Journal of Computer-Aided Molecular Design (2005) 19: 791–800
DOI 10.1007/s10822-005-9027-x

BALLView: An object-oriented molecular visualization and modeling framework

Andreas Moll[a,*], Andreas Hildebrandt[a], Hans-Peter Lenhof[a,†] & Oliver Kohlbacher[b,†]
[a]*Center for Bioinformatics, Saarland University, 15 11 50, 66041, Saarbrücken, Germany;* [b]*Center for Bioinformatics, Eberhard Karls University, Tübingen, Sand 14, 72070, Tübingen, Germany*

Received 29 August 2005; accepted 25 October 2005
© Springer 2006

Key words: Molecular dynamics, Molecular mechanics, Molecular modeling, Rapid prototyping, Molecular visualization

Summary

We present BALLView, an extensible tool for visualizing and modeling bio-molecular structures. It provides a variety of different models for bio-molecular visualization, e.g. ball-and-stick models, molecular surfaces, or ribbon models. In contrast to most existing visualization tools, BALLView also offers rich functionality for molecular modeling and simulation, including molecular mechanics methods (AMBER and CHARMM force fields), continuum electrostatics methods employing a Finite-Difference Poisson Boltzmann solver, and secondary structure calculation. Results of these computations can be exported as publication quality images or as movies. Even unexperienced users have direct access to this functionality through an intuitive graphical user interface, which makes BALLView particularly useful for teaching. For more advanced users, BALLView is extensible in different ways. Owing to its framework design, extension on the level of C++ code is very convenient. In addition, an interface to the scripting language Python allows the interactive rapid prototyping of new methods. BALLView is portable and runs on all major platforms (Windows, MacOS X, Linux, most Unix flavors). It is available free of charge under the GNU Public License (GPL) from our website http://www.ballview.org.

Introduction

Three-dimensional visualization is an essential requirement for numerous applications in structural biology and structural bioinformatics. Most of the freely available tools for molecular visualization (e.g. VMD [1], PyMOL [2], WebLabViewer [3], or RasMol [4]) are more or less monolithic applications well-suited for molecular visualization, but lacking further functionality, e.g. for molecular modeling. Only a few programs like the Swiss-PDBViewer [5] offer additional modeling capabilites. Other visualization packages, like AVS [6] are extremely powerful for general visualization tasks, but quite difficult to adapt to specific tasks in molecular modeling. Another Open Source toolkit is OpenMOIV [7], that has evolved from the Molecular Inventor [8] library from SGI. Commercial molecular modeling packages like SYBYL [9], Discovery studio [3], or MOE [10] aim to provide a convenient, easy-to-use interface to the broad functionality in molecular

*To whom correspondence should be addressed. E-mail: amoll@bioinf.uni-sb.de
†Hans-Peter Lenhof and Oliver Kohlbacher contributed equally to this work

modeling and computer-aided drug design lying underneath.

Here, we present a new visualization framework, BALLView, providing a flexible and convenient interface to the molecular modeling functionality of our C++ framework BALL [11], which was specifically developed for rapid prototyping applications in molecular modeling and structural bioinformatics. BALLView combines state-of-the-art visualization techniques and a graphical user interface with the powerful molecular modeling capabilities of BALL. The major design goals were ease of use, extensibility, and portability. These goals should make BALLView easily amenable even for inexperienced users (e.g. for students), but also provide a useful tool for rapid prototyping of visualization applications in the hands of an experienced modeler or software developer. Among the key features of BALLView are molecular mechanics methods (AMBER and CHARMM force fields), Poisson-Boltzmann electrostatics, visualization of electrostatic properties, and convenient creation of movies. All these features are not only accessible through an intuitive graphical user interface but also through the object-oriented scripting language Python [12]. BALLView is available free of charge under an Open Source license (GPL) from our website www.ballview.org. Its availability for all major platforms and its ease-of-use make BALLView a particularly valuable tool in teaching molecular modeling and structural bioinformatics.

Section 2 gives an overview of BALLView's features and capabilities. The underlying design concepts and implementation details are described in Section 3.

Features

Ease-of-Use

BALLView provides rich visualization capabilities rivaling those of established molecular viewers like PyMOL [2], VMD [1] or RasMol [4]. The main difference lies however in the user interface (see Figure 1), which is based on state-of-the-art user-interface design principles (see Section 3). The user interface of BALLView is highly configurable. Its individual components, called widgets, can be freely arranged within or outside the main window, and can be switched on or off. Through usability studies with small student groups, we have ensured that despite its functionality, BALLView remains intuitively useable. A built-in step-by-step tutorial leads the unexperienced user through the basic functionality in just a few minutes. Context-dependent help texts provide additional information for the whole interface.

Rich Functionality

In contrast to most other molecular visualization tools, BALLView also provides access to a rich molecular modeling functionality. Force field methods or electrostatics calculation capabilities similar to those of tools like GRASP [14] and DelPhi [15] are available through the same interface and do not require the user to change between tools or to get acquainted to different file formats and interfaces. The ease of use of this functionality can be demonstrated by, e.g. the computation and visualization of the electrostatic potential of the glucocorticoid receptor (PDB ID: 1GLU), as shown in Figure 2. Generating the potential and projecting it onto molecular representations like Connolly surfaces is a matter of just a few mouse clicks: first, the file 1GLU is downloaded from the PDB [16], using BALLView's file dialogs. Then, hydrogen atoms are added and optimized using BALL's implementation of the AMBER force field. Using the integrated Finite-Difference Poisson-Boltzmann solver, we can now compute a user-defined potential grid. From the variety of available molecular representations we select a cartoon model for the DNA and a Conolly surface for the protein. The surface is then colored with respect to the computed electrostatic potential.

Extensibility

The functionality of BALLView can be extended on several levels with little effort. First, the user can add C++ code for arbitrary methods which can be easily made available from the graphical user interface with minimal changes to the BALLView code (see Section 3). This option is clearly intended for more experienced code developers. Second, users can employ Python as a scripting language to extend BALLView. Python [12] is a powerful, nevertheless easy to read and easy to learn, object-oriented language. Thus, this option

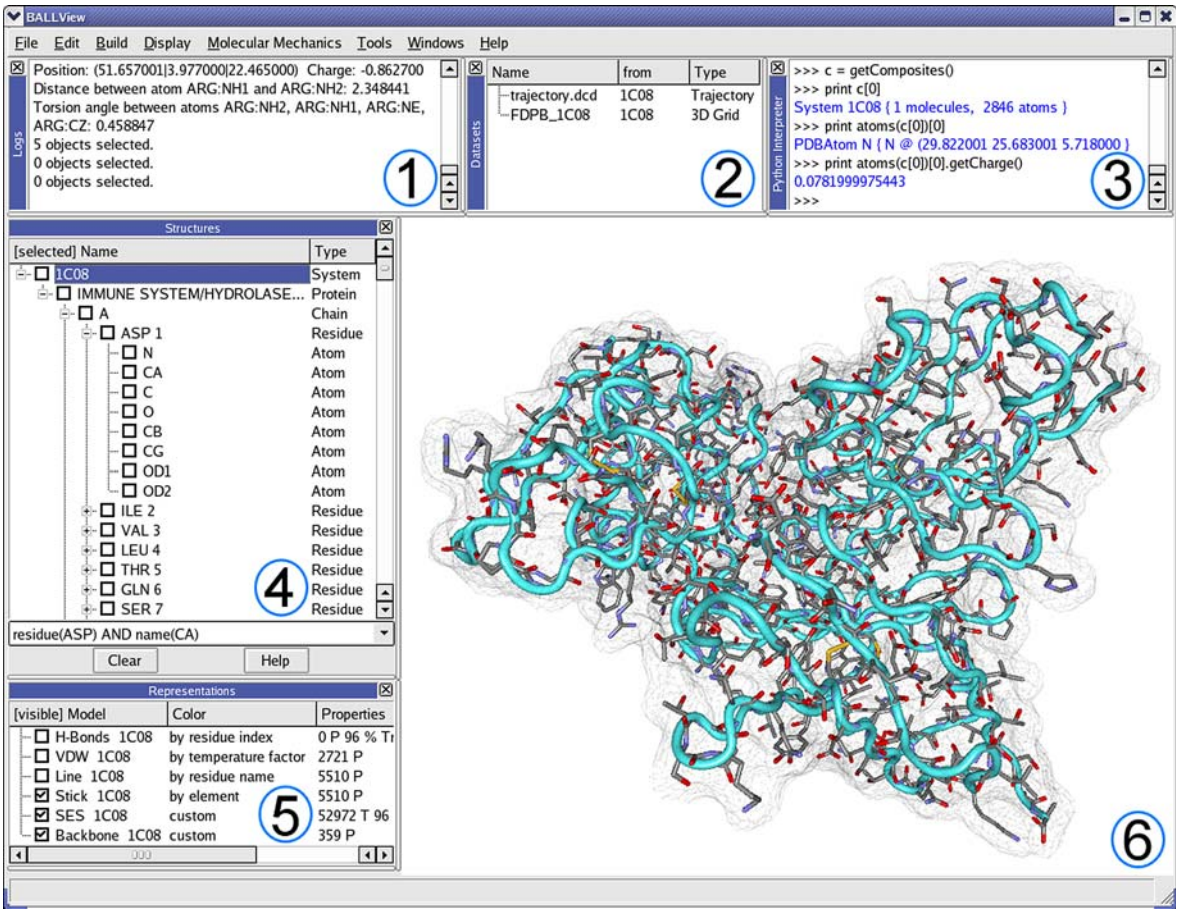

Figure 1. The user interface of BALLView is based on QT [13]. It contains several independent subwindows (widgets) that can be freely arranged, resized, and completely hidden, if they are not needed. The main widget is the 3D view displaying all currently activated models (6). The currently loaded molecular structures are shown in a hierarchical fashion in the structure widget (4). For any of these structures or parts thereof the user can construct representations, i.e. models of these structures, which can be manipulated, enabled and disabled through the representation widget (5).

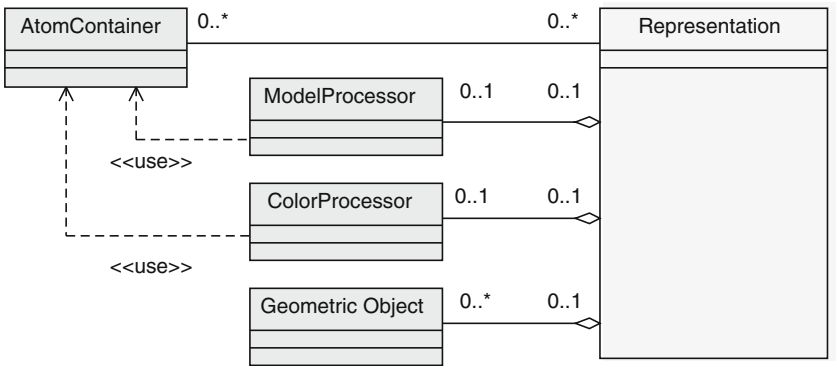

Figure 2. Each visualized object corresponds to an instance of the class Representation. The ModelProcessor creates Geometric Objects, e.g. tubes or meshes for all atoms stored in the AtomContainers. Next, the ColorProcessor colors the GeometricObjects, e.g. by element, charge or temperature factor. The individual model types and coloring methods are realized by derived classes. This approach simplifies the creation of new models and coloring methods and allows their free combination.

is also suited for less experienced users. BALLView provides an embedded Python interpreter giving access to the full functionality of BALL and allowing a simple automation of frequently occuring tasks. The object-oriented design of the underlying libraries and the application itself simplify the extension tremendously. A simple but nonetheless useful example is given below:

```
S = getSystem(0)

text = "PM3 1SCF MMOK GEO-OK\n\n\n"
for i in atoms(S):
  e = i.getElement().getSymbol()
  p = i.getPosition()
  text += "%s %f 1 %f 1 %f 1\n" % (e, p.x, p.y, p.z)

open("tmp.dat", 'w').write(text)

os.system("run_mopac tmp")
print os.popen('grep "TOTAL ENER" tmp.out').read()
```

This script creates a MOPAC [17] input file containing the atoms of the first system currently loaded in BALLView, calls MOPAC to execute a single-point calculation using the PM3 semi-empirical Hamiltonian and prints out the resulting total energy. With just a few lines more, this example could be adapted to other quantum chemistry packages or to minimize the structure's energy and return the optimized structure. The execution of such a Python script can also be linked to a user-defined hotkey, allowing to execute the script by pressing a single key.

BALLView as a Tool for Teaching

We have been using BALL and BALLView extensively in graduate and undergraduate courses on computer-aided drug design, molecular modeling, structural bioinformatics, and protein structure. In our experience, BALLView has proved to be an ideal tool for teaching structural bioinformatics and molecular modeling. This is mainly due to the intuitive interface which lowers the barriers usually imposed by the difficult handling of many standard tools. Teaching can thus focus on the methods and the theory behind them rather than spending too much time on interface and file format issues.

Due to its portability, BALLView runs on most relevant platforms. It has been successfully tested on Solaris, Linux, MacOS, and Microsoft Windows machines in different class room setups. A further advantage is its free availability. Students can easily download the latest version and install it on their own desktop machines or laptops. We are currently working on course materials specifically designed for use with BALLView and intend to make them available on the BALLView website.

Further Functionality

In order to give the interested reader a more detailed impression of its capabilities and functionality, some of the more important features of BALLView are summarized in this section.

Models

BALLView provides all standard graphical models (see Table 1) and coloring methods. The models

Table 1. Models and coloring methods supported by BALLView. Any models and coloring schemes can be freely combined and applied to arbitrary subsets of atoms in a molecule.

Models	Coloring Methods
Line	by element
Stick	by atom charge/distance
Ball and stick	by residue index/name/type
Van der Waals (VDW)	by secondary structure
Solvent-excluded/accessible surface	by chain/molecule
Isocontour surface	by forces
Backbone	by occupancy
Cartoon	by temperature factor
Hydrogen-bonds	by a custom color

and coloring methods can be freely combined. Furthermore, every model can be drawn in three different rendering styles (solid, wireframe, dots) and can be displayed transparently. BALLView's ability to freely combine different models, coloring and rendering styles enables the user to visualize complex molecular scenes, e.g. the visualization of a molecule's charge along with its structure (see Figure 2).

Among the most important models implemented in BALLView are three different surface definitions: solvent-accessible and solvent-excluded surfaces (SES/SAS) can be created with adaptable probe radius and resolution. Regulary spaced data grids can be used to calculate isocontour surfaces, e.g. for electrostatics (see Figure 2). The viewer also provides the possibility to visualize surface patches, e.g. the binding pocket in the vicinity of a ligand (see Figure 3). To offer the user an intuitive way of handling models and their coloring, we created the class Representation. For each visualized object, this class stores the selection of molecular entities, the model and coloring method, the drawing style and the geometric objects representing the model (see Figure 4). This allows the user to enable and disable individual representations or to store them in a file for later usage.

Selections

It is often desirable to work on specific subsets of atoms in a structure. These subsets can be defined in three different ways: first, users can select atoms or molecules from the OpenGL widget by clicking into the scene. The second possiblity is provided by the structure widget (see number 4 in Figure 1) which contains a hierarchical overview of the amino acid sequence and the corresponding atoms. Furthermore, advanced users can enter complex boolean expressions into a special widget. As an example, the expression "element (H) AND residue (LYS)" would select all hydrogen atoms in lysine side chains. These selections can also be used to apply a model to subsets of structures, e.g. to highlight specific regions of a molecule.

High-quality images and movies

The 3D view of BALLView is just one way of rendering representations of molecular structures. In a similar fashion, every model can also be

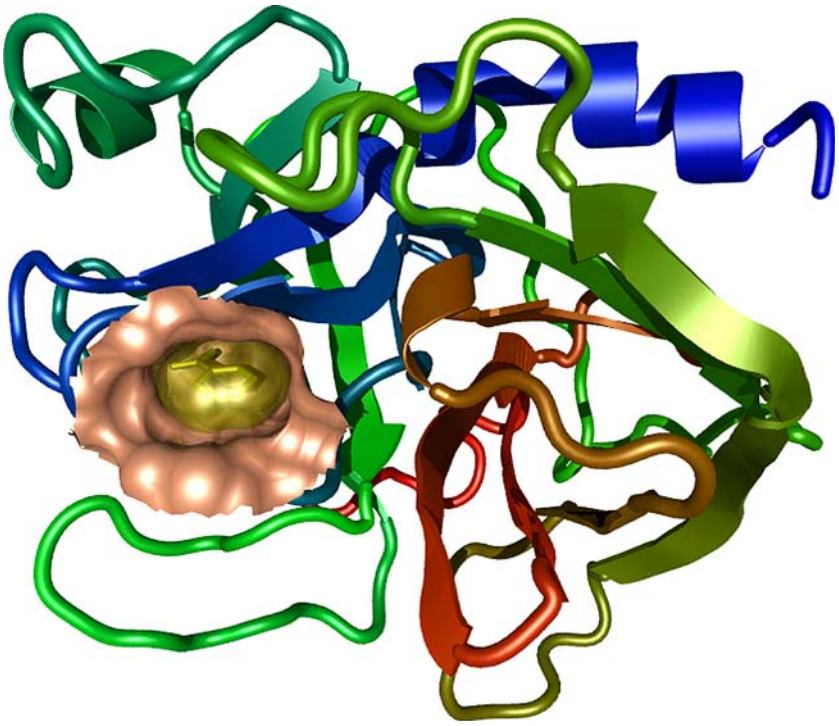

Figure 3. This image was created using BALLView's POVRay export feature. It shows a cartoon model in combination with a transparent surface and a split surface.

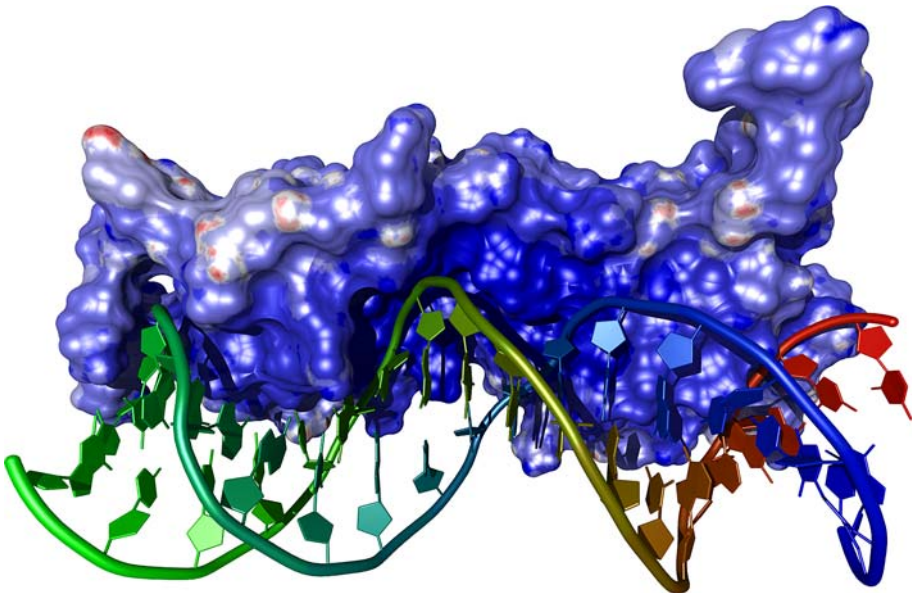

Figure 4. This image shows the DNA-binding domain of the glucocorticoid receptor binding to DNA. The receptor is shown as a solvent-excluded surface colored by its electrostatic potential (blue: positive potential, red: negative potential). The potential was computed using the FDPB solver integrated in BALLView. The DNA is drawn as a cartoon model showing the individual bases, sugar residues and the phosphate backbone in a schematic representation.

exported to other output devices, e.g. to POV-Ray format (Persistence of Vision, a raytracer [18]) to produce publication quality images (see Figures 3 and 4) with correct shadows and arbitrary resolution.

Trajectories, created by molecular dynamics simulations in BALLView or by external programs, can be read and stored in DCD-format. They can also be visualized with any possible model or combination of models. Additionally, movies can be created from trajectories.

File formats

Importing and exporting datasets are prerequisites for virtually all computational tasks. Thus, BALLView contains native support for a wide range of file formats: it can read and write e.g. PDB, HIN, MOL, MOL2, and SD files. The graphical user interface also provides the means to download PDB files directly from the protein data bank [16] either by using the PDB ID or by searching for keywords.

Molecular modeling

BALLView provides a common graphical interface for the wide range of molecular modeling capabilities implemented in the library BALL [11, 19]. BALLView thus shares BALL's molecular mechanics features: we have a native implementation of the AMBER [20] and CHARMM [21] force fields and we are currently working on adding further forcefields e.g. MMFF94 [22]. With these force fields, users can calculate single point energies or relax strained conformations by using energy minimizations. Multithreading allows the online visualization of the molecular dynamics simulations and energy minimizations. The resulting trajectories can be exported, visualized, and used to create movies. To calculate the electrostatic potentials of molecular structures we have implemented a Finite-Difference Poissons-Boltzmann (FDPB) solver. The resulting potentials can be visualized by coloring solvent accessible/excluded surfaces. It is also possible to create isocontour surfaces, which can be colored at the user's discretion. Furthermore, it is possible to export the potential grids, e.g. for usage in external programs.

A frequent problem when applying molecular mechanics methods are incomplete structures due to missing atoms, in particular hydrogens. The efficient heuristics for placing missing atoms implemented in BALL estimate good atom positions automatically. Using BALL's fragment database, BALLView can also check for common

structural problems, e.g. overlapping atoms, strange charges, or bond lengths. The atoms in question become highlighted for easier identification and manipulation.

An implementation of a variant of the DSSP algorithm [23] allows the automatic assignment of secondary structure elements.

The design and usage of the molecular modeling components have been described previously [11, 19]. It has been tested thoroughly and yields results that are consistent with those of the original packages, e.g. AMBER and DelPhi.

High performance 3D graphics

For high performance hardware accelerated 3D graphics, BALLView uses the OpenGL or Mesa libraries, which are available for almost all platforms and graphics accelerator cards. To adapt to different hardware capabilities, three different detail levels can be choosen for each model. Therefore it is possible to create visualizations for molecules with thousands of atoms even on laptops with low-end graphics cards. For best results, users can define the position and intensity of the light sources, e.g. to highlight a specific section of a molecule.

BALLView also provides support for stereo 3D viewing, both for shutter glasses and side-by-side stereo, e.g. for use with two projectors and polarization filters.

Documentation and support

BALL and BALLView provide extensive documentation, available in HTML format, describing the installation of the library, the interface of the classes and usage of the standalone viewer. To familiarize new users with BALLView, a demo and tutorial are provided. For developers, we created an additional detailed tutorial showing among other things how to extend the viewer or use BALL's data structures. Furthermore, the project's website offers an FAQ section and a mailing list for obtaining support.

Design and implementation

The design of BALLView is based on the principles employed for our application framework BALL [11, 19, 24]. The main design goals are extensibility, portability, robustness, and ease-of-use. The targeted users of BALLView are twofold: first, users requiring an easy-to-use visualization and modeling tool; second, software developers building tailor-made applications for their own tasks, e.g. as visualization front end for their own code. To satisfy these very different needs, we have developed an object-oriented application framework allowing for a number of different levels of customization and extension.

To meet the need for a state-of-the-art, user-friendly, and portable graphical user interface (GUI), we decided to base the visualization on QT [13], a GUI toolkit available for most relevant platforms. QT provides all essential functionality to build graphical applications in C++ . Its object-oriented design fits well with the design of BALL, allowing for a seamless integration of the two libraries. The three-dimensional visualization is based on OpenGL [25], the current industry standard for platform-independent 3D graphics. Since OpenGL does not provide an object-oriented API, we wrapped the key functionality of OpenGL into a class interface. This combination of QT and OpenGL ensures a maximum of performance and portability. Therefore, BALL and BALLView are almost platform-independent. Both run on virtually all common graphical operating systems: most current versions of Unix/Linux, Windows, and MacOS X are supported.

To keep the design of the viewer modular and extensible, we have developed a way to add further functionality with minimal effort. Most of the effort in GUI programming is spent on implementing the interactions of the individual elements of the user interface with each other (e.g. defining menu entries, button actions, etc.). In order to reduce these efforts, functionality has been bundled into different modules (e.g. OpenGL rendering, force field methods, an interface to the scripting language, etc.), which can be freely combined to an application. These modules automatically connect to each other and thus allow the user to add further functionality with as little as a single line of code.

To this end, we have designed a set of base classes describing the interactions of the interface elements (see Figure 5). The two most important components in this design are MainControl and ModularWidget. MainControl is the application's main window (and as such it is derived from QT's QMainWindow). ModularWidget is a common

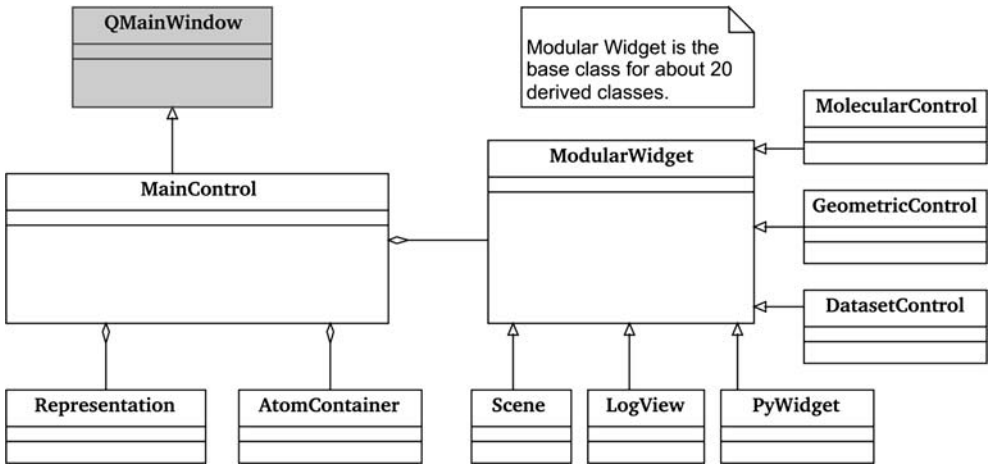

Figure 5. UML diagram of BALLView's core architecture: The MainControl is the main window of every BALLView application and the container for all loaded molecules and representations. It also connects the modular widgets with the messaging system. The classes derived from ModularWidgets, which are shown in this diagram, can also be found in Figure 1. The class QMainWindow stems from the QT-library.

base class for all the interface elements contained in the application's main window. The main window contains just the most essential data structures: a set of structures and a set of representations (i.e. geometric models). Additional functionality (e.g. reading and writing of structures, OpenGL visualization, etc.) can be easily added to the main window by instantiating one of the classes derived from ModularWidget. The following code snippet illustrates the use of these modular widgets:

```
Mainframe::Mainframe(...)
{
  new LogView(this);            // widget (1) in Fig. 1
  new DatasetControl(this);     // widget (2) in Fig. 1
  new PyWidget(this);           // widget (3) in Fig. 1
  new MolecularControl(this); // widget (4) in Fig. 1
  new GeometricControl(this); // widget (5) in Fig. 1
  new Scene(this);              // widget (6) in Fig. 1
}
```

This example creates an application with the full set of widgets as shown in Figure 1. These few lines of code (header includes were omitted for brevity) create a fully-fledged molecular structure viewer. Additional functionality can easily be added by instantiating further modular widgets.

As soon as a Modular Widget registers with the MainControl, it adds menu entries and preferences dialogs to the application. Similarly, all these widgets add new tabs to the application's preferences dialog. Users can easily implement new modular widgets and thus add new functionality to their viewer applications.

With the help of the object-oriented scripting language Python, the functionality of the viewer can be extended at runtime. For each C++ class in BALL we provide a corresponding Python class with virtually identical interface. These Python classes are semiautomatically generated by the wrapper generator SIP [26]. The use of Python can significantly reduce the development time for new methods because Python, as an interpreted language, does not require recompilation of the code between changes. Furthermore, Python scripts can be used to automate recurring tasks. A simple example is given below.

```
pdb = PDBFile("bpti.pdb")
S = System()
pdb.read(S)
getMainControl().insert(S)
getMolecularStructure().addHydrogens()
getMolecularControl().applySelector("element(H)")
amber = getMolecularStructure().getAmberFF()
amber.setup(S)
m = SteepestDescentMinimizer(amber)
m.minimize(100)
```

This script reads a molecule from a PDB file, adds the missing hydrogen atoms and selects these. Next, a steepest-descent minimization is performed for a hundred steps on the hydrogen atoms. Another common use case for the Python interface is to implement user defined visualization capabilities. The BALLView website contains a forum, where users can share these scripts.

Conclusion

BALLView is a new visualization framework, providing a flexible and intuitive interface to the molecular modeling functionality of our C++ software library BALL, which has been in use since 1996. It provides a wide range of functionality in electrostatics, molecular mechanics, solvation methods, and many other areas of structural bioinformatics. BALLView is a powerful and convenient molecular viewer. In addition, advanced users can easily expand and adapt it to their own needs. In doing so, they can benefit from the rapid prototyping capabilities and the embedding of functionality in one common graphical user interface, which considerably accelerates the development of new techniques.

Furthermore, BALLView's easy-to-use interface makes it an ideal tool for teaching structural bioinformatics and molecular modeling. It allows a faster introduction to the key issues in structural biology and structural bioinformatics. We use BALL and BALLView as a basis for student exercises on implementations and algorithms. Here, tasks which could not be performed before due to the amount of program code necessary, can now be realized with a few lines of Python or C++ code.

In contrast to most compareable software tools, BALL and BALLView are Open Source software and available free of charge. Documentation is provided in HTML format and the project's website offers support with forums and mailing lists. Furthermore, BALL and BALLView are available for almost all major platforms and operating systems.

Currently, we are working on adding more modeling features, in particular functionality for editing molecules, aligning of structures, and detection of binding pockets. We are also developing an interface for protein-protein and protein-ligand docking, which will be contained in the next major BALLView release. Furthermore, we are currently implementing the Merck Molecular Force Field (MMFF94 [22]) and other force fields which are well suited for designing and modeling ligands.

Availability

BALL and BALLView are Open Source software available under the GNU Public License (BALLView) and the Lesser GNU Public License (BALL). The source code and precompiled binaries and installers for various operating systems including Linux, MacOS X, and Microsoft Windows are available from our website at http://www.ballview.org.

Acknowledgements

This work was supported in parts by Deutsche Forschungsgemeinschaft grants BIZ 1/1-3, BIZ 4/1-1 and LE 952/2-3. We want to thank the following student workers and fellow team members for their efforts: Andreas Bertsch, Andreas Kerzmann, Anne Dehof, Bettina Leonhardt, Carla Haid, Christian Bender.

References

1. Humphrey, W., Dalke, A. and Schulten, K., VMD-Visual Molecular Dynamics. J. Mol. Graphics, 14 (1996) 33.
2. DeLano W.L. The PyMOL molecular graphics system, 2002.
3. Discovery Studio (2005): http://www.accelrys.com/products/dstudio/index.html.
4. Sayle, R. and Milner-White, E.J., RasMol: Biomolecular graphics for all. Trend. Biochem. Sci. (TIBS), 20(9) (1995) 374.
5. Guex, N. and Peitsch, M.C., SWISS-MODEL and the Swiss-PdbViewer: An environment for comparative protein modeling. Electrophoresis, 18 (1997) 2714–2723.
6. AVS visualization software (2005): http://www.avs.com.
7. Open MOIV: http://www.tecn.upf.es/openmoiv/.
8. Open Inventor: http://oss.sgi.com/projects/inventor/.
9. SYBYL 7.0, Tripos Inc., 1699 South Hanley Rd., St. Louis, Missouri, 63144, USA.
10. Molecular Operating Environment (MOE) (2005): http://www.chemcomp.com/.
11. Kohlbacher, O. and Lenhof, H.P., BALL - Rapid Software Prototyping in Computational Molecular Biology. Bioinformatics, 16(9) (2000) 815–824.
12. Python scripting language (2005): http://www.python.org.
13. QT library (2005): http://www.trolltech.com.
14. Nicholls, A., Sharp, K. and Honig, B., Protein folding and association: Insights from the interfacial and thermodynamic properties of hydrocarbons. PROTEINS, Struct. Func. Genet., 11(4) (1991) 281ff.
15. Nicholls, A. and Honig, B., A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. J. Comp. Chem., 12(4) (1990) 435–445.
16. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. and Bourne, P.E. The Protein Data Bank. Nucl.Acids Res., 28, (2000) 235–242.
17. MOPAC (2005): http://www.cachesoftware.com/mopac/index.shtml.
18. POVRay renderer (2005): http://www.povray.org.
19. Boghossian, N.P., Kohlbacher, O. and Lenhof, H.P., Rapid software prototyping in molecular modeling using the Biochemical Algorithms Library (BALL). J. Exp. Algorithmics, 5 (2000) 16.
20. Cornell, W.D., Cieplak, P., Bayly, C.I., Gould, I.R., Merz, K.M., Ferguson, D.M., Spellmeyer, D.C., Fox, T., Caldwell, J.W. and Kollman, P.A., A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J.Am.Chem.Soc., 117 (1995) 5179–5197.
21. Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S. and Karplus, M., CHARMM: A program for macromolecular energy minimization and dynamics calculations. J. Comput. Chem., 4 (1983) 187–217.
22. Halgren, T.A., Merck molecular force field: I. basis, form, scope, parameterization and performance of MMFF94. J. Comp. Chem., 17 (1996) 490–519.
23. Kabsch, W. and Sander, C., Dictionary of protein secondary structure: Pattern recognition of hydrogen bonded and geometrical features. Biopolymers, 22 (1983) 2577–2637.
24. Boghossian, N.P., Kohlbacher, O. and Lenhof, H.-P. BALL: Biochemical Algorithms Library. In Algorithm Engineering, 3rd International Workshop, WAE'99, Proceedings, volume 1668 of Lecture Notes in Computer Science (LNCS), pages 330–344. Springer, Heidelberg, 1999.
25. OpenGL library (2005): http://www.opengl.org.
26. SIP (Python bindings generator) (2005): http://www.riverbankcomputing.co.uk/sip.