Journal of Computer-Aided Molecular Design **18:** 225–234, 2004.

Short communication

A Java applet for multiple linked visualization of protein structure and sequence

Thomas J. Oldfield
EMBL Outstation – Hinxton, European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK (Fax: ++1223-494468, e-mail: oldfield@ebi.ac.uk)

Received 12 March 2004; accepted in revised form 16 June 2004

Key words: applet, Java, molecular, protein, sequence, structure, viewer, visualization, web-based graphics

Summary

The amount of biological data available from experimental techniques is huge, and rapidly expanding. The ability to make sense of this vast amount of data requires that we make correlations between distinct biological disciplines using visualization techniques to highlight the critical information. This article describes the visualization techniques of dynamic data brushing, view context maintenance, fisheye sequence view, and a magic lens that have been developed to display protein structure and sequence information.

Abbreviations: AV – AstexViewer^{TM}, AV-MSD – AstexViewer^{TM}@MSD-EBI, MSD – Macromolecular Structure Database, EBI – European Bioinformatics Institute, SSM – Secondary Structure Matching program, 1D – One dimensional, 3D – Three dimensional.

Introduction

Visualization is the technique of representing abstract data to aid in the understanding of the underlying meaning of the data [1,2]. This is different from, although makes use of, methods for rendering geometry. For example, it is useful to render a protein Cα trace as solid cylinders with depth-cueing and Z-buffering so as to visualize the fold of the protein. This article discusses visualization techniques to aid in the understanding of macromolecular structure and sequence within the graphical applet AstexViewer^{TM}@MSD-EBI (AV-MSD), while the algorithms and methods used to render the coordinate data in this viewer are described in [3].

The problem with the visualization of biological data is the presentation of information 'in context' for a scientist, and limiting information that is out of context. Since the biological sciences cover a vast diversity of ideas, it is common that experts in one field will have a poor grasp of other details; for example, a sequence biologist may have little knowledge of 3D structure or chemistry. Scientists from each particular discipline therefore have little common overlapping terminology to express concepts and so visualizing ideas that link information together represents a critical part of bioinformatics.

A number of programs have been developed [4–10] with the aim of providing web-enabled graphics suitable for molecular structure, but in most cases these programs require the installation of the programs on the client side computer before they can be used. This means the programs have to be ported and distributed for different computer architectures, and require effort from a user to install and maintain the installation. The AV-MSD was written as a Java^{TM} [11] applet as an integral part of the service provided by the macromolecular structure database (MSD) group (http://www.ebi.ac.uk/msd). The MSD provides a service to the scientific community as a deposition site for protein structural data (in conjunction with the RSCB: http://www.rcsb.org/pdb/ and PDBJ: http://www.pdbj.org/), followed by collation and validation of this data. The struc-

ture data is stored within an ORACLE® database in a data schema (http://www.ebi.ac.uk/msd-srv/docs/dbdoc/) with search and visualization methods to return macromolecular structure information to the user in context with sequence and textual information. The AV-MSD viewer was designed to be a high-performance rendering program with novel visualization ideas used to present a range of data from this database.

Background

The visualization techniques described here have been implemented within the applet AV-MSD, which builds on the functionality within the AstexViewer™ [3] from Astex technology, which is a macromolecular coordinate display applet. The combined applet is a client-side drawing program for structure, sequence and graphs. Its design as a web-based applet therefore necessitated a number of considerations:

1. Portability: the ability to run on legacy browsers (Netscape 4.7 and Internet Explorer 3) without upgrade, and of course work efficiently on the latest systems.
2. Small download size and run time size: To optimize the download times and provide the ability to run efficiently on most computers.
3. Provide the performance normally associated with hardware-accelerated graphics.

The viewer is written in Java™ 1.1 compatible source-code, which solves many problems of portability as this legacy version of Java™ is supported without modification on many browsers and computers. Older versions of Java™ have a smaller runtime memory footprint due to the limited features within the earlier versions, making it more suitable on machines with limited physical memory (run-time physical memory usage: 30 MB for Java™ 1.18 compared to 80 MB for Java™ 1.41 running on an IBM PC running Red Hat Linux version 7.3). It is also typical that many computers are maintained by a computer administration and cannot be upgraded by a user without significant delay and bureaucracy. The AV-MSD will run faster when compiled with more recent versions of Java™ where sufficient memory is available, but the portability issue is considered to be overriding for the service offered by the MSD.

A small applet size (240 kB) is required because the viewer must be downloaded to the client machine when it is first used, or after the browser cache has been purged. Typically though, most academic users will have direct access to a high bandwidth network, but efficient design of code was considered important. The entire design principle of the viewer was based on the generalization of visualization techniques within the applet, which represents reuse of code on a global scale. This means each visualization technique represents a method that can be used to show many different properties, and this also means that the AV-MSD can be reused for different services that show different data; just the control file is changed.

The performance of the applet is most critical with the 3D coordinate viewer due to the matrix multiplication and subsequent rendering of many objects on the view. The structure viewer will refresh at 45 frames per second 'running on a 1GHz Pentium III computer for a 1000 atom protein with electron density map' [3]. The 1D sequence is computationally simpler to manipulate than the structure viewer. The sequence viewer will refresh at 20 frames per second running on a 1 GHz Pentium III computer for 50 sequences of average length of 350 residues with total aligned length of 1030 residues (with insertions) using the hyperbolic distortion. The refresh rate is approximately three times as fast when displayed without hyperbolic distortion.

The AV-MSD consists of four data views and a control palette. The data views are used to show: macromolecular structure coordinates (*structure)*, amino acid and nucleic acid sequence data (*sequence*), general graphs of data (*graphs*) and macromolecular structure data hierarchy (*hierarchy*). Each of the views has the same data root that is the macromolecular structure and sequence with an interface layer between the two objects of structure and sequence. The interface between structure and sequence is required due to a number of inconsistencies:

(1) Multiple aligned sequences have insertion gaps within each sequence which are not reflected within the structure data.

(2) The structure model has a property of atom, residue, chain and assembly [12]; the sequence has property of residue and chain.

(3) Structure alignment is performed by chain, but shown by assembly, and the sequence is aligned by chain and shown by chain.

Algorithm and implementation

The ability to display information in different ways allows users with different backgrounds to comprehend it. Presentational data is held as attributes which may be sequence dependent (a function of one-dimensional sequence) or structure dependent (spatial relationship in three dimensions). The *structure* view can display presence/absence of atoms, discrete colour, colour-scale, text labels, stick models, atomic spheres and ribbon in conjunction with a magic lens, and bond thickness. The *sequence* display has the property of visibility, discrete colour, colour-scales, colour-inverse, point marks and join lines as well as additional text strings such as secondary structure. The *graphs* view shows data with symbols, lines, histograms, isometric surfaces, 3D histograms, contouring, tiles, fixed/variable radius pie charts, dendrograms, highlighting, visibility and text labels. The *hierarchy* view of data can be annotated with colour and black/white highlighting, and can fold up data to form a compact view. All these attributes can be controlled by the user interface or using server side designations. Since many of these ideas are found within other graphical programs, this article concentrates on the more novel aspects of data visualization, most of which have been used outside the field of bioinformatics (see Leung and Apperley [13] for a review).

Difference and similarity

One of the major design goals of the MSD service was the ability to return and present superposed structure and aligned sequence where multiple results are obtained from a database query. To maximize the information content of any query result, the structures and sequence are automatically superposed during the time delay between the search result being presented to the user and the action of clicking the button to show the results. The process of automated superposition is performed by FASTA [14] and SSM [15]. The AV-MSD viewer can read pre-aligned structures or can pre-multiply structure/chains coordinates using matrices supplied within a control file. It also reads FASTA or ClustalW [16] sequence alignment files for the sequence superposition display. Since the client side resources available to the viewer may be limited, the viewer will handle any number of aligned sequence and structure-references, but only actually load eight sets of structure coordinates at any one time. The user can load a structure by clicking a reference to it within the sequence view, which also closes and frees the memory to the 'oldest' loaded structure.

All loaded aligned structures can be coloured by a red-blue hue gradient based on the nearest distance to an equivalent atom, or the number of equivalent atoms closer than 1.5 Å. Join lines can also be added to highlight atoms that are close within each superposed structure. Sequence consensus alignment is shown with the same red-blue hue gradient and a normalized alignment score is provided. Both the normalized consensus alignment score and colour scheme are interactively updated during sequence editing.

View swapping

If we are to show the structure biologist the power of sequence interpretation and the sequence biologist the power of structure interpretation, it is important to show each annotation type within context. That is, to colour the sequence using the structure alignment, colour the structure by the sequence alignment and to place annotation marks derived from other contexts. This includes the marking of sequence with active site residues determined by structure and the labeling of the structure with ProSite functional assignment determined from sequence [17]. This functionality is implemented within the AV-MSD by allowing attributes to be moved from one data model to the other. In this way it is possible, for example, to colour the Ramachandran plot [18] using the sequence consensus alignment.

View context and traversal

Macromolecular structures and sequences are large complex data objects that require view manipulation to study different data aspects and concentrate on different areas of interest. It is therefore typical for a macromolecular display program to be able to manipulate the views of *structure* and *sequence*; either with a virtual track ball, or using real/virtual control dials. The AV-MSD has the ability to show preset views, which requires less effort to find detail, and also has the ability to fly the sequence and structure display between different views. The view context is defined as the extent of the information shown within a view, and may consist of the entire structure/sequence, a small magnified region of these or

a distorted representation with differential magnification of data. The AV-MSD *structure* view can show an entire molecule or a zoomed view to present a small number of residues and their interaction. The *sequence* view has distortion to simultaneously show the entire sequence and the detail within a region. A single data pick in any view (or palette) will change the display in each data view to centre and magnify/mark this data point. Traversing these large views of the data is problematic, particularly for scientists with specialties outside structure or sequence. The general action in most graphical programs is to 'snap' to this new view context where the structure/sequence display update is made immediately; this results in loss of relational context between the before and after positions. The AV-MSD viewer therefore has a *fly* mode, where any change of view context is made by animation over a period of one second. The sequence view is slid from the original position to the new position by translation of the sequence display. The structure view has 2 different fly modes. If the new residue position is contained within the current view, but is off-centre, then the view is translated progressively from the old to the new position in 3D. If the new point of interest is off-screen then the structure view is zoomed out to whole-molecule view, then back to the new position at the magnified focus residue. The view flying is performed using separate Java™ threads which do not block the action of the applet use, and the sequence and structure animations do not interfere with each other. Residue highlighting is used to maintain the context view for the old and new positions in the *structure* and *sequence* views. If the displayed molecule is picked with a <SHIFT> click action then the view is snapped to the picked position.

Animated highlights

One of the strongest visual perceptions for humans is that of motion. Within computer programs motion is simulated using animation where an object display parameter is updated either with or without user request. In the latter case this animation must have no effect on any other control feature of the applet, which means that it must be run as a separate thread. The AV-MSD can display multiple animated graphical objects that are synchronized in time. A single synchronized object is defined as containing any number of child objects which are members of the synchronized object; they start and stop as a function of time relative to the start of the parent object. A single synchronized object can therefore show a progression of events and provide the means to display time within each data view. Each child object has properties of display on/off, rotation, translation, colour or shape. A status is defined by an on and off time within the repeat cycle of the entire synchronized object. These child objects also have properties of mouse cursor enter and mouse button click to detect when the user has 'touched' an object. The design of the animation is similar to a VRML flip-book, although simplified and designed to work within the minimal specification of Java™ 1.1. Display methods so far implemented are:

(1) Static spheres for the labeling of atoms with text. Placing the mouse over the static sphere updates the applet status bar with the text string. The child object consists of a static sphere that has an *on* time of zero and an *off* time greater than the animation object repeat time and with mouse motion detection. The text object is held with the atom attribute data and linked by a structure-reference.

(2) Rotating spheres to label atoms with a URL. Placing the mouse over the rotating sphere shows the URL within the applet status bar, and clicking the atom opens a browser window at the specified URL. This allows the inclusion of large amounts of text/information as a label, and can be used for a description of a functional site. The child object consists of a rotating sphere that has an *on* time of zero and an *off* time greater than the animation object repeat time; and with mouse motion and mouse pressed detection. The URL object is held with the atom attribute data and linked by a structure-reference.

(3) The visual display of electron motion within a reaction mechanism is shown by the progressive turning on and off of spheres for the atoms within an active site. A number of child objects are defined, each with different *on* time and *off* times.

This animation method is very simple but highly effective as a means to highlight a huge range of different information, in particular time dependent processes.

Hyperbolic display of sequence data

Most data within biology is extensive; in the case of the protein sequence the information is encoded within a very long polymer which is normally shown as a long series of characters. It is therefore not possible

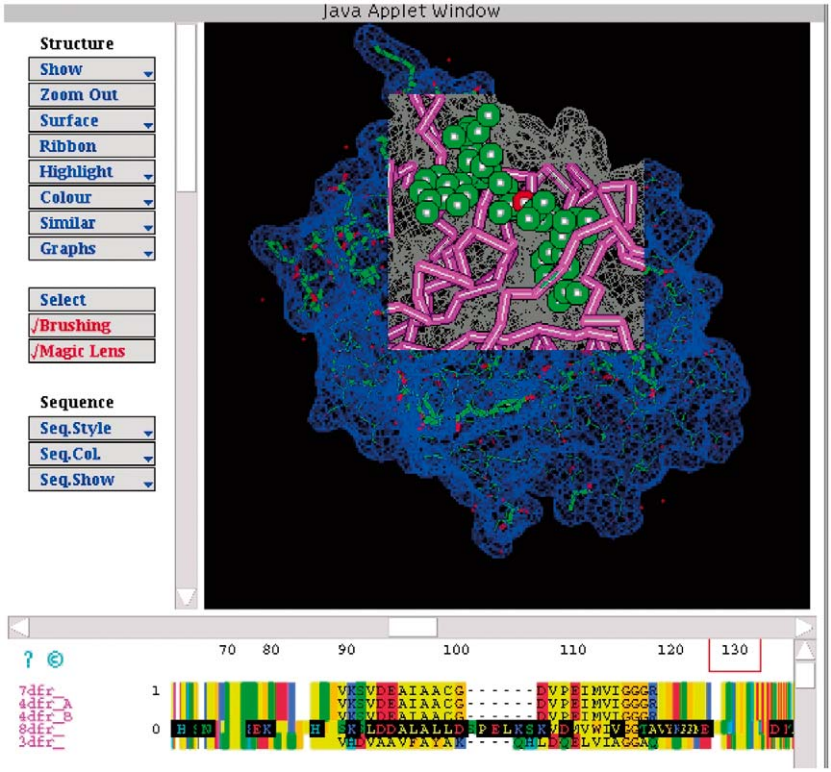

Figure 1. The structure and sequence viewer showing a hyperbolic sequence view with highlights for residues in an α-helical conformation, structure thick-bonding for the same residues, a molecular surface and the magic lens containing solid ribbon backbone trace and atomic spheres for ligand atoms. The ribbon trace is generated by cylindrical solid object rendering from the protein Cα coordinate positions with foreground highlighting and the atomic spheres are generated using spherical solid object rendering with foreground highlighting for ligand atoms.

to view the entire sequence and read the characters at the same time, particularly when viewing multiple sequence alignment. It is usual to either show the entire sequence using a simplified representation, view a small part within a window, or wrap the sequence onto multiple lines which can be particularly confusing for aligned sequences. The AV-MSD gets round this problem by the implementation of hyperbolic distortion or 'fisheye view' [19, 20] so that the data is folded so as to appear to lie on a U shaped 3D plane that extends to negative Z for large and small X (Figure 1):

$$\mathrm{X(display)} = \left(\mathrm{Tanh}\left(\frac{2 * (\mathrm{x} - \mathrm{x0})}{(\mathrm{xN} - \mathrm{x0})}\right) - 1\right) * (\mathrm{xN} - \mathrm{x0}).$$

This distorted view of the sequence means that the central region between $\frac{1}{4}$ and $\frac{3}{4}$ of the view panel is resolved at full resolution while the periphery of sequence for $\mathrm{X} < \frac{1}{4}$ and $\mathrm{X} > \frac{3}{4}$ appears compressed and only drawn with the colour attribute. Scrolling the sequence view has the appearance of it sliding around a U shaped plane so that the whole sequence is always visible and clickable.

The advantage of this visualization method is that the entire length is always observable and pickable, and the colouring attribute (for example residue polarity) can be observed for the whole sequence. This includes active sites and disulphide bond labels. Aligned sequences show up as continuous stripes at the periphery of the view. The central $\frac{1}{2}$ region of the view is shown at full resolution, allowing detailed analysis. This mapping of the sequence is clear to the user since the perception is similar to that of the human view of the environment. A small central region of the eye has a high resolving power while the periphery of the human vision has lower resolution. This view of the sequence data is therefore probably more natural to the user than the linear display. The disadvantages are technical, as display update and picking is computationally intensive.

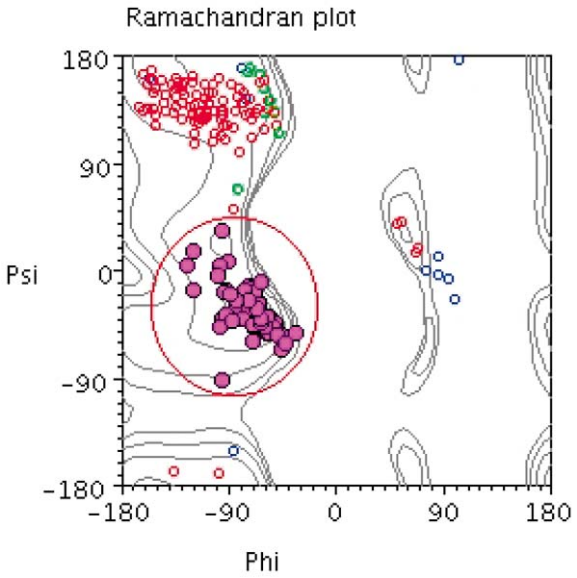

Figure 2. The Ramachandran plot of 7dfr shows a highlighted region that corresponds to residues that have an α-helical conformation. The brushed points within the red ellipse correspond to those highlighted in Figure 1.

Data picking, brushing and selection

The ability to select and label data points within the structure, sequence and any derived set of graphical data is imperative for a graphical program. The design architecture of the AV-MSD allows any number of different data views to interact in real time. Data selection involves the interactive capture of data points in each view based on mouse motion and clicking. This requires that any pick of a data point in one view has a connection to that point in all other views. If a number of points within a single view are selected, then all of these points are highlighted within all the views. This affect, when made interactive (updates rapidly as the mouse cursor is moved) is known as data brushing.

Figure 2 shows a view of the Ramachandran plot, where an ellipse has delimited an area which represents a helical backbone conformation in proteins. All the data points within this view have been highlighted showing the current selection. The corresponding data in the other views are also highlighted (Figure 1) and therefore marking helix property within the sequence and structure. This process is essentially very simple, but requires that all data (structure, sequence and derived views) within the applet are connected via a common data object, and that a rapid method of selecting data and highlighting is available.

The AV-MSD architecture has some similarity to the Model-View-Controller design [21] that consists of a model that maintains data, views for displaying the data and controllers for handling the events that affect the model or view. The AV-MSD has two models, one for the 3D structure data and a second for the 1D sequence data. There are also separate views for the structure data, the sequence data, and for each derived graph. The AV-MSD has a single controller that handles event detection (listener) that detects mouse-press (including tool buttons), mouse-move, key-press, window and scrollbar events from all the data views via a Java^{TM} listener interface. Any event is therefore captured within a single place and can therefore be handled efficiently in terms of computation and code. The data points are determined from the models, and highlights marked on all the views as required. Additional views are easily added; for example, a server supplied set of data can be provided as attributes and a new graph view is added automatically with additional menu buttons to control this view.

This MCV design is put to good use with the brush action where data is selected by a cursor area, and all equivalent points in all views are highlighted. The size of the data brush can be adjusted in each view independently using a <SHIFT> drag action of the mouse to stretch the bounding region. The highlighted views can also be locked so that all the current highlights remain active until the cursor returns to the view that generated the selection. In this way it is possible to inspect, for example, the *sequence* and *structure* views for marked helices after selecting these with the Ramachandran plot.

The implementation of brushing allows the highlighting of many aspects of structure and sequence without explicit coding. It can also be seen that this provides a myriad of views without creating many additional control buttons that results in a complicated graphical user interface.

The AV-MSD views have separate properties of highlight and selection. The highlight list is designed to show features automatically and interactively on all views while the selection list allows a user to define their own data selection and then apply their own highlighting. Selected data points are generated from the *hierarchy* data view by picking objects in this view, and the selected data marked on the other views; any subsequent annotation (such as colour change) is only applied to this subset. When no selection is active then the change is applied to all the data in a view.

The following describes the action of the selection and highlighting with respect to each view.

Structure

The selection of data from the atom coordinates is detected as single atom-pick using the AV interface and this returns a single 'atom' object. Brushing uses a circular area in the XY screen plane, which therefore defines a cylinder of length defined by the Z clipping planes in 3D. Display of highlighted data within the *structure* view is by wide bonds when both atoms connected by the bond are contained within the highlight list. Selected data is shown using yellow cubes at each atom.

Sequence

The selection of *sequence* is defined using a mouse click of the sequence data or any associated label within the sequence window. The brushing is defined using a rectangular area that covers a number of sequences in the y-ordinate and multiple residues within the x-ordinate. The rectangular area is distorted by the hyperbolic distortion. The highlighted residue list is indicated by display inversion of the sequence character. A normal residue is drawn as a black character on a background coloured by attribute such as residue polarity; highlighted residues use a coloured character on a black background. Selection is shown as magenta boxes superposed on the sequence.

Graph

A *graph* is a view of attribute data; four attributes are predefined within the applet, and any number of server side attribute data can be supplied. The graph data can be plotted as a function of residue (for example: Ramachandran plot), or as a function of atom (for example: B-value). Picking a data point on a *graph* can therefore represent a single atom or a residue and these picks will therefore return selection objects to either residue or atom object level. Brush selection is defined using an ellipse of adjustable size. Highlighted data is displayed on a *graph* using filled symbols, and a text string is used to identify a picked data point.

Hierarchy

The *hierarchy* view displays data using an expandable tree of objects where each level of hierarchy can be hidden or shown by clicking an object in the view. Picking this view is used to select objects and these selections are propagated to *structure* and *sequence* as selection objects; brushing creates highlighted objects. This *hierarchy* view shows selection using a colour change of the label text from blue to red, and brush highlighting as background colour change from white to gray. If the selected point is for an atom, then this single datum is returned, if the selected point is a chain then all the atoms within the chain are returned. A range is created by a <SHIFT> mouse click, and all objects from the last pick to the current pick are selected.

Lensing: The use of a magic window to show hidden information

A major issue with protein structure information is that the bonded structure of a protein gives adequate information to the expert structural biologist but contains no visual clues to a scientist not used to viewing three-dimensional representations of macromolecules. A 'cartoon' representation is therefore useful to display the overall fold and ligands but unfortunately the functional chemistry is lost with this reduced representation, as we no longer have the atomic positions. Addition of a molecular surface exacerbates the loss of detail. A method to address this problem is the movable 'magic lens' originating at Xerox PARC [22,23], in which alternative views of the macromolecular structure can be displayed. The normal display of macromolecular structure consists of bonds, ribbons and surfaces and is defined here as the *standard* view. An *overlay* is defined as a display contained within the lensing region and shows cartoon representations of macromolecules. Since the AV uses a software implementation of bit planes to render the image, the lensing is implemented by direct manipulation of these bit planes. The lens window is defined as a 2-dimensional bounded window smaller than the structure view panel and is tied to the motion of the mouse pointer. Within the boundary of this window any pixel in the *standard* view, not in the background colour, is set to gray. The *overlay* view is rendered from attributes (for example depth-cued, Z-buffered solid ribbon and atomic spheres) for the entire structure view, but only shown within the lens window. The result is a display window where the *standard* view is washed out while additional information is superposed in the *overlay*. This visualization technique works well because the human perception is able to 'continue' the missing informa-

tion from the gray image but allows stylized objects to identify key information. The effect used within the magic window can be controlled from the server, so this lens can be tuned to different uses.

Window generation

The AV normally redraws the molecule display from the current atom positions by first drawing these on a bit plane followed by a *double buffer* swap to present this bit plane. Turning on the lens consists of:

(1) Copying the entire current screen pixel buffer into an overlay buffer.
(2) Setting any pixel, not in a background colour, to gray within the overlay.
(3) Rendering the entire overlay view attributes onto overlay buffer.

The movable window is defined as an adjustable 2D box that extends about the mouse cursor position; it is viewed as a square box that follows the mouse cursor position. As the mouse cursor position is moved, the two buffers are rendered inside and outside the movable window followed by a buffer swap. To improve the performance of this rendering process it should be noted that any movement of the 2D box results in changes to the display only at the edges of the 2D box. Consider the movement of the window from a screen position X=0, Y=0 to a position X=1, Y=1 where the window size is 400 by 400 pixels; it is immediately obvious that a change in the position does not require the update of the entire window area. In fact, the changed region is only 1 pixel wide, and this requires the copy of just four lines of pixels that surround the lens window. It can be seen that the window movement requires very little computational overhead. First, the molecule is not rendered from the atom positions, the repaint only consists of a redraw of the bit plane, and second it requires only setting/unsetting a small number of pixels. This high-speed refresh results in a very responsive visualization technique attached to the motion of the mouse cursor. It is necessary to recalculate the overlay view whenever there is a change to the viewpoint or annotation of the structure view. This calculation is therefore performed when the cursor enters the *structure* view and also after manipulating the 3D *structure* view with the virtual track ball. This is implemented as *on* when a 'window entered' event occurs or when a 'mouse released' event occurs after a 'mouse dragged' event. The lensing is turned *off* when a 'window exited' event occurs or a 'mouse pressed' event is followed by a 'mouse dragged' event.

Figure 1 shows an example of this type of display for the structure of dihydrofolate reductase (7dfr), where the standard display consists of a bonded representation and molecular surface. The lens overlay consists of a solid ribbon and CPK representation for the ligands.

Solid object rendering

Java 1.1 does not have hardware convenience methods to enable the rendering of a 3D solid object so these must be implemented in software. The result is a performance hit compared to other programs; although this does not matter if a user interface can be designed to give the appearance of an interactive display. Bit planes for the structure viewer were designed and implemented within the AV [3], and separate double-buffered graphics were implemented for each of the other views of the AV-MSD. A convenience method was provided by Hartshorn [3] to allow direct manipulation and control of the AV graphics and bits were manipulated directly to render the 3D solid models onto the back buffer. A simple approximation to solid model rendering was implemented in the AV-MSD to draw cylinders and spheres for the display. Objects are drawn as spheres and cylinders after sorting on Z-ordinate, and each was rendered in ascending order to produce an approximation to hidden surface removal. Spheres are drawn as concentric circles using colour changes to give a front lighting model, while changes in hue were used to provide a depth-cued effect. Cylinder ends are drawn first using circles followed by the cylinder body as lines generated by the standard Bresenham algorithm [24] with colour change to give a front lighting model and hue change to provide a depth-cued effect. On completion of the 3D rendering, the buffer is swapped to make the image visible without flicker. This quality of solid models is low and is certainly not suitable for publication. This algorithm represents a compromise between visualization of information within the constraints of Java™ 1.1 to give portable software of small download size.

Tree view menus

A recurring problem with graphical programs with many features is the complexity of the interface. There are too many buttons and drop downs. Therefore an alternative user interface is provided which consists of a tree view of menu items using the same computer code

as that used for the *hierarchy* view. The advantage to the user is that any tool in constant use can remain fully unfolded, while those tools of little use to a particular user can be folded up and take less space within the palette. The default viewer uses the button palette (Figure 1) with a 'tree-view' button that converts the control palette to this hierarchy tree of tools. There is little code overhead with providing the user with both types of interface due to code and algorithm reuse.

Discussion

This article has described the AV-MSD viewer for the display of structure, sequence and graphs using a number of visualization techniques that has been implemented as part of the MSD services. The MSD services, although based on macromolecular structure, allow a wider audience of biologists and chemists to recognize the information contained within protein structure by reference to links to sequence (UniProt) and chemistry data (MSDChem: http://www.ebi.ac.uk/msd-srv/msdchem). It is therefore imperative that visualization techniques are applied that allow the appreciation of this information to the scientific non-expert of protein structure. The AV-MSD provides a number of visualization techniques new to the field of macromolecular graphics and provides these in a general and extensible system using MVC type architecture. These visualization techniques include a lens to overlay additional structure annotation in a dynamic way, hyperbolic distortion of sequence data and dynamic brushing of data.

The AV-MSD does not currently generate quality graphics suitable for publication and its performance is dependent on the Java™ implementation and has been found to work best using Internet Explorer 5 on a PC running Windows 2000 [3]. The design goal of the AV-MSD is the display of multi-discipline data for a broad range of scientists, so it does not contain advanced features specific to a single scientific field (for example, 3D stereo graphics, coordinate modeling).

Availability

The use of the AV-MSD viewer from the MSD service is unrestricted. A tutorial that uses the AV-MSD is provided (http://www.ebi.ac.uk/msd/education/Tutorial.html) as part of the education pages at the MSD while new features not yet used within an MSD service can be sampled: http://www.ebi.ac.uk/oldfield/AV-MSD/index.html. Online documentation is viewed in a browser window with a <SHIFT> click of a tool button, while a tool hint is shown in the status line of the applet when the cursor remains stationary over a button for more than 0.6 seconds. We have observed a rapid escalation of users of the MSD services since the first release in February 2003, including 1000's of page requests for this viewer. Third party use of the AV-MSD as part of a web service is possible under license from Astex Technology and EMBL at no cost and at the discretion of both Astex Technology and EMBL. The viewer is provided as an obfuscated jar file with documentation to allow setting up a third party service. To date the AV-MSD is used by Expasy service and the Weissman institute to display structure and sequence information from their own public web resources.

Acknowledgements

This work was funded by the European Commission as the TEMBLOR, contract-no. QLRI-CT-2001-00015 under the RTD programme 'Quality of Life and Management of Living Resources'. The author would like to thank Astex Technology for providing the AstexViewer™ under license and particularly Mike Hartshorn for modifications that allowed the development of the AstexViewer™@MSD-EBI and whose AV viewer made the development of the AV-MSD possible. The author would also like to thank Professor Bob Spence for discussions on visualization techniques.

References

1. HyperDictionary: ©2000-2003 WEBNOX Corporation (http://www.hyperdictionary.com)
2. Webopedia: ©2003 Jupitermedia Corporation (http://www.webopedia.com)
3. Hartshorn, M.J., J. Comput.-Aided Mol. Design, 16(12) (2002) 871.
4. Sayle, R.A. and Milner-White, E.J., TIBS, 20(9) (1995) 374.
5. MDL Information Systems, Inc., San Leandro, CA, USA.
6. Accelrys Inc., San Diego, CA, USA.
7. PDBViewer: Guex, P.C. and Peitsch, M.C., Protein Data Bank Quart. Newsl., 77 (1996) 7.
8. PyMol: DeLano, W.L. (2002) DeLano Scientific, San Carlos, CA, USA. http://www.pymol.org
9. JavaMage: http://kinemage.biochem.duke.edu/software/javamage.php

10. MolMol: Koradi, R., Billeter, M. and Wüthrich, K., J. Mol. Graphics, 14 (1996) 51.
11. Sun Microsystems, Palo Alto, CA, USA (http://www.Javasoft.com)
12. Henrick, K. and Thorton J., TIBS, 23(9) (1998) 358.
13. Leung, Y.K. and Apperley, M.D. ACM Trans. Computer-Human Interaction, 1 (1994) 126.
14. Pearson, W.R. and Lipman, D.J., Proc. Natl. Acad. Sci. USA, 85 (1988) 2444.
15. Krissinel, E. and Henrick, K., Acta Crystallogr., D. In Press.
16. Higgins, D., Thompson, J., Gibson, T., Thompson, J.D., Higgins, D.G. and Gibson, T.J., Nucleic Acids Res., 22 (1994) 4673.
17. Sigrist, C.J.A., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., Bairoch, A. and Bucher, P., Brief Bioinform., 3 (2002) 265.
18. Ramachandran, G.N. and Sasiskharan, V., Adv. Protein Chem., 23 (1968) 283.
19. Furnas, G.W., Generalized fisheye views. In: Human Factors in Computing Systems, CHI'86 Conference Proceedings, Boston, April 13–17, 1986, pp. 16–23.
20. Robertson, G.G., Mackinlay, J.D. and Card, S.K. 1991. The perspective wall: Detail and context smoothly integrated. CHI '91 Conference Proceedings, pp. 174–179.
21. Reenskaug, T.M.H. (1979) Xerox PARC white paper.
22. Fishkin, K. and Stone, M.C. (Magic lens) SIGCHI 1995, pp. 415–420.
23. Stone, M.C., Fishkin, K. and Bier, E.A. (1994) CHI '94 Human factor in computing systems 306–312, Boston, MA, USA. ACM Press, New York, USA.
24. Bresenham J.E., IBM System J., 4 (1965) 25.