UCSF Chimera—A Visualization System for Exploratory Research and Analysis

ERIC F. PETTERSEN, THOMAS D. GODDARD, CONRAD C. HUANG, GREGORY S. COUCH, DANIEL M. GREENBLATT, ELAINE C. MENG, THOMAS E. FERRIN
Computer Graphics Laboratory, Department of Pharmaceutical Chemistry, University of California, 600 16th Street, San Francisco, California 94143-2240

Received 24 February 2004; Accepted 6 May 2004
DOI 10.1002/jcc.20084
Published online in Wiley InterScience (www.interscience.wiley.com).

Abstract: The design, implementation, and capabilities of an extensible visualization system, UCSF Chimera, are discussed. Chimera is segmented into a *core* that provides basic services and visualization, and *extensions* that provide most higher level functionality. This architecture ensures that the extension mechanism satisfies the demands of outside developers who wish to incorporate new features. Two unusual extensions are presented: Multiscale, which adds the ability to visualize large-scale molecular assemblies such as viral coats, and Collaboratory, which allows researchers to share a Chimera session interactively despite being at separate locales. Other extensions include Multalign Viewer, for showing multiple sequence alignments and associated structures; ViewDock, for screening docked ligand orientations; Movie, for replaying molecular dynamics trajectories; and Volume Viewer, for display and analysis of volumetric data. A discussion of the usage of Chimera in real-world situations is given, along with anticipated future directions. Chimera includes full user documentation, is free to academic and nonprofit users, and is available for Microsoft Windows, Linux, Apple Mac OS X, SGI IRIX, and HP Tru64 Unix from http://www.cgl.ucsf.edu/chimera/.

© 2004 Wiley Periodicals, Inc. J Comput Chem 25: 1605–1612, 2004

Key words: molecular graphics; extensibility; visualization; multiscale modeling; sequence alignment

Introduction

Since its inception, the UCSF Computer Graphics Laboratory has worked on molecular visualization systems to meet the needs of researchers in the field, beginning with MMS/MIDS[1] in 1976 to our current offering, UCSF Chimera[2] (henceforth "Chimera"). Chimera's immediate predecessor, the Midas/MidasPlus system[3] (henceforth "Midas"), provided us with the insight that extensibility should be considered critically important in the design of a visualization system.

Midas was a highly successful molecular graphics system. However, it was relatively difficult for users to add new functionality. The first extension mechanism introduced into Midas was the ability to send an annotated Protein Data Bank (PDB)[4] file representing the currently displayed scene to an external program.[5] This was motivated by our desire to interface to rendering programs such as RASTER3D.[6] Once this extension mechanism was available, it became relatively easy to develop rendering "back ends" of our own, and soon thereafter we developed both a fast space-filling renderer with shadows, Conic,[7] and a Jane Richardson-style[8] ribbon-depiction program with many capabilities, Ribbonjr. Furthermore, outside developers exploited the mechanism in ways that we had not anticipated. Thomas Hynes (then with Genentech) wrote Neon, a program to allow Midas to depict shadowed ball-and-stick scenes. Neon acted as a filter between Midas and Conic, taking the PDB file output by Midas and producing another PDB file with an order of magnitude more atoms. Neon output contained larger-radius atoms for balls and series of many closely spaced, smaller-radius atoms to simulate sticks. To our surprise, Conic was able to process the Neon file quickly enough (seconds to minutes on computers of the day) to be usable. This whole Neon concept was something that would never have occurred to us, and opened our eyes to the power of an effective extension mechanism.

A second extension mechanism was subsequently created to allow Midas to communicate on an ongoing basis with external programs. The Midas user could send commands to an external program and the external program could issue Midas commands to cause changes in the Midas session. Although this mechanism was also quite successful, it was ultimately constrained to the available Midas command set. The restrictions imposed by the original

Correspondence to: T.E. Ferrin; e-mail: tef@cgl.ucsf.edu
Contract/grant sponsor: NIH; contract/grant number: P41-RR01081

© 2004 Wiley Periodicals, Inc.

 KELONIA EXHIBIT 1027

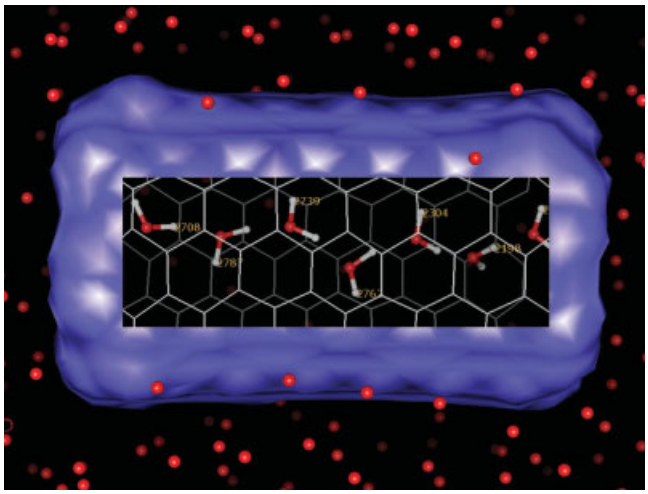

Figure 1. A single frame of a molecular dynamics trajectory of a buckytube in water, shown with the Movie extension. A "lens" has been placed over the center of the tube to strip away the surface and reveal the hydrogen-bonded chain of waters passing through the tube. The trajectory was computed with polarizable molecular dynamics using the Amber 8[26] Sander module (J. Caldwell, UCSF, unpublished data).

Midas design were proving problematic, and we were motivated to create a new system with greater extensibility.

Chimera was designed with extensibility as a primary goal. We also wanted Chimera to be portable to a wide variety of platforms, and to include state-of-the-art graphics capabilities such as transparency and interactive ball-and-stick, space-filling, ribbon, and solid surface representations. Another design goal was to make Chimera accessible to users at all skill levels by providing both a graphical menu/window interface and a command-line interface.

Chimera Architecture

Chimera's primary programming language is Python.[9] Importantly, Python is an interpreted, object-oriented programming language that is also easy to learn and very readable. Because Python is interpreted, it is good for rapid development and debugging. Readability is important for a team development project like Chimera, and an easy-to-learn language enables others to develop extensions without undue effort. Chimera includes the Python-standard IDLE interactive development environment[10] to help diagnose problems during extension development.

Chimera is divided into a *core* and *extensions*. The core provides basic services and molecular graphics capabilities. All higher level functionality is provided through extensions. This design, with the bulk of Chimera functions provided by extensions, ensures that the extension mechanism is robust enough to handle the needs of outside researchers wanting to extend Chimera in novel ways. Extensions can be integrated into the Chimera menu system, and can present a separate graphical user interface as needed using the Tkinter,[11] Tix,[12] and/or Pmw[13] toolkits.

The Chimera core consists of a C++ layer that handles time-critical operations (e.g., graphics rendering) and a Python layer that handles all other functions. All significant C++ data and functions are made accessible to the Python layer. Core capabilities include molecular file input/output, molecular surface generation using the MSMS algorithm,[14] and aspects of graphical display such as wire-frame, ball-and-stick, ribbon, and sphere representations, transparency control, near and far clipping planes, and lenses (screen areas with different display attributes; see Fig. 1).

Another core service is maintenance and display of the current *selection*. Users may select parts of structures by picking with the mouse, by making menu choices (e.g., aromatic rings), or via certain extension actions. The selected structure areas are highlighted with a particular color or a colored outline. Extensions can query for the contents of the selection. Many menu actions (such as coloring or setting the display style) work on the current selection.

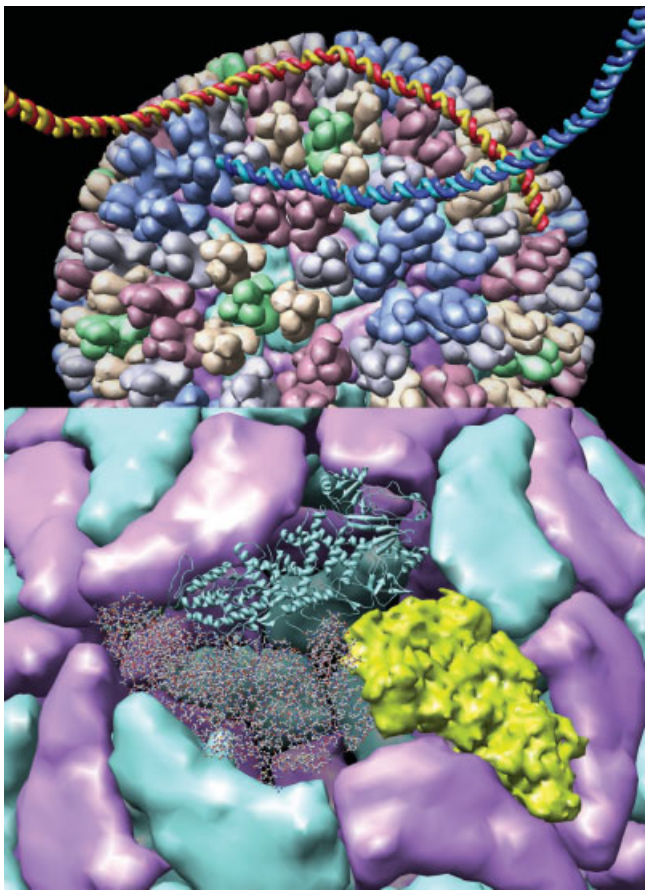

Figure 2. Bluetongue virus core particle (PDB identifier 2btv[16]) with double-stranded RNA attached to the surface (1h1k[17]) (top). Trimers in the outer protein layer that are equivalent under the icosahedral symmetry are given the same color. The free end of the RNA attaches to other viral particles in the crystal. A closeup of the inner layer (bottom) shows ball-and-stick and ribbon models and a surface at higher resolution.

The core also maintains a *trigger* mechanism wherein changes to core data structures or state are reported to extensions that have registered callbacks with the corresponding trigger. For example, there is a "selection changed" trigger that fires whenever the current selection changes.

Extensions are written either entirely in Python or in a combination of Python and C/C++ (the latter using a shared library loaded at runtime). Extensions can be placed in the Chimera installation directory (which would make the extension available to all users) or in the user's own file area. Extensions are loaded on demand, typically when the user accesses a menu entry that starts the extension. The class structure of molecular data and other extension programming information can be found at http://www.cgl.ucsf.edu/chimera/docs/ProgrammersGuide/Examples/.

We demonstrate Chimera's extensibility by presenting several extensions. The Multiscale and Collaboratory extensions are quite unique, and demonstrate the wide spectrum of abilities that can be generated. The others provide insight into the use of core facilities by extensions and the integration of extensions with the Chimera environment.

Multiscale

The Multiscale extension adds capabilities for interactively exploring large molecular assemblies. We have focused on viral structures, condensed chromosomes, and ribosomes; additional examples include cytoskeletal fibers and motors, flagellar structures, and chaperonins. The Multiscale extension displays structures from the PDB[4] and generates their multimeric forms by using transformation matrices to position the subunits. Multiscale can also be used with large assemblies where there are no repeated subunits. PDB chains can be displayed as low-resolution surfaces, or in any of the standard molecular representations available in Chimera. Multiscale permits biologically meaningful levels of quaternary structure to be defined. The abilities to build multimeric forms, display low-resolution representations, and define levels of structure are important for interactively exploring large complexes.

Multiscale uses Chimera's core molecular display abilities, data structures, file reading, and selection management, and the Volume Viewer extension for surface calculation and rendering. This ready-to-use infrastructure allowed us to focus on the new capabilities needed for displaying complexes.

Most of the available atomic-resolution viral structures are icosahedral particles with 60-fold symmetry. PDB files provide atomic coordinates for only one subunit. To build a multimeric model, subunits are positioned using rotation/translation matrices read from the PDB file header. PDB "REMARK 350" records give matrices that can be used to generate the biological oligomerization state. Crystallographic (SMTRY records) and noncrystallographic symmetry (MTRIX records) matrices or matrices inferred from the space group (CRYST1 record) of crystal structures can also be used. Multiscale's visualization capabilities have revealed shortcomings in the matrix information for many large-scale structure entries that otherwise would have been difficult to detect. We are working with the PDB to find and correct these entries.

For efficiency, the Multiscale extension only loads atomic coordinates for subunits when they are needed. When a model is first displayed, only a low-resolution surface representation is shown, so no additional copies of the coordinates need to be loaded. Chimera's atomic and residue-level display styles are also available, but are typically used for only a small number of subunits so that the amount of detail depicted does not overwhelm the user. The Multiscale extension does not use the high-resolution molecular surfaces that are a core feature of Chimera. Such surfaces would render too slowly for a large multimer and provide too high a level of detail to best illustrate the organization of subunits.

The Multiscale extension was originally written entirely in Python. To speed up the surface calculation, certain critical routines were rewritten in C++. Converting these routines from interpreted Python to compiled C++ made them run about 50 times faster. Translation is generally straightforward because Python objects such as molecules, atoms, and lists have equivalent C++ objects. Rendering the surface with OpenGL,[15] another time-critical step, uses the C++ module in the Volume Viewer extension.

Subunits can be selected with the mouse. To simplify selecting larger pieces of a structure, new structure levels can be defined hierarchically. For example, the bluetongue virus core particle[16] has two protein layers, the outer layer being composed of 260 trimers (Fig. 2). For this structure, it is useful to define inner and outer layers and trimers as structural levels. After an individual outer layer monomer is selected, the selection can then be promoted to the containing trimer, and subsequently promoted to the whole layer of trimers. The whole outer layer can then be hidden if the object of interest is the inner protein layer. Besides being promoted, a selection can be extended to all identical copies of the currently selected subunits. It is also possible to select just the subunits for which atomic information has been loaded.

Structure levels can be specified in a Python script. Structural hierarchies are sometimes described in text in the headers of PDB files. The mmCIF file format available from the PDB[4] has a limited ability to describe such higher levels of structure in a computer-readable form, but few submitted data sets provide this information.

For multimeric structures, investigations are often facilitated by the presence of more than one copy of the asymmetric unit. The bluetongue virus structure[16,17] illustrates how working with the full viral shell can aid analysis. The crystallographic data used to determine the capsid structure revealed viral double-stranded RNA stuck to the outside of the particles[17] (Fig. 2). To investigate the specific atomic contacts between the RNA and virus, it is helpful to locate the several subunits of the icosahedral particle adjacent to the RNA by inspecting the full particle. These can then be examined using an all-atom display to determine the contacts accounting for the stickiness of the capsid.

The Multiscale extension is intended for problems where both large-scale and atomic-scale details are relevant. The tools needed to explore models with many levels of structure and large numbers of atoms are necessarily complex; the Chimera Multiscale extension has addressed only the most immediate needs. We anticipate increasing its capabilities significantly in future releases.

Multalign Viewer

The Multalign Viewer extension allows Chimera to display sequence alignments together with associated structures (Fig. 3).

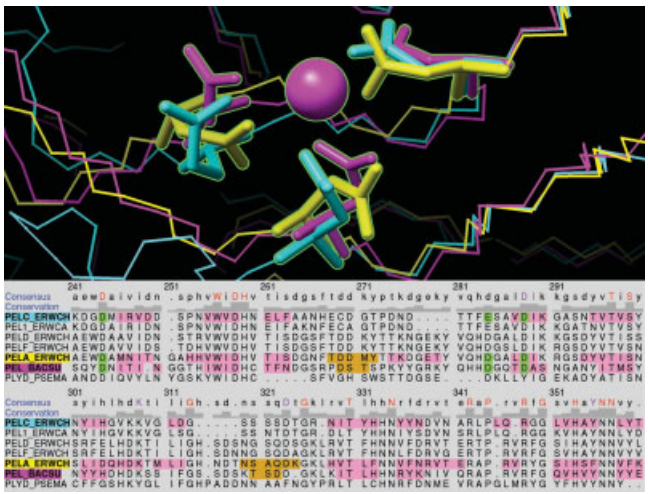

Figure 3. Three structures in Chimera associated with sequences in an alignment shown by Multalign Viewer. The sequences with color swatches behind their names are associated with the pectate lyase structures 1jta,[45] 1bn8,[46] and 2pec[47] (shown in yellow, magenta, and cyan, respectively). The structures were superimposed using the sequence alignment; the fits were refined by iteratively removing bad residue pairings. The sequences are colored by secondary structure (strand and helix regions are pink and gold, respectively) and selected structure regions are green (indicated on the structures with a green outline). Chimera's zone selection method was used to select all residues within 3.6 Å of the active-site metal ion in one of the structures.

Multalign Viewer can read and write sequence alignments in a wide variety of popular formats (currently Clustal[18] ALN, "aligned" FASTA, GCG MSF, GCG RSF, "aligned" NBRF/PIR, and Stockholm).

Multalign Viewer facilitates analysis of alignments in the context of structural information and vice versa. First, structures in Chimera must be associated with their corresponding sequences in an alignment. When Multalign Viewer opens an alignment, it examines the structures currently open in Chimera and checks each chain for high sequence identity with an alignment sequence. Chains with high identity are associated with the best-matching sequence (however, only one chain per structure is associated with a sequence). Multalign Viewer registers with the Chimera core's "model opened" trigger so that as new structures are opened, they will also be examined and associated if appropriate. Conversely, if the alignment sequence names are recognizable as including SCOP[19,20] or PDB[4] identifiers using a few simple criteria, the researcher can use a Multalign Viewer menu item or preference setting to load all of the corresponding structures into Chimera. This was easy to implement, because the Chimera core offers functions for opening PDB/SCOP files based on their identifiers, and will retrieve them via the World Wide Web or local disk as appropriate. If Multalign Viewer fails to make an appropriate automatic association between a sequence and structure, it can be manually directed to make the association. Associations are indicated by showing the color of the associated model behind the name of the sequence (Fig. 3).

Once associations have been set up, many useful features become active. Positioning the mouse over a sequence character shows the number of the corresponding residue (in the structure) in a status area. Making selections on the structures highlights the corresponding regions of the sequence alignment. Dragging boxes on the sequence alignment selects and highlights the corresponding structure regions. Structure regions can be selected based on conservation in the alignment, greatly facilitating coloring by conservation level or showing only conserved side chains. Secondary structure elements can be depicted on the alignment with colored boxes. Clicking on residues in the sequence makes the Chimera window zoom in on the corresponding structure residues. Structures can be superimposed using the sequence alignment, optionally using only highly conserved residues, and also optionally, iteratively refining the fit by pruning poorly superimposed residues.

An alignment can be searched with a literal string or a PROSITE[21] pattern. Matches are highlighted on the alignment, and can also be highlighted on associated structures.

Other extensions can call Multalign Viewer to show alignments. For example, the SSD (Structure Superposition Database)[22] extension uses Multalign Viewer to show sequence alignments corresponding to structural alignments of interest.

Multalign Viewer is under active development. Important short-term goals are to provide more sophisticated editing facilities and to display and interact with phylogenetic information.

ViewDock

The ViewDock extension facilitates interactive screening of ligand orientations from DOCK.[23,24] DOCK calculates possible binding orientations given the structures of ligand and receptor molecules; often, a large database of compounds is searched against a target protein, where each compound is treated as a ligand and the target is treated as the receptor. Simple scoring methods are used to identify the most favorable binding modes of a given molecule and then to rank the molecules. The output consists of a large number of candidate ligands in the binding orientations considered most favorable by DOCK.

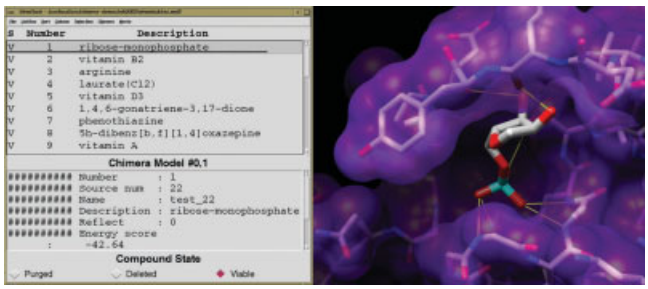

Figure 4. The ViewDock interface lists docked molecules; clicking on a line displays just the corresponding molecule and shows its information in the lower part of the panel. Ribose monophosphate is shown docked to H-Ras (121p[48]). Carbon atoms are light gray, oxygen atoms are red, nitrogen atoms are blue, and phosphorus atoms are cyan. Hydrogens are not shown. Potential hydrogen bonds are indicated with yellow lines.

ViewDock reads the DOCK output and provides a convenient interface for filtering results in the context of the target structure. When a line in the list of compounds is clicked, just the corresponding molecule is shown in the putative binding site, and its information is shown in the lower part of the panel (Fig. 4). The information may include compound name, description, and various scores and score components. Any of the descriptors can be shown in the list and/or used to sort it. It is also possible to view more than one docked molecule at a time.

Compounds can be deleted from the list if visual inspection reveals them to be unsuitable. Compounds can also be screened by the number of hydrogen bonds formed with the target structure and by whether or not the hydrogen bonds involve specified groups in the site. Hydrogen-bond detection uses a set of detailed distance and angle criteria from a published small-molecule crystal survey[25] and is an extension-provided capability of Chimera.

Movie

The Movie extension allows Chimera to show molecular dynamics trajectories. The trajectories may be played forward or backward, either a single frame at a time or continuously. All generic Chimera capabilities such as coloring, hydrogen-bond detection, lenses (Fig. 1), and saving PDB files are available for use with the trajectory. Movie explicitly supports execution of a script (Python or Chimera commands) at each frame. This makes it easy (for example) to save images for later assembly into a QuickTime or MPEG video.

Movie currently supports all versions of AMBER[26] trajectory files, and support for GROMOS[27] and NAMD[28] formats is in progress.

Volume Viewer

The Volume Viewer extension displays three-dimensional (3D) grid data such as density maps from electron microscope reconstructions or X-ray crystallography, calculated electrostatic potential, and solvent occupancy from molecular dynamics simulations. It reads several file formats (CCP4[29] or MRC, BRIX or DSN6,[30] CNS[31] or XPLOR,[32] SPIDER,[33] DelPhi[34] or GRASP[35] potential maps, PRIISM,[36] NetCDF,[37,38] and DOCK[23] scoring grids), displays isosurfaces, meshes, and translucent solids, and allows interactive adjustment of thresholds, transparency, and brightness. Volume data is often displayed with related molecular models.

The display is automatically updated when settings are changed. For example, dragging a threshold indicator shown on a histogram of data values updates the displayed surface or mesh. For large data sets, subsampling can be used to improve the response time when the display is rotated or a threshold is changed. Subsampling with step size 2 renders the data after omitting every other data plane along each axis. Data sets of 256 by 256 by 256 values can be displayed with a new threshold once a second or rotated at 10 frames per second on generic desktop PC hardware equipped with a mid-range graphics adapter. A subregion of the data can be chosen by dragging a box with the mouse and then shown instead of the whole data set. Subregions can be named so that it is easy to return to them in later sessions.

The Volume Path Tracer extension allows marker placement and path tracing in grid data. Markers are placed by mouse-click; the marker is positioned on the closest visible data maximum along the line of sight under the cursor. Markers can be moved after they are placed. Consecutively placed markers can be linked with segments to trace a path. Additional connections can be added with the mouse to build simple structural models. Volume Path Tracer was developed to trace protein backbones in intermediate-resolution (5–8 Å) density maps from electron cryo-microscopy (Fig. 5) and fluorescently labeled chromosomes in 3D multiwavelength light microscopy data (Fig. 6).

Markers and associated connecting segments are implemented using the same mechanism as atoms and bonds. Thus, display styles and colors can be changed in the same way as for atoms and bonds, distances between pairs of markers and between markers and molecular structures can be measured, and traced structures can be aligned using Chimera's molecule manipulation capabilities. Traced paths can be displayed as smoothly interpolated curves (Fig. 6). Markers and connecting segments can be saved in an XML[39] file for analysis by other software.

Collaboratory

Chimera's Collaboratory extension enables researchers at geographically distant locations to share a molecular modeling session in real time. By default, all users connected to the same session have equal control over the models (structures) being viewed. A change made by any participant is immediately propagated to all other participants, so that a synchronized view of the data is maintained throughout a session.

Because of the complex 3D nature of molecular models, interactive examination of models in a real-time collaborative environment is far more effective than traditional asynchronous forms of communication, such as passing molecular data back and forth through e-mail. A crucial element of real-time collaboration software is the efficient transfer of information. Not only must the software ensure that information is transmitted rapidly, it must also consider the availability of network resources, such as bandwidth, and transmit information in an efficient format so that the application can run in parallel with other bandwidth-heavy applications such as videoconferencing software. Given these considerations, application-independent collaboration tools are not as responsive for sharing molecular modeling sessions. Such desktop-sharing applications (e.g., Microsoft NetMeeting and Virtual Network Computing[40]) function by transmitting the contents of the screen from the workstation running the application to all who are sharing it. This can be bandwidth-intensive, especially for molecular graphics, where most operations alter a large amount of screen content. Instead, the Collaboratory works on a lower level, by transmitting small messages describing just the data that has been modified.

The Collaboratory utilizes a star architecture, with a central hub connected to multiple nodes. Each node is a user running Chimera, while the hub is a separately running application that acts as a rendezvous point between the nodes. Participants' instances of Chimera are notified when a model has been opened, closed, or modified. A data file need only be present on the system of the participant who opens the file in Chimera. Parameters monitored

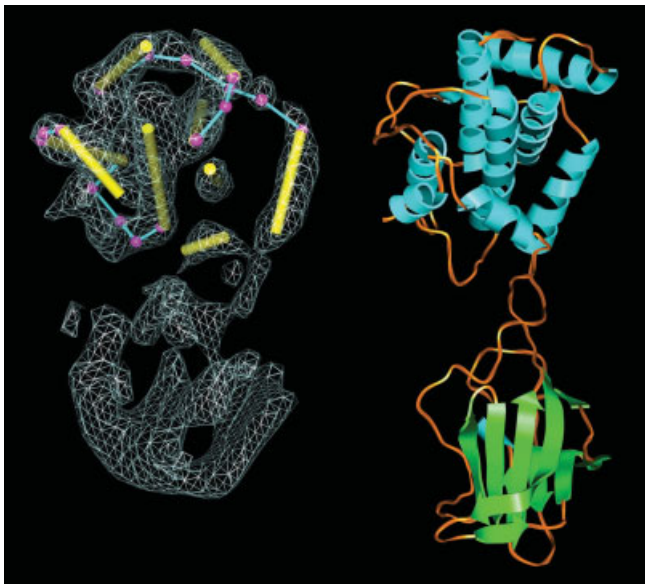

Figure 5. Density map for rice dwarf virus P8 capsid protein shown as a mesh (left). The map was obtained by electron cryo-microscopy and has 6.8-Å resolution.[49] Computationally identified alpha helices are shown as cylinders[50] and the connecting turns have been hand traced using Chimera's Volume Path Tracer. On the right is a crystal structure of the distantly related bluetongue virus capsid protein (1bvp[51]).

for changes include display (color, molecular representation) and spatial properties such as position and orientation. Change tracking is accomplished using the Chimera core's trigger mechanism. CORBA (Common Object Request Broker Architecture)[41] messages are used to relay information; they are platform- and language-independent, allowing the Collaboratory to work on a variety of systems.

The person who starts the hub acts as the session administrator. The administrator can specify a password that others must supply to join the session. The administrator can also obtain information about current participants (such as username and total time connected), close the session to new participants, and control whether each user can affect the state of the session. Participants can join the session at any time; when they sign in, their local instances of Chimera will be updated to reflect the global state of the session. Participants can leave the collaborative session at any time and continue their modeling sessions in an isolated setting.

Several features facilitate remote collaboration. A participant can display the position of his pointer on collaborators' screens to draw attention to a region of interest. Although users are expected to communicate primarily by telephone or an independent videoconferencing application, the Collaboratory provides a "chat" mechanism for passing text messages. This can be especially useful for transferring URLs or sequence information. Users can also view other participants' commands as they are entered and see who is actively manipulating the models.

The star architecture scales well because each node has knowledge of only the hub, not other nodes. The Collaboratory has been used productively with three participants in a session, and in a test situation has been able to accommodate up to 10 users. In practice, human factors tend to limit the size of a useful collaboration. Interpersonal communication can be difficult when subtle visual cues such as body language and eye contact are missing, as is often the case with videoconferencing. However, when used in a sensible collaborative setting (i.e., efficient voice/video transmission, a reasonable number of participants), the Chimera Collaboratory and its rich set of features help to alleviate these issues.

Results

Chimera is freely available to academic and nonprofit researchers from http://www.cgl.ucsf.edu/chimera/, and can be licensed by commercial institutions for a fee. Extensions to Chimera developed by outside researchers can be redistributed freely. Since Chimera's first public release in March 2000, downloads of Chimera have steadily increased, amounting to approximately 1000 per month at this writing and totaling more than 12,000. (Multiple downloads to the same IP address for the same OS platform in a single 24-h period are counted as a single download.)

As stated earlier, portability has been one of the primary design goals of Chimera. Our implementation, based on many widely available standards such as OpenGL,[15] has proven to be quite portable, and there are Chimera distributions for Microsoft Win-

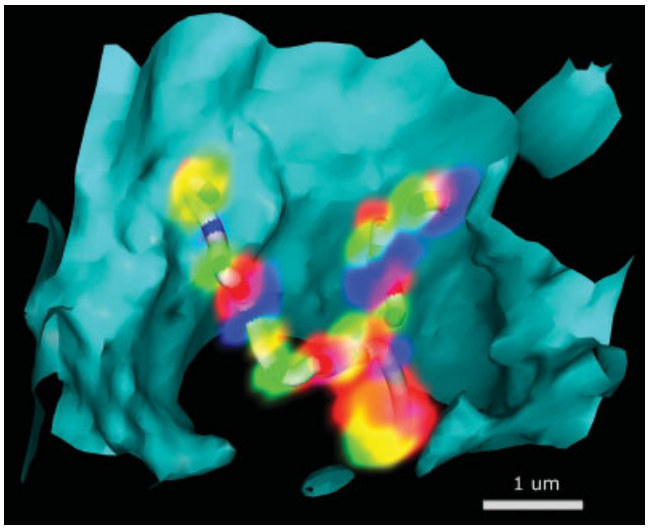

Figure 6. Fluorescently labeled *Drosophila* chromosome imaged by wide-field deconvolution microscopy (M. Lowenstein and J. Sedat, UCSF, unpublished data). Three fluorophores were used to label specific segments of chromosome 2L. Two homologous copies of the chromosome are shown. The cyan isosurface is the nuclear envelope. The individual fluorescent spots have been marked with the Volume Path Tracer extension and paths connecting the markers are shown as smooth tubes. Traced structures from many cells can be clustered to study structural patterns of chromosome organization in the nucleus.

dows XP/NT/2000/98, Linux, Apple Mac OS X, SGI IRIX, and HP Tru64 Unix.

The graphical menu/window and command-line interfaces provide rich and overlapping sets of functionality, which will continue to grow as Chimera is developed. Detailed user documentation is included with the program.

Our goal of enabling others to extend Chimera has enjoyed increasing success as the use of Chimera has become more widespread. Currently, extensions written and/or distributed by others include: SSD, which extends Chimera for use with the Structure Superposition Database[22] (a Web-accessible resource); ViewFeature,[42] which allows Chimera to show results from FEATURE,[43] a program for predicting metal-binding or active sites in biomolecules; and EMANimator (S. Ludtke, Baylor College of Medicine, unpublished), which makes it easy to create animation sequences in Chimera and save them as MPEG files. In addition, Chimera is supported as a display application for sequence and structure information by the Web-accessible databases ModBase[44] and the Structure Function Linkage Database (Babbitt laboratory, UCSF).

Discussion

Chimera has been designed to facilitate the addition of new functionality. In particular, nearly all concepts, for example, atoms, bonds, and molecules, are represented as Python objects, which means they can take advantage of the object-oriented nature of Python. The downside of making everything into Python objects is a performance penalty, both in speed and in memory usage. We decided to favor programmability over performance. Whereas performance deficiencies can be addressed transparently to end users by algorithmic changes and hardware improvements such as increased processor speed and faster graphics adapters, modifications to user and programming interfaces are typically more disruptive.

Currently, one of our biggest challenges is to improve performance in the context of the large-scale systems for which the Multiscale extension was designed. Although many structures from crystallography and NMR determination are not very large (tens of thousands of atoms), the complexity increases rapidly as one considers objects such as viral particles. The order-of-magnitude increases in scale require better algorithms for simplifying the representation and more efficient use of desktop computing resources.

In the orthogonal direction of programmability, we are investigating avenues for facilitating interactions between Chimera extensions. In our current implementation, extensions must explicitly monitor predefined triggers to detect changes made by the Chimera core and other extensions. The change-detection granularity is currently quite coarse, meaning that extensions may need to sift through a large amount of data to detect a few interesting data modifications. We plan to alter the low-level object implementation to address these shortcomings.

Acknowledgments

We thank David Konerding and Joe Murray for their contributions to early versions of Chimera. We would also like to thank Patricia Babbitt, Jim Caldwell, Ranyee Chiang, John Gerlt, David Konerding, Mike Lowenstein, Steve Ludtke, Sean Mooney, Walter Novak, Connie Oshiro, Walter Scott, and Nick Ulyanov for their many helpful suggestions.

References

1. Langridge, R.; Ferrin, T. E.; Kuntz, I. D.; Connolly, M. L. Science 1981, 211, 661.
2. Huang, C. C.; Couch, G. S.; Pettersen, E. F.; Ferrin, T. E. Pac Symp Biocomput 1996, 1, 724.
3. Ferrin, T. E.; Huang, C. C.; Jarvis, L. E.; Langridge, R. J Mol Graph 1988, 6, 13.
4. Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. Nucleic Acids Res 2000, 28, 235.
5. Couch, G. S.; Pettersen, E. F.; Huang, C. C.; Ferrin, T. E. J Mol Graph 1995, 13, 153, 193.
6. Bacon, D. J.; Anderson, W. F. J Mol Graph 1988, 6, 219.
7. Huang, C. C.; Pettersen, E. F.; Klein, T. E.; Ferrin, T. E.; Langridge, R. J Mol Graph 1991, 9, 230, 242.
8. Richardson, J. S. Adv Protein Chem 1981, 34, 167.
9. Python; http://www.python.org/.
10. IDLE (Interactive DeveLopment Environment); http://www.python.org/idle/.
11. Tkinter; http://www.python.org/topics/tkinter/.
12. Tix (Tk Interface eXtension); http://tixlibrary.sourceforge.net/.
13. Pmw (Python megawidgets); http://pmw.sourceforge.net/.
14. Sanner, M. F.; Olson, A. J.; Spehner, J. C. Biopolymers 1996, 38, 305.
15. OpenGL; http://www.opengl.org/.
16. Grimes, J. M.; Burroughs, J. N.; Gouet, P.; Diprose, J. M.; Malby, R.; Zientara, S.; Mertens, P. P.; Stuart, D. I. Nature 1998, 395, 470.
17. Diprose, J. M.; Grimes, J. M.; Sutton, G. C.; Burroughs, J. N.; Meyer, A.; Maan, S.; Mertens, P. P.; Stuart, D. I. J Virol 2002, 76, 9533.
18. Chenna, R.; Sugawara, H.; Koike, T.; Lopez, R.; Gibson, T. J.; Higgins, D. G.; Thompson, J. D. Nucleic Acids Res 2003, 31, 3497.
19. Lo Conte, L.; Brenner, S. E.; Hubbard, T. J.; Chothia, C.; Murzin, A. G. Nucleic Acids Res 2002, 30, 264.
20. Murzin, A. G.; Brenner, S. E.; Hubbard, T.; Chothia, C. J Mol Biol 1995, 247, 536.
21. Sigrist, C. J.; Cerutti, L.; Hulo, N.; Gattiker, A.; Falquet, L.; Pagni, M.; Bairoch, A.; Bucher, P. Brief Bioinform 2002, 3, 265.
22. Chiang, R. A.; Meng, E. C.; Huang, C. C.; Ferrin, T. E.; Babbitt, P. C. Nucleic Acids Res 2003, 31, 505.
23. Ewing, T. J.; Makino, S.; Skillman, A. G.; Kuntz, I. D. J Comput Aided Mol Des 2001, 15, 411.
24. Kuntz, I. D. Science 1992, 257, 1078.
25. Mills, J. E.; Dean, P. M. J Comput Aided Mol Des 1996, 10, 607.
26. Case, D. A.; Pearlman, D. A.; Caldwell, J. W.; Cheatham, T. E. I.; Wang, J.; Ross, W. S.; Simmerling, C.; Darden, T.; Merz, K. M.; Stanton, R. V.; Cheng, A.; Vincent, J. J.; Crowley, M.; Tsui, V.; Gohlke, H.; Radmer, R.; Duan, Y.; Pitera, J.; Massova, I.; Seibel, G. L.; Singh, U. C.; Weiner, P.; Kollman, P. A. AMBER; http://amber.scripps.edu/.
27. van Gunsteren, W. F.; Berendsen, H. J. C. GROMOS; http://www.igc.ethz.ch/gromos/.
28. Kale, L.; Skeel, R.; Bhandarkar, M.; Brunner, R.; Gursoy, A.; Krawetz, N.; Phillips, J.; Shinozaki, A.; Varadarajan, K.; Schulten, K. J Comput Phys 1999, 151, 283.
29. Acta Crystallogr D 1994, 50, 760.

30. Jones, T. A.; Kjeldgaard, M. The O protein crystallographic package; http://www.imsb.au.dk/~mok/o/.
31. Brunger, A. T.; Adams, P. D.; Clore, G. M.; DeLano, W. L.; Gros, P.; Grosse–Kunstleve, R. W.; Jiang, J. S.; Kuszewski, J.; Nilges, M.; Pannu, N. S.; Read, R. J.; Rice, L. M.; Simonson, T.; Warren, G. L. Acta Crystallogr D 1998, 54, 905.
32. Badger, J.; Kumar, R. A.; Yip, P.; Szalma, S. Proteins 1999, 35, 25.
33. Frank, J.; Radermacher, M.; Penczek, P.; Zhu, J.; Li, Y.; Ladjadj, M.; Leith, A. J Struct Biol 1996, 116, 190.
34. Honig, B.; Nicholls, A. Science 1995, 268, 1144.
35. Nicholls, A.; Sharp, K. A.; Honig, B. Proteins 1991, 11, 281.
36. Chen, H.; Hughes, D. D.; Chan, T. A.; Sedat, J. W.; Agard, D. A. J Struct Biol 1996, 116, 56.
37. Rew, R. K.; Davis, G. P. IEEE Comput Graph Appl 1990, 10, 76.
38. Network Common Data Form; Unidata Program Center, Boulder, CO; http://www.unidata.ucar.edu/packages/netcdf/.
39. Extensible Markup Language; http://www.xml.org/.
40. Richardson, T.; Stafford–Fraser, Q.; Wood, K. R.; Hopper, A. IEEE Internet Comput 1998, 2, 33.
41. Common Object Request Broker Architecture; http://www.corba.org/.
42. Banatao, D. R.; Huang, C. C.; Babbitt, P. C.; Altman, R. B.; Klein, T. E. Pac Symp Biocomput 2001, 6, 240.
43. Bagley, S. C.; Altman, R. B. Protein Sci 1995, 4, 622.
44. Pieper, U.; Eswar, N.; Braberg, H.; Madhusudhan, M. S.; Davis, F. P.; Stuart, A. C.; Mirkovic, N.; Rossi, A.; Marti–Renom, M. A.; Fiser, A.; Webb, B.; Greenblatt, D.; Huang, C. C.; Ferrin, T. E.; Sali, A. Nucleic Acids Res 2004, 32, D217.
45. Thomas, L. M.; Doan, C. N.; Oliver, R. L.; Yoder, M. D. Acta Crystallogr D 2002, 58, 1008.
46. Pickersgill, R.; Jenkins, J.; Harris, G.; Nasser, W.; Robert–Baudouy, J. Nat Struct Biol 1994, 1, 717.
47. Yoder, M. D.; Jurnak, F. FASEB J 1995, 9, 335.
48. Wittinghofer, F.; Krengel, U.; John, J.; Kabsch, W.; Pai, E. F. Environ Health Perspect 1991, 93, 11.
49. Zhou, Z. H.; Baker, M. L.; Jiang, W.; Dougherty, M.; Jakana, J.; Dong, G.; Lu, G.; Chiu, W. Nat Struct Biol 2001, 8, 868.
50. Jiang, W.; Baker, M. L.; Ludtke, S. J.; Chiu, W. J Mol Biol 2001, 308, 1033.
51. Grimes, J.; Basak, A. K.; Roy, P.; Stuart, D. Nature 1995, 373, 167.