.

It is important to realize that by default all atoms and therefore all charges are used in the calculation. If the user wants to perform a calculation on a subset of atoms then those atoms not required must have their radii and charges set to zero. The algorithm will ignore any atoms with radius zero in so much as they will not contribute to the water exclusion (=low dielectric) volume. However it will NOT ignore the charges on these atoms. Therefore one needs to neutralize the charges on the unwanted atoms as well. Since changing radii to zero also causes atoms not to be displayed, the user ought to remember to reset those atoms to their correct radius after the calculation, for instance by reading in a default size file.

A typical use of the above procedure for removing some atoms from an electrostatics calculation is where the original crystallographic coordinates are included for several water molecules. It is always an interesting question as to whether such waters should be treated as low dielectric, i.e. as constrained in their motion, or high dielectric, i.e. as bulk water. A good rule to use is that if a water is highly coordinated to the protein, i.e. makes two or more hydrogen bonds a case can be made for low dielectric, otherwise set it as high, i.e. set its

one disconnected constructed convex surfaces. For this reason the user should calculate cavities on the surface of the whole molecule, not subsets. Printed to the screen are the number of triangles that make up each cavity so found. Cavities are automatically displayed in the same display mode as the molecular surface. Alternatively, as described above, they can be sequentially colored.

Building DNA boxes requires that a DNA pdb file has been reading. (Note that mixed files i.e. ones containing DNA and protein are fine.) Display should be automatic. Grasp can handle structures with up to four independent backbone strands.

Making contours requires two inputs by the user, the isopotential value and the color to be assigned to that contour. The latter should be a color index, i.e. an integer between 1 and 99. One can enter more than one isopotential value to create more than one contour at a time, as long as one enters the same number of colors. For instance,

>> Enter Contour Value (written to screen)
>> 1.0,2.0,3.0 (user enters)
>> Enter Color(s) (written to screen)
>>2,3,4 (user enters)

will create the contours at one, two and three kt, and give them colors red, green and blue.

To delete a contour (which may be necessary to make room for new ones) one goes through the same procedure as making a contour of the same isopotential value, except one gives it color zero, i.e.

>> Enter Contour Value (written to screen)
>> 1.0 (user enters)
>> Enter Color(s) (written to screen)
>>0 (user enters)

will remove the contour at 1.o kt. Note that this remove is NOT the same as hide or uncolor, it actually removes the contour data from the program.

Although the usual use of "Build Contour" will be isopotential contours it can be used for more varied purposes since the actual contents of the map are irrelevant to the contour facility. For example, one can contour a DelPhi "eps" map if one has read one in, or one can contour a consensus volume map (see below) if one is calculated.

Finally, one should remember that there are two internal maps in Grasp. The contouring proceeds on which ever map is "current". This is usually map one, but will be map two, for example, if one has just read in a map. To be sure of which map is current, set it via the menus "Miscellaneous: Change Current Map" or with Control C (Note, be careful not to hit Control C while the cursor lies over the textport or the program will abruptly terminate).

Consensus Volume produces a map, i.e. a 3D lattice of values. It is calculated by adding the value 1.0 to each grid point which lies within the Van der Waals volume of each molecule. Thus if there are five molecules and a grid point lies within all five it will be assigned a value of 5.0. Some points will fall within only some molecules. This map can then be contoured at any level desired. For instance contouring at the level of the number of molecules will give the surface

Molecular and accessible surfaces are made by essentially the same algorithm. On choosing to construct one or the other the user has to consider two options. The first is which atoms to use in forming the surface, the second is whether to add this surface to previously constructed surfaces, or to overwrite them, i.e. delete previous surfaces (Note that this is the only way to delete a surface from within Grasp). The menu for selecting a subset of atoms is one which occurs quite frequently within Grasp. It gives the user the choice of:
All Atoms
A Molecule
A Format Subset
Enter String
The first is obvious, the second will will result in a menu containing all molecule numbers, e.g. molecules one through five, the third will give a menu with the names of all atom based format subsets, and the fourth will cause the cursor to jump to the textport and wait for the user to enter a subsetting command. If no atoms are chosen at the end of this procedure the routine aborts.

The process of constructing the molecular surface occurs via the construction of a temporary accessible surface. A correspondence between the vertices of this intermediate surface and the underlying atoms improves the accuracy of the final surface. It also allows for a unique mapping between atoms and molecular surface, i.e. each accessible surface vertex is assigned an underlying atom, and each molecular surface point is assigned an accessible surface point. The combination of these two assignments leads to the association of each molecular surface point with an atom, an association which is termed "contact" within the program.

The process of surface formation will cause plenty of information to be written to the screen, including the scale at which the surface is constructed and the number of vertices and triangles in the completed product. The information here can be useful in debugging (for example the total number of vertices might exceed the maximum allowed number).

Surfaces should appear automatically after being calculated. They will not be colored however. This must be done by the user via whatever method chosen, i.e. calculate potentials at the surface, calculate the surface curvature etc.

Backbone Worm and Boxes are built for all current data, i.e. all molecules. There are no options as yet associated with these objects. If they fail to appear after construction go through the "Display" menu explicitly. The backbone worm can be slow to display. Sometimes this can be put to good use. For instance if the user switches to single buffer mode (see above, Control R) while the worm is being drawn the path of the chain from N terminus to C terminus is nicely illustrated. The backbone worm only requires the carbon alpha positions to be correctly produced, while the backbone boxes require also the carbonyl oxygen and amide nitrogen positions to successfully complete construction

Building Cavities causes the program to check all vertices for connectivity. The first point, or seed point, is chosen at an extrema, and so can not belong to a cavity. All points associated with it, i.e. which can be reached by travelling along triangle edges, are deemed the "non-cavity surface". All others belong to cavities. Note that this will give an incorrect assessment if there are more than

option. Note that if the user has defined one or more formal subsets (see later) then only the views of the subsets on the side for which the dials are attached can be moved.

The next two entries involve the "stereo" part of the "stereo/split screen" functionality. The first of these causes the right hand view to be twisted in an axis running through the center of its world coordinate system in the Y direction (vertical) by a certain angle This twist is set to eight degrees by default. This value can be altered in several ways. If the user has a dialbox then the left bottom dial will alter the stereo twist. The user can also fix the Z-rotation function to stereo twist via the "Mouse Functions: Alternative Z-translation" menu combination. Finally, if the user chooses the fifth menu option in this menu then the second option of this submenu allow the user to enter the value explicitly. This submenu also allows the user to return to the default value or to remove the twist all together AND remove the stereo separation, i.e. to superimpose the left and right views.

Even if the user is in "twist" mode the user can still independently manipulate the two views spatially. This is not recommended since the purpose of twist mode is to allow stereo viewing which requires the two views to be essentially identical *except* for the vertical twist.

If whilst in split screen mode the user attempts to change display properties Grasp will prompt the user as to which "side" the changes should be made. For instance, if the user attempts to hide the bonding display the choice is presented of "left", "right" or "both", the implications of which should be clear. Thus the display on the left can be representing one facet of a molecule (e.g. electrostatic potential on a surface) while the right represents another (e.g. atomic B-value).

When the user quits the "stereo/split screen" option the left-hand side orientations (world and subset) are the ones retained for the single screen view. Also all differential display characteristics the user may have applied to the right hand view are, at present, lost. The stereo twist value is however still stored and may be retained throughout a session.

Build

The "build" and the "calculate" menu options are sometimes easy to confuse. For instance, does one *build* a contour or *calculate* a contour. Or are field lines *built* or *calculated*. In general, build deals with the calculating the data intrinsic to a display structure, such as a surface, or a backbone representation, or internal cavities, where as calculate provides numbers which may or may not be related to such structures, such as potential maps, volumes of surfaces etc. The list is,

Molecular Surface
Accessible Surface
Backbone Worm
Backbone Boxes
Cavities
DNA Boxes
Contours
Consensus Volume

length and the field strength this should correspond to, but only when these quantities are calculated, as described later. Note that it is necessary to calculate a vector quantity before displaying it and that the cylinder mode for field lines can be very slow to draw.

The interaction matrix has three draw modes, i.e. ways to draw the interaction strands. These are as lines, fixed width cylinders and variable width cylinders. The first two are self explanatory, the third means that the width of each cylinder will depend upon the absolute value of the interaction. This value is divided by some maximum strength value, rounded off to unity if greater than one and then multiplied by an internal width to give the resultant cylinder thickness. Of these parameters the "maximum strength" value may be altered by the user. This is achieved via the "Extras.." menu option at the bottom of the matrix draw mode menu.

The "Extras.." menu also includes options to alter the exact coordinates of the point within each residue each strand emanates from. Upon choosing this option the user is prompted for a selection on the command line. For instance the user could then input the command "a=ca" where upon the strands would begin on each residues alpha carbon. If more than one atom is selected for a residue then the strand begins at the average position of those atoms.

Also included are options to multiply or divide the interactions by the distance between interaction sites. The new maximum and minimum values are written to the textport upon each use of this option. Note that if the draw mode is set to variable widths then this can affect the relative widths of the strand cylinders. To maintain a similar spread of widths the following process of rescaling the maximum cylinder width value is enacted. The ratio of the largest (absolute) interaction value (before distance scaling) to the value set for the maximum width is found. Then the largest (absolute) value is found after distance scaling, and the value for maximum cylinder width set so the ratio just calculated is maintained.

Finally we describe the stereo or split screen option (note that his option can also be accessed via Control S). When selected the user will be presented with the following menu,

Dials to Both
Dials to Right
Dials to Left
Right-Hand Twist
Stereo Parameters
Stereo/Split Off

The first three entries decide which side is going to be "attached" to the rotations and translations as entered by the mouse or dialbox. The default entry for this menu is the first, i.e. both views are moved equally. Note that in this default mode the two views differ only by an imposed separation (of 0.5 box units) in the X direction, although the views may not *appear* identical due to the perspective automatically included in all Grasp views. Choosing the second option means the left view can be spatially manipulated independently of the right. To move the right hand view the user has to reinvoke this menu and select the third menu

Help
Quit

Display

This menus controls what is displayed and how it is displayed. With the exception of the "stereo/split screen" option, submenus of this menu are ordered in the following manner,
operation : structure : options.
The operations are to *show* a structure, *alter* it, *hide* it, *hide everything*, followed by the stereo/ split screen option. *Alter* differs from *show* only in that the user is not prompted as to whether a 'default' display is required for the structure chosen. *Hide* causes that structure to disappear from view, but retains all characteristics. Note this is different from using the 'uncolor' option for some structures. If a *hidden* structure is *shown* it returns to view as it was before hidden. *Hide everything* is just a shortcut to clearing the view completely.

The *structure* list contains SURFACES, ATOMS, BONDS, CAVITIES, OBJECTS, CONTOURS, VECTORS and interaction MATRICES. All have been mentioned before in some detail except vectors, which consists of field related items such as dipole vectors, field lines and field vectors. The object entry comes with a submenu listing the possible objects, i.e. backbone worm and boxes, DNA bases, backbone, sugars and axis. For some structures the menu entries just set flags to tell the program to draw this structure, or hide it, depending on the previous menu choice. However most have options associated, especially those in the more developed parts of Grasp.

Options for atoms and surfaces are the property to be displayed, e.g. potential, distance, discrete color, etc., and the drawing mode for that structure, e.g. for surfaces whether lit, pseudo-lit, mesh or points, for atoms, flat spheres, full spheres (CPK), flat but patterned, small "bond" atoms, line spheres or point spheres. For bonds the user gets the option of three coloring schemes, namely to let the user set them (default color=1), adopt the underlying atom colors, or apply an internal color scheme, followed by the bond draw mode, i.e. lines, sticks and cylinders. (N.B. cylinders are very slow to draw). For cavities one has the option of displaying the surfaces as the molecular surfaces are displayed or individually colored. Backbone objects come with no menu options, although see the command line entry for backbone boxes as to how to individually color them.

DNA objects come with several possible options. Most of these are only appropriate if data in the form of a Curves file has been read in, because the options refer to which of the many Curves parameters are to be displayed.

Contour draw modes can be altered to the usual types for surfaces, i.e. mesh, points, lit, pseudo-lit. Control of depth shading for contours is via the depth shading entry in the miscellaneous menu. Colors of contours as well as their values are set when calculated, as later described.

The vectors possible are for molecular dipole (a large 3-D arrow of length 0.3 box units), electric field lines which can be drawn as lines or tubes, or electric field vectors which can be drawn of constant length or with length dependent on the field strength. In the latter case the user can specify the maximum vector

Grasp Menus

Menus carry most of the functionality of Grasp. All menus are accessed via the right-most button. All selections within a menu must also be chosen with the right-most button. Menus appear when this button is depressed. It remains when the button is released. Note this first release of the right-most button does NOT select an item. The program is essentially frozen until the right-most button is depressed and released AGAIN. If the cursor lies over a menu entry when the button is released this second time that menu item is chosen. If the user 'clicks away' i.e. releases the button when the cursor does not lie on an entry either the function will abort, or in some circumstances it will continue with a default value. For instance when the user is altering the molecular surface they first get a menu for the quantity displayed on the surface and secondly for the draw mode (e.g. mesh, lit, points..) of the surface. If one does not want to change the quantity displayed one clicks away from the first menu and continues on to the menu for the draw mode. On the other hand, if the user were to choose the Build entry in the root menu and then click away the program exits from the menuing system.

Many menu entries will produce another menu. This should appear positioned so that the upper left hand corner is at the same position as the previous menu. The root menus should appear such that the upper left hand corner coincides with the cursor position when the right-most button is initially depressed. Upon completion of menuing the cursor should return to this same position.

Some parts of some widgets, such as the color scale, are *sensitive* to the right-most button. This means that if that button is depressed the normal root menu will not appear. Instead the user will get a menu associated with that widget. Clicking anywhere else in the graphics window will get the user the "root" menu for Grasp.

The description below of Grasp functions will follow the order of the root menu, with detours where appropriate.

The Root Menu

Display
Build
Calculate
Mouse Functions
Read
Write
Formal Subsets
Programs
Set Parameters
Miscellaneous

above, i.e. suppose we want to select all strands running between lysines and glutamates. The following set of commands will color these, and only these, red.
ic=0 , uncolor all strands
ic=3, r=lys , color all strands originating or ending on lysines green.
ic= 2, icd=3,r=glu , color all strands originating or ending on glutamates, which are already colored green
ic=0, icd=3 , uncolor, i.e. hide, all those strands which are still green.
There is no undo or restore for strand colors.

Finally, one can find the connectedness of sites via strands using the following variant of the 'ic' command. Typing,
ic=c
causes Grasp to work out the connectedness of all visible strands. That is to say if two strands will be assigned to the same group if they are connected to the same residue, or if it is possible to go from one to the other via other visible strands. Grasp will then report the number of such patches and color each differently. (Only nine different colors are used so if there are more than nine patches the colors repeat).

Note that there is no restriction applicable after this command i.e. one can not restrict this command to certain residues. Instead the proper usage would be, for instance, to first color all strands which are greater than a certain strength, then issue the 'ic=c' command. This will then show up how far 'webs' of that interaction threshold spread.

All other matrix strand operations, such as scaling by distance, deciding upon the display mode, and the maximum interactions strengths used in width encoding, are under menu command and are described in that section.

the color command,
kc=d(efault)
This command can be followed by any subsetting commands for atoms, just like the regular 'kc' command.

Like bonds, backbone boxes also have a single unique property assigned to them, that being the color assigned to them. This can be invoked by the command 'kcd', for instance,
kd=4, rn=(8,33), kcd=3 , i.e. color blue (=4) all backbone boxes which are a part of residues 8 to 33 which are currently colored green (=3).

Matrix Strands:

Matrix values are not calculated at present by any part of Grasp. They must be imported via a data file. There are two formats for such files which are described in appendix A on file formats.

There are some special commands and syntaxes for pair-wise interactions. These include:

strand strength, ip=*x*, ip=(*x*,*y*), ip=>*x*, ip=<*y*
e.g. ip=> 0.0 , all positive interactions.

strand strength rank, ip=*n*
e.g. ip=2 , select all those strands which are the most positive, or second most positive for that residue.
e.g. ip=-1, select only those strands which are the most negative for that residue.

Both the above commands can be subsetted by using any atom commands. This is implemented by adding commands to an 'ic' command which would select a set of atoms. If then any atom of a residue is selected then that residue becomes selected. For instance,
ic=2, ip=>0.0, a=oe1
will color all strands which have positive strength and originate (or end) on a glutamate or glutamine since these are the only residues with atoms labelled 'oe1'. Similarly,
ic=2, ip=1, r=lys
will select all strands which are the most positive strand coming out of either residue it connects, and for which either of these residues is a lysine.
This latter point is worth repeating, i.e. that the program will check both ends of the strand for selection and uses OR to determine selection. (As such there is no direct way to select on the basis of both ends of the strand, e.g. to only strands which go between lysines and glutamate. However see the 'icd' command below.) This is a notable exception of the usual Grasp philosophy of commands always being AND based, and reflects the two-site nature of matrix strands.

Like all other structures the assigned color becomes a property and can be used in subsetting, i.e. the command 'icd' will select on current color. For instance,
ic=0, icd=2 , i.e. uncolor (=hide) all strands currently colored red.
Note that this command can be used to do the "double" selection mentioned

but, for instance, if one calculates the maximum potential on a surface, or portion of a surface, the vertex number of that point is also returned. Hence 'vn' along with a 'vc' command can be used to locate this point.

calculated surface curvature, C=*x*, C=(*x*,*y*), C=>*x*, C=<*y*
e.g. C=>0.0 , all concave parts of the surface.
N.B. must first calculate the surface curvature.

constructed surface number, m=*n*, m=(*n*,*m*), m=>*n*, m=<*m*
e.g. s=(1,3) , the first three surfaces read in or constructed.
N.B. analogous to molecule number except that surfaces are also constructed as well as imported.

Bonds:

One selects bonds on the basis of the atoms they originate from (except see above on *Projections*). Note that although bonds conceptually go between pairs of atoms, in Grasp they go half-way, i.e. each bond 'object' goes from the center of an atom half way to the other atom of the bond pair. Each 'bond' is then uniquely associated with an atom. And so when a bond is colored actually only half the bond is acted upon.

All atom commands may be used in assigning bond colors, for example
bc=4,a=n???
will assign the color 4, normally blue, to all half-bonds originating from any nitrogen atom.

In Grasp the only property bonds possess other than those of their atoms is the color they have been assigned, the default value of which is one. This can be accessed via the command,
bcd=*n*
e.g. bc=2,r=pro,bcd=3 ,
will color red all those bonds in any proline which are currently colored green. There is no undo or restore for bonds at present.

Note that bonds colors can be automatically mapped from underlying atoms via the appropriate menu command. There is also an internal color scheme for bonds which can be selected from this menu.

Backbone Boxes:

One selects backbone boxes some what similarly to how one selects bond colors, i.e. by specifying a subset of atoms. The backbone boxes are constructed out of four atoms, the alpha carbon of the first residue, the carboxyl oxygen of that residue, the amine nitrogen of the next residue and the alpha carbon of that next residue. If any of these atoms are selected the backbone box that uses that atom is assigned that color.

As has also been mentioned there is an alternative color scheme for the boxes where the corners are colored white (alpha carbon), red (oxygen) or blue (nitrogen) to indicate the electrostatic character of the peptide plane. This is the default color scheme for backbone boxes but it can be explicitly invoked by using

The default color for atoms is one.

formal subset name, sub=*abcd.*....
e.g. sub=m1:a
N.B. if you choose subset names, rather than accept those produced for you by Grasp, try not to start the name with a "-", since this will be interpreted as a NOT symbol.
N.B. subset numbers can also be substituted for subset names, i.e. if m1:a is the first subset created then,
sub=1
will have the same affect as sub=m1:a

Surfaces:
Many of the letter codes for surface properties are the same as those for atoms listed above. Since the context should be there is no danger of non-uniqueness, i.e. when one is using any of the following with the vertex coloring command 'vc=*n*' it is clear the properties being specified belong to the vertices not atoms. Similarly any requests for subsetting via the menus will entail a similar understanding. This said, the following commands are identical for both atoms and vertices:
p=
x=
y=
z=
X=
Y=
Z=
d=
sub=
p1=
p2=
i.e. they both refer to the same property but that for atoms in one case and for vertices in the other.
Commands which are specific for surface vertices are:

assigned discrete color, vcd=*n*
e.g. vcd=1 , all parts of the surface assigned color index one
N.B. The default discrete color for surfaces is *91*. The index color one is not used because it is usually altered to be less than pure white, which although is okay visually for atoms, looks awful on surfaces, whereas color ninety one is always unaltered white..

vertex number, vn=*n*, vn=(*n*,*m*), vn=>*n*, vn=<*m*
e.g. vn=15677
N.B. vertex number is assigned like atom number, i.e. serially upon being imported or constructed. Vertex number might not seem as useful a variable,

Assigned charge, q=*x*, q=(*x*,*y*), q=>*x*, q=<*y*
e.g. q=>0.0 , all positively charged atoms.

Assigned Radius, R=*x*, R=(*x*,*y*), R=>*x*, R=<*y*
e.g. R=-(0.0) all atoms which have been assigned a non-zero radius.

Calculated potential, p=*x*, p=(*x*,*y*), p=>*x*, p=<*y*
e.q. p=>10.0 , all atoms at whose center the calculated potential is less than 10.
N.B. potentials are in kt, (1 kt = 0.6 kcals).

Original Coordinates, x=*x*, y=(*x*,*y*), z=>*x*, z=<*y*
e.g. x=(50.0 , 60.0), y=>45.0
N.B. these are the coordinates as they appear in the pdb file, in Ångstroms. These are unchanged by any user applied rotations or translations.

Screen Coordinates, X=*x*, Y=(*x*,*y*), Z=>*x*, Z=<*y*
e.g. X=>0.5
N.B. these coordinates are relative to the screen and in box coordinates. They alter as the molecule is moved. The coordinates here are relative to the unit box, i.e. which runs from plus to minus one, so the example given above will pick all atoms which are in the right most quarter of the screen.

General Property One, p1=*x*, p1=(*x*,*y*), p1=>*x*, p1=<*y*
General Property Two, p2=*x*, p2=(*x*,*y*), p2=>*x*, p2=<*y*
e.g. p1=(5.5,6.5)

molecule number, m=*n*, m=(*n*,*m*), m=>*n*, m=<*m*
e.g. m=1, the first imported molecule.
N.B. molecule number refers to the order in which molecules are read in, i.e. the first structure read in is assigned the number one, the second two and so on. If more than one molecule is read from one file the molecule numbers are sequentially assigned.

accessible area, S=*x*, S=(*x*,*y*), S=>*x*, S=<*y*
e.g. S=0.0 , all buried atoms.
N.B. one should first done an accessible area calculation to be able to select based upon it. See later under *Calculation* for the appropriate menu entry.

distance, d=*x*, d=(*x*,*y*), d=>*x*, d=<*y*
e.g. d=<3.0 , all atoms three Ångstroms or less from some set of points.
N.B. as above one must have calculated a distance map to sensibly select based on this property.

assigned discrete color, cd=*n*
e.g. cd=7
N.B. The user can select against color zero, i.e. select for atoms not displayed.

Notice case INsensitivity.
N.B. one can use question marks to act as wild cards
e.g. a=oe? will pick atoms with names "OE1" and "OE2". One can use "_" to indicate an intentional blank,
e.g. a=o_1 will pick only the field "o 1" or "O 1"
There are four characters read for each atom name.
There are a couple of short cut names for backbone atoms and for side chain atoms (defined as NOT backbone atoms). They are,
a=ba which selects atoms C, CA, N, O, HA, and HN.
a=sch which is equivalent to a=-ba.

residue name, r=*abc*
e.g. r=lys , all atoms in all lysines.
N.B. As above for atom name, except there are only three characters for residue name.
There are several short cut names for residues based upon their hydrophilicity. They are:
r=crg, which selects all residues which are normally assigned formal charges, i.e. lysine, arginine, glutamate, aspartate
r=pol, which selects all the residues formally polar, i.e.
serine, threonine, tyrosine, histidine, cystine, asparagine, glutamine, and tryptophan.
r=hyd, which selects all hydrophobic residues, i.e.
alanine, valine, leucine, isoleucine, methionine, phenylalanine and proline.
The only residue not included above is glycine, which can of course be selected by 'r=gly'.

atom number, an=*n*, an=(*n*,*m*), an=>*n*, an=<*m*
e.g. an=(1,500) , the first five hundred atoms.
N.B. atom number refers to the internal numbering scheme, i.e. atom 256 is the 256th atom to be read in, NOT an atom assigned an atom number in the pdb file. (The field reserved in the pdb specification for atom number is not interpreted by Grasp.)

residue number, rn=*n*, rn=(*n*,*m*), rn=>*n*, rn=<*m*
e.g. rn=(1,25) , all residues which have residue numbers between 1 and 25.
N.B. residue numbers ARE as described in the pdb file, i.e. unlike atom numbers. A consequence of this is that while every atom has a unique number different residues (e.g. on different chains) could have the same number. If a residue number is not assigned in the pdb file it is assigned a value of one for internal purposes.

chain name, ch=*a*
e.g. ch=-B , all atoms NOT in chain B
N.B. if there is no chain specifier in the pdb file it is assigned the letter "A".

e.g.
c=3, p=-(1,2)
will clearly color all atoms whose potential is NOT in the range 1.0 kt to 2.0 kt, but
c=1, q=-1.0
will color all atoms with a charge of minus one, i.e. will not color all atoms which do NOT have a charge of one. If such an ambiguity exist one should wrap the value in brackets, i.e.
c=1, q=-(1.0)
will color all atoms which do NOT have a charge of one.

All subsetting commands can be combined on the command line by separating them by commas. For instance,
c=2,r=lys,q=-(0.0), X=<0.0
will color all atoms which are in lysines, have non-zero charge AND which are in the left hand side of the screen. In other words a comma, when not inside of brackets, .e.g. specifying a range of values, act as a logical AND.

Projection allows the user to select groups of atoms based upon a different, usually smaller, selection. The major use of this at the moment is to select a residue based upon whether an atom of that residue has been selected. For instance, suppose one wants to know which residues are within five angstroms of a certain cofactor which has a residue label of CYT. First one calculates a distance map for all atoms to this cofactor. One would then select all atoms which are within five Ångstroms of the cofactor. Finally one wants to *project* this onto the 'parent' residues, i.e. find all residues which have at least one atom within five Ångstroms of the cofactor. The command to do these last two operations and to color all atoms in such residues red is,
c=2, d=<5.0 , r=-cyt >r
i.e. the symbols '>r' will cause projection to the appropriate residues.

The only other current use for projection is when coloring bonds. Usually one colors bonds via the command line by selecting a color and a subset of atoms. Those bonds which originate from those atoms are then so colored. If, however, the user wants to color based upon which atoms the bond is ending on the correct syntax is ">p". For example,
bc=0, a=sch >p

This will 'uncolor' i.e. hide all bonds made *to* side chain atoms.

The concept of "projection" will be expanded in future versions.

<u>Subsetting Syntax</u>

The above command or functions, e.g. coloring, altering and listing, are usually are followed by subsetting specifications. One can subset by many properties of surface and atoms and other objects. There follows a complete listing of the current key letters or words for such properties, along with examples of how each should be used.

<u>Atoms:</u>
atom name, a=*abcd*
e.g. a=oe1 will select all atoms with the name "oe1" or "OE1" or Oe1" or "oE1".

One can alter the list in one of three ways, namely adding properties, removing properties, or resetting the whole list. Examples of the commands to do each are listed below,

si=pC , reset surface list to ONLY potential and curvature.
ai=qr , reset the atom list to charges and radii, BUT do not remove default atom information if it is on the list already.
ai=+b , add subset name to the atom list.
ai=-a , remove default atom information from atom list.
si=+xd , add absolute coordinates and distance to the surface list.

Note that the order of the parameter entered does not change the order in which the parameters are written to the textport. That order is determined by the position in the above lists of properties from top to bottom which corresponds to left to right output.

Alter

The command to alter a radius or a charge is:

alt(r=*x*)
and
alt(q=*x*)

respectively, the *x* being the new value. Also supported are the following:

alt(r=r+*x*)
alt(r=r**x*)
alt(q=q+*x*)
alt(q=q**x*)

i.e. implicit modifications of each. Typically these commands will be followed by subsetting command. For instance,

alt(q=1.0), a=nz, r=lys

will assign a plus one charge to all zeta nitrogens of lysine.

Note that charge is a display property for atoms and so if the charge distribution is altered via the command line it will cause a recalculation of the maximum and minimum values used for color scaling.

Note also that one can use these commands in an external file as an alternative method of assigning charges and radii to the DelPhi control files. Simply edit a file to contain the list of commands to assign charges or radii, or both, and then read in the file as a Grasp Script File (see section on Menu Operations).

Finally, note that atoms of zero radii are not displayed, or used in calculations of curvature, accessible area, or low dielectric volume.

Negation, Concatenation and Projection

All subsetting commands listed below can be negated. This is achieved by putting a minus sign immediately after the equals sign. For instance,

c=2, r=-lys

will color red all residues which are NOT lysine. If the selection variable is a number not a character then one has to consider whether confusion would result,

yr=*x* i.e. rotate about the y axis by *x* degrees
zr=*x* i.e. rotate about the z axis by *x* degrees
Xt=*x* i.e. translate in the x direction by *x* Box Units
Yt=*x* i.e. translate in the y direction by *x* Box Units
Zt=*x* i.e. translate in the z direction by *x* Box Units
N.B. These commands can be supplemented only by formal subset specifications, as described below. For example,
xr=90, sub=m1:a
will only rotate the atoms and/or vertices of formal subset "m1:a"

List

By entering the keyword 'list' Grasp will print atom information to the textport on each atom. The type of information is controlled by the *atom information parameter list* as described below. Without any restriction this command will list this data for all atoms. More often the user will want information on a certain subset of atoms, for instance,
list, r=lys,a=nz
will list the information only on the terminal zeta nitrogen of lysines.

If there are more than four atoms selected Grasp will automatically expand the textport to list up to twenty atoms at a time. If there are more than twenty atoms in the list, hitting the space bar will give the next twenty atoms, while hitting 'return' lists the next atom. Hence the output is controlled like that supplied by the Unix command 'more', except that the output can be terminated by hitting any key *other* than 'space bar' or 'return'.

Upon completion of the listing hit 'return' to reduce the textport to its original size and position.

Changing atom/surface information parameters

Grasp maintains a list as to which atom or surface properties are written to the textport by a list command (atoms only) or by "picking" with the mouse (both). The default property for the surface vertices is potential, while for atoms it is the atom name, residue name, residue number and chain name. (In fact for atoms it is simply the character field (12:26) in the pdb file). Each possible property for surfaces and atoms is assigned a single letter code, thus:

Atoms	Surfaces
default atom information: a	absolute coordinates:x
absolute coordinates: x	surface potential: p
box coordinates: X	curvature:C
distance: d	box coordinates: X
radius: r	distance: d
charge: q	constructed surface number: s
potential: p	subset name: b
molecule number: m	cavity number: V
general property #1: 1	general property #1: 1
general property #2: 2	general property #2: 2
subset name: b	

Mapping Atoms Colors to the Surface

An exception to *n* being an integer involves transferring atom colors to associated vertices. By 'associated' is meant the atom to vertex mapping provided by the intermediate accessible surface. See the section on making a molecular surface for further details. The syntax necessary to map atom colors to the surface is,

vc=a

This can be subsetted only by *atoms*, i.e. one can map the discrete colors of a subset of atoms onto the surface, e.g.

vc=a, ch=b

only maps atoms colors onto those parts of the surface which are associated with chain 'b'. Note that this is the only exception to a general "like-with-like" rule for atoms and surfaces, i.e. that only atom properties can subset atoms and only surface properties surfaces.

Undo and Restore

There are two further exceptions to *n* being an integer for 'vc' and 'c'. The first is

c=u(ndo), or vc=u(ndo)

(The brackets indicate that it is not necessary to complete the word 'undo'). The 'undo' command will change the color of each atom/vertex to what it was before the last coloring command. In other words, whenever atom colors are changed a backup copy of the original array is made and this will then replace the current colors if an 'undo' is entered, and similarly for surface vertices. The most common use of this function is to correct a mistakenly typed command.

The second exception is,

c=r(estore), vc=r(estore)

This command only acts upon atoms or vertices respectively which have been assigned color zero, restoring the color to what it was before it was so set. Note that while 'undo' is only one color command "deep" restore persists until colors are changed from zero. The primary use of this command is usually to "unhide" parts of a molecule or surface.

Note that although these commands may often be entered without specification of a subset they can both be used, with subset commands i.e. one can 'restore' only certain atoms, or 'undo' a color command on only part of a surface.

Precise Rotations and Translations

Sometimes is is necessary to move objects by exact distances or angles of rotation. Grasp makes this possible through the command line via the following commands,

xt=*x* i.e. translate in the x direction by the *x* Angstroms
yt=*x* i.e. translate in the y direction by the *x* Angstroms
zt=*x* i.e. translate in the z direction by the *x* Angstroms
xr=*x* i.e. rotate about the x axis by *x* degrees

The Command Line

Entering commands from the keyboard proceeds as outlined above, i.e. the cursor should initially be over the graphics window. The user then just begins to type the command. The cursor should then jump to the textport where the characters should appear. Upon completion of the command hit 'return' and the command will be executed and the cursor return to the graphics screen.

(N.B. The textport may originally not be in the foreground, i.e. has to jump to the front from the background. This usually takes a finite length of time to complete and if the user types particularly fast the second character entered may get lost in the transition, i.e. will not appear in the typed command. If the textport is in the foreground this usually does not occur. Also if the user tries to backspace over the first letter of the command they will find they can not! Simply reenter the command in this case.)

The commands than can be entered via the keyboard are all color assignments for all structures, precise rotations and translations, the setting of radii and charges of atoms, the listing of atom properties to the screen, the type of such information for atoms and surfaces, and subsetting in the context of requests from menu driven functions or the two previous functions. The specific syntaxes for these commands and subsetting is described below. To simplify the description the following notations will be observed for variables:

n or *m*= any integer

x or *y*= any real number

a, *b*, *c* or *d*= any character

Note the Grasp is case insensitive where character variables are concerned, e.g. atom names. This is not always true of the subsetting code letters described below, as will be evident.

Coloring

For altering the colors of atoms the construction is:

c=*n*

where n can be between 0 and 99 and indexes the Grasp color set. Zero is always the color (0,0,0) i.e. black, and ALSO means not to display. Thus typing,

c=0

will cause all atoms to disappear from view. Note there is also a "hide" feature under the menu system which appears to the same thing. The difference is that "hiding" does not alter a property of atoms or any other structure whereas color *is* a property. Also "hide" may only be applied to all atoms, whereas "c=0" can be applied to any subset.

The command for coloring surfaces, bonds, backbone boxes and pair-wise interaction (matrix) strands are respectively,

vc=*n*

bc=*n*

kc=*n*

ic=*n*

In each case coloring to zero has the same effect.

plane is linked to the Grasp Box it does not move when the view is rotated, hence the *map* position of each point on the plane will change, since maps are rotated with the view. Hence the potentials are recalculated upon every time the view is moved. The color coding is calculated as if each point in the plane indeed belonged to a molecular surface, i.e. using the same colors and control values. Hence they may be altered the same way, via the color scale menu (see below). One advantage of the projection plane is that by moving the molecule back and forth in the Z direction one can see patterns of potentials within the molecule. It is also possible to move the projection plane rather than the molecule, i.e. alter the Z value from the default of zero. This is achieved via the "Z-Trans. Alternatives" option under the "Mouse Functions" menu option.

Control Z: Unix Shell

In Unix control Z halts a process and return the user to the shell. In Grasp it has a similar purpose. It actually launches a new shell rather than the one from which Grasp arose, but the effect is the same. The user can issue any unix command, e.g. edit files, change current directory etc. Of course the user can always do these things from within another window on the Iris, but sometimes its just more convenient to hit Control Z. To exit the shell type "quit" or "exit" or "logout". Note the user MUST do this before returning to the program, which is "frozen" until the shell is successfully exited. (N.B. this temporary shell does not have access to any of the aliases in ones login in file, nor will it understand the "~" symbol, or other short-cuts which are set up for the users usual shell.)

Control R: Toggle between Single and Double Buffered Mode
Animation is achieved on an Iris by drawing the view into a secondary or "back" buffer, then switching it into the primary or "front" buffer only when the drawing is complete. In this way none of the actual drawing is seen. Iris machines come with a limited amount of hardware for these two buffers. Power series machines have two 24-bit buffers allowing "full" color mode for animation. Most lower machines come with one 24-bit buffer which is split into two 12-bit buffers for animation. Having 12 bits for three colors means that there are only 4 bits per color, rather than 8 in full mode. Hence the same variety of colors are not available for animation. If the user has such a system and wants to get a 24-bit picture they should switch to single buffer mode. Then all 24-bits are used for coloring, but one "sees" the drawing as the view is constructed. The best use of this is then to enhance a view the user is not going to change, e.g. if one is going to take a photograph from the screen. And occasionally it may be instructive to see how a view is drawn.

Control S: Stereo/Split Screen Mode Menu.
This menu is described in the menuing section

Control T:

Control U: Unhook Dials.
If the dials are used extensively the user may find they suddenly cease to work in one direction. They have come to the end of their range. If this happens Control U will remove the dials from their usual functions. The useless dial can then be rewound without moving the view. Pressing Control U again brings the dials on-line once again.

Control V: Repair Surface.
Occasionally upon construction of a molecular surface there main occur "defects" in the structure, i.e. holes in the rendered surface. These can usually be fixed by hitting Control V once or twice. This is best done before another structure is constructed or imported.

Control W: Swap buffers.
Swap the current front buffer for the back buffer (see Control R).

Control X: Toggle Cross-Hairs On/Off.
Sometimes the cross-hairs are not useful in the view and can be turned off by Control X.

Control Y: Toggle Projection Plane:
The projection plane was briefly described in the features section. Essentially it is like having a molecular surface stretched across the Z=0 plane from X=(-1,1) and Y=(-1,1). The potential at any point in this plane is calculated from whichever potential map is current (although, of course, the values in the maps do not have to be potentials) by the usual trilinear interpolation. Because the

program. What should appear when Control P is hit are nine colored squares, each one with a quadrangle of white, red, green and blue attached to the lower, left, upper and right sides respectively. The colors inside the squares are the first nine indexed colors of Grasp. The index number of each color is written in the center of each square and is such that one is at the bottom left, two the one above it, three above it in the upper left, four in the middle bottom, five above it, six above that, seven bottom right, etc.

When the tool is first invoked the colors, along with their RGB triplet (i.e. Red, Green and Blue components as integers from 0 to 255) are:

index	color	red	green	blue
one	white	255	255	255
two	red	255	0	0
three	green	0	255	0
four	blue	0	0	255
five	magenta	255	0	255
six	cyan	0	255	255
seven	yellow	255	255	0
eight	grey	125	125	125
nine	orange	200	100	50

These values are written in the appropriate quadrangles for each color.

To alter a color the user positions the cursor over one of the side quadrilaterals of that color. Then either the left or middle button is depressed. The middle button increases that component, the left button decreased that component. If the cursor is on the white component it increases/decreases all components. Colors are updated within the squares as the button is held down, as are the red, blue or green component values..

Altered colors have their RGB triplets automatically stored in the file "defcol.dat" in the current directory. This file is automatically read in the next time grasp is started within the same directory.

The next nine colors are accessed by hitting the space bar, and the next nine after that by hitting it once more, and so on. After colors 91 to 99 the screen returns to colors 1 to 9.

Colors 10 to 89 are left intentionally blank for the user to create their own. Colors 91 to 99 repeat color 1 to 9. Whenever the program is restarted colors 91 to 99 are always as listed above for colors one to nine, i.e. when they are altered within the program the changes are not stored in the external color file, unlike changes to all other colors. The user will probably find that uniformly decreasing the red/green/blue components of colors one through nine from their above given values will give richer display colors.

When the user is finished with this tool hitting any key, other than the space bar, will remove it from the screen. Any structure colored by an indexed color which has been altered should automatically update its color.

Control Q: Quit the Program

Control F: Toggle Full Screen View On/Off.
The graphics window can be expanded to fill up the entire screen, removing even the window border from view. When this is turned off the window returns to its previous size and position.

Control G:

Control H:

Control I:

Control J:

Control K:

Control L: Free/Fix Light Source
The direction from which the light source will produces lighting affects on rendered surfaces can be altered after hitting Control L. Moving the mouse (without depressing any button) will then move the light source in that direction. (The source is set at infinity and so only the direction of the light source is altered) Hitting control L a second time fixes the light source's new direction. Note that this will affect the lighting of ALL surfaces, including those pseudo-lit and those which make up the surface of "CPK" atoms. The user should experiment since different lighting angles often bring out different features of surfaces.

Control M: Toggle Textport Depth
Control M is the same as 'return' i.e. it will pop the textport to the front if it is lower down, and push it to the back if it is in front.

Control N:

Control O: Toggle Grasp Box On/On/Off
The Grasp Box has been described before. It represents a box of +/-1, in screen coordinates, in each direction. The first Control O produces a box which is depth shaded, i.e. the box sides get darker the further from the viewer. The edges are also outlined in black. One advantage of this view is that because it has simulated depth it can fool the the eye into expecting any other object in view to have depth. Hence it "trains" the eye to see the objects in the box as three dimensional. Pressing Control O again removes the depth shading of the cube, leaving all sides white. This is provided in case the user is capturing pictures to send to a postscript printer (apparently it makes a difference). Hitting Control O once more removes the box from view.

Control P: Bring Up The Color Palette.
The Color Palette is the tool with which users may alter color from within the

button is used the number will increase.

If the scale is in three color mode and the value being altered is the lowest value, the rate of increase or decrease depends upon the difference between that value and the middle value, i.e. it increases by a constant fraction (10%) of that difference. Similarly it changes the upper value based upon the difference between that value and the middle value. The middle value changes based on the difference between it and the value towards which it is being altered. In two color mode only the left-most and right-most values change, and they do so based on their difference.

The ">-<" part of this widget allows compression and expansion of the color scale range. Depressing the left-most button and holding it on this symbol is equivalent to increasing the left-most number while simultaneously decreasing the right-most number, i.e. it compresses the range. Similarly the middle button will expand the range by decreasing the lowest number, increasing the highest number. This is often useful in viewing electrostatics since the range of potentials is usually much wider than is useful for distinguishing positive and negative parts of a surface.

The color scale menu allows the user to enter new values for the control (i.e. minimum, middle, maximum) numbers, also RGB triplets for the colors used. It also allows the user to reset the control values to their original (extrema) values. There is also support for changing the draw mode and property displayed. As two color continuous is invoked by setting the middle value to the lowest value, as since values can only be altered by fractions of a difference, one must use this menu to so set the control values, and to reset to three color continuous. The color bar menu also allows the user to alter the colors used for display, the draw mode for the surface or atoms, and finally the property being displayed. The latter two options produce menus identical to those which appear in regular menu use for these features and are described later in more detail.

One color bar should appear for each different quantity displayed, i.e. if the user is displaying atom potentials and surface potentials two should appear. If the user is using the split screen mode and different properties appear in the left view and right view, one scale should appear for each.

Control C: Change Current Map

The current map is the 65 cubed set of grid values which are used to build contours, interpolate potentials at a slice plane, interpolate potentials at a surface etc. There are two such maps in Grasp, the first of which is the default space for all internally generated maps, the second for all maps read into Grasp. Hitting control C produces a menu the user can use to choose which is "current". The user should beware of hitting control C when the cursor is NOT over the graphics window, since if it is over the textport the program will terminate.

Control D:

Control E:

Control Keys

Control keys at present in Grasp mean any alphabetical character depressed while the control key is depressed. Grasp does not yet use any "Alt" keys or any function keys. To active control keys the cursor must be on the graphics window. They are included because people (for instance me) find them useful to have as an alternative to menu driven functions. Not all control keys are at present used, and some may be altered in the future if it becomes clear their particular use is so infrequent a more used function should be assigned to them. Most control key functions can also be found in menus. The current list of such functions is given below:

Control A: Toggle Active Subset Rotations On/Off.
If Grasp is manipulating more than one object (defined as "formal" subsets, see later) independently only one such object can be "attached" to the dials or mouse-dials at a time. This can be set via the menus. However if one wants to switch back and forth between different subsets continuously this can be frustratingly slow. Switching to 'Active' rotations means that when the user is using the mouse to enact translation or rotations the object moved is that upon which the cursor lies when the a mouse button is first depressed. Essentially the object is "picked" then moved. If the cursor lies over no object at the beginning of the motion then the mouse-dials are attached to the world, i.e. everything is moved together. If dials are used instead of the mouse then the object moved is either the one last moved by the mouse or that initially "attached" to the dials when the active mode was invoked. Turning the Active mode off (i.e. pressing control A again) leave the dials attached to the last moved object.

Control B: Toggle Color Scale Widgets On/Off.
Color scales should automatically pop up when a continuous coloring scheme is being displayed. If it does not , or if the user wants to remove them from view, the user can turn them on and off with control B. The color scale has two components, the title space, from which a menu may be accessed by the right-most button and the rest, which controls the color coding.

The color coding part should consist mostly of a colored section and a small white section, with the symbol ">-<" on it, at the right-hand end. The colors should be the colors in use for that property and structure, e.g. red, white and blue for electrostatics, and should vary continuously from right to left. If the coloring mode is two color continuous the colors will vary from the first to third colors, three color continuous will vary from the first to second to third color. Printed on this colored strip should be five, equally spaced, numbers which increase from left to right. The left-most, middle and right-most numbers are those used in the color coding as described in the previous section on program features. If the cursor is placed over any of these three numbers and the left-most button pressed and held down that number will decrease, and continue to decrease while the button is held depressed. When the button is released it causes the display to update color based upon this new number. If the middle

textport and then bringing it back will usually force a redraw. If for any reason the graphics look funny, for instance 'damaged' by the movement of some other window or some other program, or if the initial view upon starting the program looks strange, this is a simple way to redraw the view.

If one has read in a pdb file or a srf file there should now be something displayed from within grasp, either a molecule or a surface. The default display of either of these can be altered as described later. Upon reading a structure (i.e. atoms or surfaces) a scale is assigned to the unit box, i.e. a width in Ångstroms is calculated (and written to the textport). This is calculated such that the structure will fill up two thirds of the the box in its longest dimension. If the user instead has initially read a potential map the scale is such that the potential map will fill the unit box exactly, i.e. the boundaries of the potential map are at +/-1 in each direction. This scale is now set for the duration of the Grasp session as there is, as yet, no facility for altering global scaling. The view can now be manipulated, quantities calculated, structures built, etc. Operational details follow in the next two sections.

Exiting the program can be done three ways. The first is NOT recommended except in emergencies (i.e. the program has inexplicably locked up) and is to put the cursor over the textport and hit control C. The normal way to exit is either through the main menu or via control Q (see 'Control Keys'). If the program is correctly exited one should notice that the cursor is no longer yellow, which it is during normal Grasps operation, unless, of course, this is the users normal color for the cursor.

stationary). The language used in Grasp is of "absolute" or "internal" coordinates, which "belong" to the molecule, and "box" coordinates which belong to the user (and hence the box). One can change molecules into the users frame of reference, i.e. make rotations "real", via formal subsetting and some options for file export, and which are detailed later.

One can now read in one of three type of data files, atom files, surface files or maps (3-D grids). Note that instead of reading in a file once the program has executed one can give the name of a data file after one types grasp, e.g.

grasp lys.pdb

which will load in the coordinates of lysozyme from the pdb file lys.pdb. Other than pdb files one can also give the names of maps, with the extension .phi, or surface files .srf. If the name of the file does not have one of these three extension then the program will prompt the user as to which type of file it is.

Reading a data file from within Grasp involves using the menu system. This is discussed later in greater detail. Clicking on "Read" and then on one of the primary data file type will prompt the user as to enter the file name or select a default file or see a list of files. The list will be all files in the initial directory which have the correct extension for that type of input.

When character input is requested in Grasp it is via what is called the textport, which is the character based window from which Grasp is initially launched. To enter information to the textport the cursor has to be positioned over this window. If Grasp is expecting information, e.g. expecting a file name, it tries to make this easy for the user both by *automatically* positioning the textport over all other windows, and by *automatically* placing the cursor over this window. And when the information input is complete, i.e. return has been hit, the cursor will *automatically* jump back to the spot on the *graphics* window it was before the request for information. Similarly if the user wants to type something to the command line interpreter the cursor will *automatically* place itself over the textport when the user begins to type, jumping back when return is hit.

Note that in both these examples the cursor starts over the graphics window and end there too. The user should NOT attempt to move the cursor onto the textport themselves except in the following two cases. Sometimes the user may have moved the textport, or resized it, and as a consequence the cursor may miss it when made to "jump" by the program. Also possible is that the cursor will come to rest upon the textport when instructions are not being entered. This causes a "change in input focus" for the program i.e. it expects input from the textport rather than from the graphics window. When this happens, for instance, the molecule will not rotate when the dials are twiddled because the program is not "listening". This question of "focus" can often give beginners the most problems in getting started with Grasp, so when in doubt check the cursor position.

Hitting return when the cursor is over the *graphics* window causes the textport to alternate between background and foreground, i.e. being behind all other windows or on top of them. If the user has resized the textport, for instance made it bigger to review more information, or repositioned it, hitting return will also resize and reposition the textport. Another use of this is that pushing the

Once the **grasp** command is issued, and the appropriate files searched for and read, the default graphics window opens and shows a set of axes or cross-hairs for the X, Y and Z directions. The Z direction is towards the viewer in Grasp, positive nearer the user, negative farther away, the X direction is left to right, right positive, left negative, the Y direction is up and down, with up positive, down negative. The cross hairs run between plus and minus one in Grasp internal coordinates. To help visualize this domain one can view the Bounding Box. This is done by pressing control O (see 'Control Keys'), or via the menu entry under 'Miscellaneous' (see 'main menu' below).

One moves this view by either using the dials on a dial box or use of the mouse. The dials work accordingly:
left row, top to bottom= x rotation, y rotation, z rotation, stereo twist
right row, top to bottom= x translation, y translation, z trans. , stereo separation

The mouse moves the view by depressing the left or middle buttons or both and moving the mouse. Specifically,

Left-most button: Rotations about the axis perpendicular to the direction of mouse motion. Hence there is a sense of "rolling" the molecule as if the molecule where resting on a solid surface in the XY plane and the cursor was the user finger.

Middle Button: Up and down moves the molecule away and towards the viewer respectively. Note that this is NOT a scaling, but an actual motion in the Z-direction and hence corresponds to pulling the molecule towards or pushing it away from the user. Left or right motion rotates the molecule about the Z axis.

Left and Middle Together: Translates the molecule in the XY plane in the direction of mouse movement.

This implementation of dials via the mouse (mouse-dials) is a little different from some programs, i.e. only two mouse buttons are used. This is partly because only two are actually required to allow the six independent rotations and translations, but more importantly because the third button, the right-most button, is reserved exclusively for the menu interface.

One further convention is adhered to in Grasp which is that, where appropriate the middle button adds and the left button subtracts. For instance when adjusting the indexed colors in Grasp the middle button increases a color component, the left button decreases it.

The rate of rotation or translation, i.e. the sensitivity to mouse or dials, can be set via the menus. It is also possible to assign different functions to the mouse, such as surface scribing, or projection plane position. These are accessed via menus and will be detailed later.

Note that the box does not rotate with the the cross hairs. The significance of this is that box represents a space which is invariant with respect to the user. Rotations and translations do not actually affect the molecules coordinates, only the viewing of them. One way to think of this is that it is not the molecule which is moving but actually the user and the box (since both move the box appears

The Basics

One can start Grasp simply by typing the program name, e.g.

grasp

There are a few things one should check first however. The first is to ensure that one has write privileges in the directory the command is issued from, which can often be a problem if working from someone else's directory. Grasp needs this permission to enable it to write temporary files, which are removed upon exiting the program, and some permanent data files, such as a color map if the user alters those provided, and also "error" files if it detects odd situations (such as finding too many bonds for an atom when reading a pdb file, or fractionally charged residues upon reading a charge control file).

Secondly Grasp reads in a few small data files upon start up. It needs to know what directory these files are in. To do this it reads the environment variable "GRASP" (note capitals) which should be set to the correct directory. For those not familiar with Unix, the command to do this is

setenv GRASP *dirnam*

where *dirnam* is the directory name. One should place this command in the file ".login" in ones root directory so it is read and executed when the user logs in. One can check the 'value' of this variable by entering,

echo $GRASP

Grasp also has a directory of "last resort", i.e. if it can not find the directory set in GRASP. The backup directory it looks for is "./aakdat", i.e. in a directory one lower than the user is in called "aakdat". These data files are listed in the appendix and involve such things as default radii for atoms, default charge sets and information used in surfacing molecules.

Thirdly, Grasp will check for a file called ".init_grasp". This file can contain commands which set variables within Grasp, such as default display modes. The form of the commands are listed in the appropriate appendix. Grasp searches for this file in three places. Firstly it checks the directory stored in the environment variable GRASP, secondly it checks the users root directory, and lastly it checks the local directory from whence the command was issued to start the program. The purpose of having it check all three locations is to allow for hierarchical control of grasp settings. For instance one might want to set some parameters for all users, in which case they are set in the GRASP directory. Individual users might want different parameters for their own work, and so alter the file in their root directory. Finally, the individual user might find that for some projects different parameters are better, e.g. small molecules might want one set of display parameters, large molecules others, in which case control should be via the file in each particular directory. The order the files are read is important because if two commands set the same parameter preference is given to the later command.

The ends of these lines or cylinders are by default set to the average position of all atoms in the residue. This can be altered to any subset within the residue. For instance, in some cases the position of the residue charge might be more appropriate, in others the alpha carbon, or center of the side chain etc.

Finally since interactions will be distance dependent, the user can explore this dependence by multiplying or dividing each value by the distance between its two sites. This scaling can help the user determine which interactions are unusually strong or weak.

Stereo/Split Screen

Stereo viewing has traditionally been achieved by duplicating the view, separating the views by a certain distance so that there is no overlap, and then giving the right-most view a twist about the vertical direction of about 8 degrees. Grasp follows this approach and extends it to a 'split-screen' capability.

Firstly the stereo separation and twist are under user control. (Tests within our labs showed conclusively that everyone has their own preferential stereo twist.) The separation and twist are under mouse and dial control. Twist may also be entered explicitly.

Secondly, the duplicate view can be treated as completely independent, i.e. nearly all display possibilities can be used *either* on the left *or* the right. This allows the user to display alternate views side by side. For instance one might want to view the surface color coded by potential and also by curvature at the same time. Views can also be superimposed, i.e. the left-right separation eliminated.

Furthermore one can manipulate (i.e. translate or rotate) either view independently. Thus, for instance, one can display front and back of a molecule simultaneously. Formal subsets in each view are also independent, and so different arrangements of such subsets can be portrayed in right and left views.

hydrogen. In this mode it is easy to see, for instance, where backbone loops have all carbonyls or all amide hydrogens aligned in one direction (which can be significant electrostatically). It is also makes secondary structures particularly clear. Boxes can also be colored by subsetting based upon the underlying atoms, or 'uncolored' so as to display only parts of chains.

DNA representations have three components: the phosphate backbone, the pentose sugars and the DNA bases themselves. The backbone can be represented as a thick ribbon smoothly splined through backbone phosphates. The pentose sugars are represented either as rings or line pentagons. The latter are color coded by the endo- or exo- nature of the sugar carbon. Finally the bases are represented as rectangular slabs, colored by base type. This latter representation can also be made to width-encode a DNA base pair parameter. Support is provided for the output from the program CURVES which describe 38 such parameters. These can also be mapped to a helical sheet and color coded. This work is still under development towards a more complete 'module' which will support more typical Grasp-type controls.

Pair-Wise, or Matrix, Representation

One quantity which is difficult to represent by conventional means is pair-wise interactions. This is because the variable has both a value and two positions instead of just one position. For this reason this is like attempting to display a matrix, as opposed to a vector. Such variables occur in electrostatics as the interaction energy between each charge in a set of charged sites. Another example is effective inter-residue forces which some have developed to model protein stability. Since both these uses are essentially residue based, the formulation of the pair-wise interaction representation is residue based in Grasp, i.e. it acts between residues, not atoms.

In Grasp these forces are represented by means of lines running between pairs of sites. These lines have as properties the interaction strength and those properties of both interaction residues themselves. Hence lines may be colored (or 'uncolored') by standard subsetting commands.

Interactions also have the property of 'rank' i.e. since sites may have several interactions each such interaction also has a rank amongst those assigned to that site. Thus one can subset by rank, i.e. only show the strongest interaction for each site. Since interactions can be strong by being very attractive or very repulsive, i.e. the interaction strength very negative or very positive, support is provided for ranking by either criteria. This can be useful in determining 'zones' of interactions i.e. patches of residues which interact mostly amongst themselves.

Grasp goes one step further than merely representing forces with lines by expanding the lines into cylinders. As well as being visually striking this allows the width of the cylinder to act as an indicator of the absolute strength of interaction. By default, Grasp sets the maximum width to represent the maximum absolute interaction, but this can be set to be larger or smaller by the user. All other coloring operations still apply. Grasp also allows for the cylinder representation without width encoding.

one field is to be used as a dummy field. For instance, if charges were assigned to represent helix forming potential via a DelPhi charge file format they can then be passed on to one of the general property fields.

Another use is property inversion. As well as inverting the normals of a surface to make it appear similar to its complement one might also want to invert its electrostatic potentials since one might expect a complementary surface to also be complementary in potential. This is simply achieved by selecting the surface of interest and multiplying its surface potential by -1.0.

Finally *simple math* includes a facility to map an atomic parameter directly to to the surface. One advantage of this is that a surface is to some extent simpler than the collection of underlying atoms and as such is often a better vehicle for displaying properties. One further advantage is that by using the accessible surface one can project these properties into space away for the molecule.

Grasp Settings

Some internal parameters for Grasp can be set in an external 'start-up' file. This feature is not well developed at the moment but does allow the user to specify the initial display of a molecule to be as atoms or as bonds, and also whether surfaces are displayed as lit, pseudo-lit, meshes or points. Both these settings address the concerns that arise if one has a low end Iris but want to look at large molecules. In expensive draw modes these can take several seconds to draw, hence it makes more sense to use a more 'rotatable' representation for examining the molecule, such as a bonding representation.

This facility will be expanded to eventually include most, if not all, of Grasp's internal parameters so that it may be more fully user customizable.

Objects

Objects represent both a new direction for Grasp and an extension of the original concept. The latter because surfaces are in a sense an abstraction of the underlying atoms positions, and as such can be thought of as an 'object', i.e. a form which represents in some way both the shape and a property of a set of atoms. In this sense, back-bone ribbons are objects, even bonds could be so classified (the property being the direction of the bonding force).

In Grasp there are currently two objects for proteins and three for DNA. The first protein object is a backbone trace I call a 'worm'. It consists of a set of cylinders forming a smooth spline though alpha carbon positions. See *Future Developments* for the ideas to be incorporated herein for secondary structure.

The second is a peptide plane representation. As is well known, the peptide bond has double bond character and so is stiff to torsional rotations. As a consequence the set of backbone atoms CA(n)-C(n)-N(n+1)-CA(n+1) lie in a plane. This can be represented as a quadrilateral with corners at carbon alphas (CA) and at the oxygen and hydrogen of carbon (C) and nitrogen (N) respectively. This is given a little width for display purposes to make a quadrilateral box.

These backbone boxes can then be colored in several ways. The default color scheme is white at the alpha carbons, red at the oxygen, blue at the

Other than with the above method Grasp syntax is exclusively AND based, i.e. subsetting by progressive refinement based upon properties. The reason for this is that one of the most useful applications of Grasp is to use it to look for correlations. So one might want find all residues which are a certain distance from the molecular surface AND charged. Ideally Grasp would support a query language such as SQL based upon the properties intrinsic, calculated or imported to it, but at present subsetting represents a limited form of the ability to explore for meaningful or interesting connections.

Finally, there is one further method of selecting a surface subset. By activating the surface scribing mode the user can "draw" upon the surface the border of a region of interest with the mouse cursor in much the same way in which one might with a Macintosh-like draw program. The border appears as triangles colored bright blue. When the outline is complete the user double clicks inside the border. This causes the triangle clicked upon to be "selected" and colored bright green. The selection then spreads outward from this triangle until it comes up against the border. If desired the border can then be added to the selection. If the border is not "water-tight" this selection will "spill" out. Such mistakes can be undone. The filling process can also be halted by a further mouse click. This also allows the user to make a borderless selection, i.e. just double click and a patch from this point will gradually grow until the user clicks once more. Once the selection is finalized it can be made into a formal subset or used as a choice when a surface subset is required by the program. The selected region can be 'unselected' within the scribing mode and is automatically unselected when the scribing mode is turned off.

Simple Math

Sometimes the properties calculated or imported into Grasp are not exactly the ones one is interested in. For instance one might be interested in not the surface potential interpolated from map one or that from map two, but some average of them (for instance weighted by the average ionization state of two residues). The same might be true of the maps themselves i.e. one might like a combination of the two maps. *Simple math* addresses this by providing for arithmetic on one or two fields, putting the result in another (or the same) field.

The operations available for maps include addition or division of two maps, along with multiplication by a constant and swapping the two maps. Those for atom and surface properties are more extensive and include addition, subtraction, division and multiplication of two properties, as well as addition or multiplication, by a constant, of a single property. Also supported are special functions on one field including square root, reciprocal, exponentiation, logarithm, cosine, sine and hyperbolic functions.

More useful to some extent are the contraction operations which take a field and return a single value, i.e. maximum, minimum, average and sum. These can be combined with subsetting so that only portions of the selected fields are acted upon. For instance one can find the accessible area of all lysines, or all charged groups, or all charged groups of a particular helix etc.

Simple math can also be used to shift fields about. For instance, one can multiply a field by 1.0 and place the result in a second field. This can be useful if

position, potential, distance, charge, radius, general atom properties one and two, molecule number, formal subset name, assigned discrete color, atom number as well as atom name, residue number, residue name, chain name, and accessible area. There is a deliberate similarity between these two lists based on the program philosophy of equality between surfaces and atoms.

Variables are either characters or numbers. The latter may be specified as ranges, for instance one can select all atoms which have a residue number between 5 and 10, or a charge greater than zero. Character variables (except formal subset names) may include wild cards, so one can select, for instance, all carbon atoms or all carbon atoms which have the character "1" in the third position of the atom name etc. There are also 'short cut' names for atoms in a protein backbone or in side chains, and for residues which are ionizable, hydrophobic or hydrophilic. All individual specifications can also be negated, i.e. one can subset by NOT a certain quantity.

The purpose of subsetting depends upon the context. One of the simplest uses is to alter the display color of this subset. For instance, one might want to employ a color scheme which displays all positively charged atoms one color and all negatively charged atoms another, or all hydrophobic residues yellow and all hydrophilic purple, etc. Colors are specified by the integer index (1-99) assigned to each within Grasp.

The use of assigned color as a *property* allows for some quite flexible subsetting. For example it can act as a bridge between atomic and surface properties. Suppose we want to find all vertices of a surface which are concave, i.e. have a calculated surface curvature less than zero, and which are formed by hydrophobic residues. One way to do this is to color all hydrophobic residues one color, then transfer that color to the surface, then select all vertices which have that color AND which have curvature less than zero.

Color can be used to build up subsets based upon disparate properties which could not be specified in a single command line. For instance if we want to select all atoms which are in a region of positive potential and belong to negatively charged residues *plus* all those atoms which are negative but belong to positively charged residues. This is a way to include an OR statement into Grasp's subsetting vocabulary.

Another use of subsetting is in creating a *formal subset*. Formal subsets are assigned names, either by the program, in which case a hierarchical naming convention is enforced, or by the user. Formal subsets may be spatially manipulated independently of the rest of the surfaces/atoms. They can also be referred to by name in all subsequent operations. Formal subsets may be sets of atoms or portions or collections of surfaces, or may have mixed character. By mixed is meant that a surface may be associated with a set of atoms, or a set of atoms associated with a surface. For instance, as well as selecting an active site surface one might want to associate all atoms which are in contact with that surface.

Most other uses depend on the action being undertaken. For instance, when the surfacing subprogram is activated the user has the option to enter a subset of atoms. For example, one could surface a single helix in a protein.

increasing graphical complexity. The former are the traditional line drawings used by most programs. Cylinders are just a three dimensional variant of this, the diameter of which can be set by the user. Sticks, however, follow the method of Kuznetsov and Lim in which bonds are represented by quadrilateral tubes. The advantages of this approach are that the bonds are made significantly more three dimensional, inter bond angles are well brought out, and the display is relatively quick to draw.

Bonds can be colored based upon a preset pattern, upon transferring the discrete colors of the underlying atoms, or by subsetting based on the properties of those atoms. They can also be selectively undisplayed by being "uncolored" (see *color* below). Bond colors are depth shaded. Properties of atoms can be queried by picking at bond ends.

Contours

Isopotential contour surfaces can be constructed at any potential value for either internal *map*. Contours can be displayed in all the surface modes available to molecular surfaces, i.e. lit, pseudo-lit, mesh and points. They are also independently depth shaded. Contours are not automatically recalculated if the parent map changes, though contours can be deleted and recalculated. Volume and surface areas may be calculated for any contour.

Colors

Grasp supports ninety-nine independent, indexed colors. These can be set during program execution, using a color 'palette', or in an external file as RGB triplets. All changes to colors are automatically saved, thus the user can design their own set of colors, or use those provided. These colors are numerically indexed, i.e. assigned numbers from one to ninety-nine. There is also color zero, which is always equal to black but which is also used as a flag to prevent display of that object, i.e. an atom colored zero is hidden. This 'uncoloring' applies to surfaces, bonds, backbone boxes and matrix strands. Grasp also has undo and restore commands such that undo removes the previous coloring, while restore acts on objects colored zero by giving them the color previously assigned. Once assigned to atoms or surfaces these discrete colors become a property which can be used in subsequent subsetting selections.

Subsetting

A central feature of Grasp is the ability to specify a subset of atoms, surface vertices, bonds, objects, etc., based upon a very wide range of properties. The method of doing this is via a command line which accepts a series of specifications and forms the intersection of each. There is one exception to subsetting being entirely by command line, this being 'surface scribing' as described below.

The properties which can be used for surfaces are coordinates, screen position, potential, distance, curvature, general surface properties one and two, constructed surface index, formal subset name, assigned discrete color and vertex number. For atoms the list is a little longer, namely coordinates, screen

Since charges are only of importance for electrostatic purposes and since charge is also a display property, it may be utilized as a 'dummy' variable, i.e. actually represent another physical property. This becomes particularly useful when combined with the DelPhi control file format. For instance if one is interested in a per residue property, say helix-forming propensity, a control file can be constructed with one line for each residue and its property value. This can then be read into Grasp, the atoms of each residue will be assigned a 'charge' equal to that residues helix-forming propensity, and can then be color coded and displayed.

Grasp also supports three variants on the standard PDB file which involve the fields to the right of atom coordinates. These typically contain occupancy and B-values. One option is to read this information in as general atom properties one and two, which are generic data fields for discretionary use by the user. There is also an option to read these fields in as the radius and charge of each atom because this is what they are used for in DelPhi modified PDB files. Finally, for higher precision, the entire field to the right of the coordinates can be read, in free format, as general properties one and two. Files in the above formats can all be written from within Grasp.

Separate molecules will be recognized from within a single PDB file if separated by "TER" statements. Each will be assigned an index, i.e. a molecule number, which as a property is analogous to "constructed surface number" in Grasp. Molecules can be superimposed using the Kabsch algorithm, which gives the best rotation and translation (RMS-wise) between molecules or parts of molecules. The only restriction is that the same number of atoms from each molecule must be used to determine the minimum RMS difference. (Note Grasp will not yet superimpose surfaces).

Grasp contains algorithms for calculating both the volume and surface area of molecules or subsets of atoms. Surface area can be either accessible area or that of the van der Waals surface. Control is given to the user over the precision of these calculations.

Grasp does not contain methods to alter structures such as torsional rotations, minimization etc. with one key exception. Grasp allows independent rotations and translations of defined subsets, which may then be "fixed" relative to each other. For instance, a substrate may be selected and moved relative to an active site. Upon making the transformations new surfaces, distances, electrostatic fields etc can be calculated based on the new coordinates. There is support for undoing transformations which have not yet been "fixed".

Atom properties can be queried in the same way as surface properties, i.e. point and click. Atom properties can even be queried when covered by surfaces. The "atom picking" function can also be set to report geometric parameters such as distance, angle and torsion angle between picked pairs, triplets and quadruplets of atoms.

Bonds

Bonding patterns are calculated upon the reading in of a PDB file. Bonds may be represented in three ways, as lines, as sticks and as cylinders, in order of

could be internally generated or both externally generated by DelPhi. Difference maps are particularly useful to highlight the effect of changing parameters such as charge, radii, salt concentration, dielectrics, etc.

As one of the three primary external data files supported by Grasp (the other two being protein data bank (pdb) files for atoms and surface representation files (srf) for surfaces) maps may be read at startup, i.e. one can analyse maps without any atom data or surface data. Although maps are usually associated with electric potential they can be used quite generally for any 3-D data, though at present Grasp requires the grid to be 65 cubed. For example, the consensus volume option in Grasp, i.e. finding the common volume between a set of molecules, results in values assigned to a 65 cubed grid which can then be manipulated and displayed as a potential map (e.g. Z-plane projection, isopotential contours). The dielectric boundary map, or salt exclusion map from DelPhi can also be read in as this form.

Atoms

Atomic coordinates are the fundamental data structure from which everything else is derived. Grasp does not contain any 'build' utility and hence is dependent on external files for this data. Primary support is for PDB files, the standard crystallographic format, along with certain variants.

Atoms can be displayed in several ways. The traditional method, which is included in Grasp, is as spheres of given radius. This is often referred to as CPK modelling. In Grasp the surfaces of the spheres are lit. CPK can be demanding on the graphics resources for large molecules. An efficient alternative is to represent the atom as a circle of the correct radius always oriented flat with respect to the viewer. With a little differential coloring these flat circles can be given an apparent three-dimensionality. There is also an option to instead color these circles with patterns. Also, there is a representation which gives small, uniformly sized circles for use with bond representations (see *bonds*). Finally, spheres may be drawn with lines or dots, i.e. unrendered.

Atoms can be colored discretely, i.e. any subset of atoms can be assigned any color, or continuous color colding may be used, as described in *color coding*. The properties supported for this are potential, distance, charge and two general property fields. Atoms can be uncolored to remove them from view. Atom colors are depth shaded.

Upon reading a PDB file radii are set to default values from an external file (which the user may edit). This file is in the same format as the control file for atom size used by DelPhi. Similar files may be read (i.e. radii assigned) during program execution. Charges are zero by default when a structure is read, but Grasp can read DelPhi charge files and assign charges based upon the descriptions therein. Some sample files are provided, such as those to assign charges to each ionizable residue of a protein. Radii and charges may also be assigned via the command line by specifying a radius/charge and the subset of atoms to have this radius/charge. The command line also supports intrinsic operations such as multiplication of radii/charges by constants, or addition of constants.

hence that program has to be run separately.

The Grasp PB solver uses two 33 cubed grids, one nested within the other. The inner grid dimension is set to be larger, by the diameter of one water molecule, than the maximum *x*, *y* or *z* dimension of the collection of atoms used in the calculation. The second grid is twice as big as the first, with the same center. The potentials on the outer grid are solved for first, then interpolated and refined further on the inner grid. Potentials are then interpolated to a 65 cubed grid the same size as the outer grid. This final grid, or 'map' as it is referred to, is then used in all subsequent calculations. It may also be written out in DelPhi form. Typical calculation times are around five to six seconds.

Although there is no choice in the sizing of these grids, the user has control of the inner and outer dielectric constants, the probe radius used to determine water inaccessibility, the salt concentration and ion exclusion radius. There is no support for the nonlinear equation, for periodic boundary conditions, membrane slabs and holes, or any other DelPhi features.

Once a map is calculated it can be evaluated in several ways. Isopotential (also referred to as 'through space') contours may be calculated at any value, given any color, and displayed as solid surfaces, meshes or points. Potentials may be interpolated at any molecular surface, and at any set of atoms. (Trilinear interpolation is used throughout). The electric fields may be calculated at a set of points and represented in magnitude and direction as a three dimensional arrows. Molecular dipoles may be calculated and similarly displayed. Field lines can be calculated from a set of points, colored and displayed in 1-D or 3-D (i.e. lines or tubes). The potential may also be interpolated at a slice plane perpendicular to the Z direction (i.e. parallel with the screen). This latter display is updated as the map/molecule is moved, or alternatively as the position of the slice plane is altered.

Values at surfaces and atoms may be colored by 2 or 3 color continuous or by discrete colors, where as the Z plane may only show the former. Field vectors may have their magnitude encoded in their length. Field lines can be assigned directionality and color when calculated.

Distance Calculations

Grasp will calculate minimal distances from surfaces to surfaces, from surfaces to atoms, from atoms to surfaces and from atoms to atoms. In each of the preceding the ability to define a subset is assumed understood. In the case of atoms there is also the option to subtract the assigned van der Waals radii from the distance.

An example of a novel use of distances in Grasp is to calculate a 'depth' map, i.e. the depth of atoms from the surface to every atom. Distance maps are also useful in defining interfaces between domains, either surface-wise or atom-wise.

Maps

Grasp contains room for two internal 'maps', i.e. 65-cubed, cubic lattices. Internally generated maps are stored in the first of these arrays. Maps read in are put in the second array. Simple operations are allowed on and between maps, such as differences, sums etc. Maps can also be swapped so that both

another surface or surface subset, or from a set of atoms, and in each case are the minimum such distances. Surface curvature is as defined in Nicholls *et al* and is derived from a concept of local hydrophobicity. Briefly, each possible placement of a water "sphere" against the surface of the molecule reduces the accessibility of that water to other waters. Against a concave surface this accessibility is less than against a convex surface, and a formal correspondence can be made between contact to an arbitrarily complicated surface to contact with a sphere of a certain curvature. Curvature thus defined is a property of the accessible surface, but can be uniquely mapped to the molecular surface. When color coded it reinforces the effect of SGI lighting in that surface hollows are made distinct from surface projections, and so is useful in visualizing patterns of surface shape.

Surface data (which means vertex coordinates, connections, and normals), and any properties, calculated or assigned, of any surface or subset of surfaces can be written to a data file. These data files can be read in during program execution or at start up. Surface data can be appended to other files to make surface 'libraries'. Since subsets of surfaces can be saved one could, for instance, make a library of the surfaces of the active sites of different enzymes.

The property data of a surface or subset can also be saved as an ascii file for analysis, or for temporary storage. This file can also be read back in and hence a user generated function mapped to a particular surface.

Surfaces or subsets of surfaces can be inverted, i.e. the surface normals reversed. In this way one can look at molecules from the inside. One can also do like-with-like comparisons of two surfaces which might form complexes by inverting one surface of the pair. Surface properties can also be "inverted", for instance positive turned to negative and vice versa through the *simple math* utility.

Along with cavity surfaces (which can be thought of as a surface subset) and contour surfaces, one can calculate the area of any molecular surface or subset of a molecular surface, and similarly the volume enclosed (noting that the volume is not meaningful if the surface is not closed). The surface area for an accessible surface gives a measure which has often been associated with hydrophobicity, since it is related to the number of water molecules in contact with the molecule. Grasp also has a more accurate surface area subroutine for atom by atom accessible area.

Any surface or subset can also be attached to the rotation and translation dials alone. This creation of a formal subset, for which a unique name is assigned, can then be manipulated independently (see *formal subsets*). This allows for parts of surfaces to be removed, compared, docked etc.

Electrostatics

Grasp includes a Poisson-Boltzmann (PB) solver which is a similar but simpler version of that used by DelPhi. The fields calculated by it are for qualitative use only. For quantitative use there is full support for the output from DelPhi, in terms of the potential maps, the dielectric maps, the modified pdb files, charge files, size files, etc. Grasp does not (yet) contain an interface to DelPhi,

The first is a rendered surface (i.e. the triangles are filled in) but the lighting calculations are simplified (pseudo-lit) and done in software rather than by using the SGI hardware calls. The second is a mesh representation, i.e. the triangles are not filled in. The third is a points only representation. The default surface produced by the program upon construction can be preset in an external file (see *grasp settings*).

All displays of surfaces (and also contours, atoms and bonds) can be depth shaded. This means that the farther away a part of the surface the darker the color. More exactly, its color is interpolated to black based on the distance from the viewer. Grasp uses a dynamic depth range, where the front edge, i.e. where interpolation begins, is determined by the nearest point on the structure to the viewer, and the back edge, i.e. the depth at which the color is set to black, is a fixed distance behind this point. Depth shading makes a dramatic difference to the mesh representation, and to a lesser extent with other representations. A default depth is set for all relevant structures. However, since the optimal values tends to depend upon the size of the structure or feature under consideration, the depth can be altered by the user.

For the rendered and lit, and rendered and pseudo-lit surfaces the direction of the incident light may be varied. At present other material properties of the surface (reflectivity, transparency, etc.) are not user accessible.

Surfaces are colored by assigning colors to each vertex. This can be done according to any value associated with that vertex as described in *color coding.* Alternatively surfaces can be colored by selecting a subset and assigning a discrete color. As described in *subsetting* there are many ways of selecting a surface subset based on vertex properties, associated atom properties, or by hand with the surface scribing mode. Any surface or subset can be 'uncolored' i.e. undisplayed. Hence one can remove portions of surfaces to create "windows" into the underlying molecule. Any surface property can be interrogated using the mouse buttons, i.e. placing the mouse at any given point and clicking returns a value or values associated with the nearest surface vertex. Which property values are returned, and by which mouse button, can be set by the user.

Surface properties can be classified as those which can be used in subsetting and displayed, and those which can be used in subsetting but can not be displayed. The latter tend to be intrinsic or assigned properties and include absolute coordinates, relative position, current discrete color, surface construction number, formal subset name (see below), and vertex number. Surface properties which can be displayed are generally calculated within the program and include electrostatic potential, curvature, distance (to another surface or set of atoms), and two general property fields. The latter two can be used in a variety of ways. For instance any atom property can be mapped to the surface and stored (and hence displayed) as general property one or two. Or the user can produce a new quantity out of others by the *simple math* facility. Or properties can be imported (see below).

Surface potentials are interpolated from potential maps as described in *Electrostatics.* Distances as a surface property are calculated either from

they are red, white and blue, with red for the minimum value (typically negative), white at zero, and blue for the maximum value (typically positive). However these colors may be also set by the user.

Default maximum and minimum control values are taken as the maximum and minimum values of the property being represented (surface potential, atom distances etc), the middle value is set to zero unless the maximum and minimum are both greater than or less than zero, in which case it is set to the average of those two values. These control values can also be explicitly altered.

Continuous color maximum, minimum and middle values may also be adjusted via a mouse activated widget. This is useful in rescaling the color code to bring out particular features on the fly. Independent widgets appear for each continuous property displayed. These also offer access to other features via drop-down menus.

Surfaces

Grasp supports two types of surfaces, molecular and accessible. The molecular surface is defined as the boundary of that volume within any probe sphere (meant to represent a water molecule) of given radius sharing no volume with the hard sphere atoms which make up the molecule). The accessible surface can be defined as the locus of the centers of all possible such probes described above in contact with the hard sphere atoms. Alternatively it can be defined as the hard sphere surface if each atomic radius is increased by the probe radius. The default probe radius for each type of surface is 1.4 Ångstroms, but can be set by the user.

Everything which can be done with molecular surfaces can be done with accessible surfaces. For brevity therefore, except where differentiated, they will both be referred to as molecular surfaces, since both are derived from molecules, to distinguish them from other surfaces, such as isopotential surfaces.

The surfacing resolution, i.e the lattice spacing used to generate the surface, is determined automatically. The lattice spacing used in the process is scaled relative to the largest dimension of the molecule. Hence coarser lattices are used for larger molecules. This scaling has the advantage that the surface of very large molecules can be as easy to manipulate as that of small molecules. Though the surface of larger molecules will then be less accurate one is often interested in courser features of such molecules. Grasp does not currently support surface refinement.

If the molecule has only a few atoms this method can lead to a lattice spacing at which the method Grasp uses is inefficient, hence there is a minimum allowed lattice spacing. This parameter can be set larger by the user to *force* use of a coarser grid (as might be preferred to improve draw speed, since fewer triangles will comprise the final surface, or to overcome memory limitations).

Surfaces can be constructed for all atoms or for subsets of atoms (see *subsetting*). The process takes a few seconds at most and results in a smooth tessellation of the surface which is colored white but shaded by the SGI lighting routines. This can be quite slow on some machines as there may typically be 20,000 triangles. To enhance drawing speed there are three other draw modes.

Features

What follows is a fairly complete listing of Grasp's capabilities, in some sort of logical ordering.

General

Grasp uses a perspective based view, i.e. things farther away are smaller. To enlarge a view one simply moves it closer to the eye. Manipulations are by mouse or by dials. Molecules are mapped to a "unit box" which can be displayed with the front side removed. There are also embedded cross-hairs to remind the user of any rotations and translations they have applied. The default clipping planes are very close to the "eye" position and very far away. These can be temporarily altered via a slice control tool. The background is either black or that produced by the unit box. The default window size may be changed in the usual window resizing manner. There is also a full screen option where the entire screen is given over to the display. All functions are accessed by hierarchical menus via the right-most mouse button, or via the command line. This has the advantage of leaving the Grasp field of view completely uncluttered. Commands may also be read in from an external "script" file. All display objects (surfaces, atoms etc.) are independent, i.e. they can all appear in view at the same time or can be individually hidden from view.

Color Coding

Grasp supports two different modes of color coding. The first mode, 3 color continuous, requires three numerical values, called "control" values, along with three colors, one color per value (colors are defined by RGB triplets, or by an index into a list of colors, see *color*). Color coding is then implemented as follows:

If a number is less than the minimum of the three control values it is assigned the color associated with that minimum value. If it is greater than the maximum value it is assigned that value's color. If it is between the minimum and middle values the assigned color is found by linearly interpolating between the minimum and middle colors, and if it is between the middle and maximum values by linearly interpolating between the middle and maximum colors. By linearly interpolating is meant the following: Colors are made up of red, green and blue components, each component having a strength of 0.0 to 1.0. If a number is, for example, halfway between one value and another, then its interpolated color is similarly halfway between the two colors assigned to those values, i.e. its red, blue and green components are half those of one color and half of the other.. Grasp also supports 2 color continuous which is equivalent to 3 color continuous but with the middle value and color set to be the same as that of the minimum value and color.

The second method is zonal coloring, wherein a certain range of values is assigned a certain color. The color boundaries between different zones are sharp. This is also referred to as discrete coloring.

Default colors are provided for all properties, e.g. for electrostatic potential

Grasp, that surfaces and atoms should be treated with equal importance.

There was still a need for other electrostatic representations, such as isopotential contours, projection planes, field lines etc. Also, since typical use of the program was visualization of large molecules with thousands of atoms I devised a simple representation for those atoms that was fast to display and could be colored by property. This led to other representations of atoms and groups of atoms.

The program grew beyond my initial plans. Hopefully, however, the program maintains a coherent philosophy. For instance the use of surfaces both for displaying properties and as objects in their own right, the visualization of electrostatic properties, and more lately the generalization of the idea of an object representing both a set of atoms (as the surface does) and a property (such as electrostatic potential). An example of the latter is the DNA representational project of Rex Bharadwaj. Here DNA bases can be represented as elongated boxes whose width can represent a property associated with that base, such as base twisting, sliding or rolling.

The program has achieved most of the goals I had concerning the development of these ideas. Of course in their actualization they have spawned many more. But hopefully the present version is at least complete enough to be useful.

Comments as always are much appreciated.

Anthony Nicholls
October 1992

Introduction:

Grasp came about because I wanted to visualize electrostatic potentials at surfaces, in particular the surface of biological molecules. Barry Honig's lab is well known for the program DelPhi which calculates electrostatic potentials from the Poisson-Boltzmann equation, and has the led the way in applying this equation in structural biology. The program can be used to get productive quantitative numbers for a variety of biochemical phenomena but qualitative visualization had been limited to isopotential contouring, typically with a program such as Insight from Biosym Technologies. The limitation of such an approach is that such contours do not capture local topology or shape. They often extend significant distances away from the molecule while one would expect most of the 'action' to be close to the molecule, in fact at the surface of molecules. So I decided it was time to attempt a graphics program which emphasis surfaces and electrostatics.

I would not have been so confident in starting this project had it not been for considerable groundwork laid by others, in particular Kim Sharp. He and Mike Gilson had devised a novel algorithm some years ago for calculating the molecular volume (i.e. water inaccessible volume) using a cubic lattice. This method, though not efficient when the lattice spacing is small relative to atomic radii, can be made rapid at lower resolution. Given the molecular volume, Kim had also reinvented a technique commonly known as the "Marching Cubes" algorithm to produce a surface tessellation. Putting these together in an optimized form produced a rough and ready surface description which could be easily visualized on a Silicon Graphics Iris (SGI) computer. The potentials at surface points could then be interpolated from the 3-D map produced by Delphi and color coded to indicate the result.

The initial results were surprisingly good, both in terms of aesthetics and usefulness. The large difference in dielectric between water and the interior of proteins modelled by DelPhi means that local electrostatic effects can dominate global ones. So, for instance, an active site can be negative even when the protein total net charge is very positive. This is seldom seen if Coulomb's Law is used to calculate potentials because of the long range nature of (1/r). Grasp was immediately able to display this consequence of electrostatic screening, for instance showing the deeply negative binding site of the catalytic magnesium of RNAse H ,crystallized without that ion by Yang and Hendrickson.

So it was clear that this approach held some merit. The combination of surface shape and electrostatic potential was synergistic. Moreover I began to see that just having a rapid surfacing and visualization algorithm was useful. For instance it was simple using surface connectivity to display only the internal cavities of proteins. Here the initial use was the bacteriorhodopsin structure of Henderson which has numerous "holes" surrounding the retinal moiety.

It was also instructive to "project" properties of the underlying atoms onto the surface and color code them. An example being the B values normally accompanying crystallographic structures. So I began to see the surface itself as useful construct, regardless of electrostatics. This became a central tenet of

GRASP: Graphical Representation and Analysis of Surface Properties