

amiraMol™ 0.9

A Molecular Visualization Extension for Amira

User's Guide and Reference Manual

Copyright Information

©1995-2001 Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany
©1999-2001 Indeed - Visual Concepts GmbH, Berlin, Germany

All rights reserved.

Amira is being jointly developed by Konrad-Zuse-Zentrum für Informationstechnik Berlin and by Indeed - Visual Concepts GmbH. It is based on the latest release of the Open Inventor toolkit from TGS Template Graphics Software, Inc.

Indeed - Visual Concepts GmbH has selected TGS Template Graphics Software, Inc., as the worldwide distributor for Amira.

This document contains proprietary information of Konrad-Zuse-Zentrum für Informationstechnik Berlin and Indeed - Visual Concepts GmbH, and is protected by German and international law. The contents of this document may not be translated, copied, or duplicated in any form, in whole or in part, without the express written permission of Konrad-Zuse-Zentrum für Informationstechnik Berlin or Indeed - Visual Concepts GmbH.

The information contained in this document is subject to change without notice. Neither Konrad-Zuse-Zentrum für Informationstechnik Berlin nor Indeed - Visual Concepts GmbH or their employees shall be responsible for incidental or consequential damages resulting from the use of this material or liable for technical or editorial omissions or errors made herein.

Trademark Information: Amira is a registered trademark of Konrad-Zuse-Zentrum für Informationstechnik Berlin. AmiraDev is a trademark of Indeed - Visual Concepts GmbH. Hewlett-Packard, HP, and HP-UX, are registered trademarks of Hewlett-Packard, Inc. Windows is a registered trademark of Microsoft Corporation. Visual C++, Visual Studio, Developer Studio, and Internet Explorer are trademarks of Microsoft Corporation. Netscape is a trademark of Netscape Communications, Inc. OpenGL is a registered trademark of Silicon Graphics, Inc. Open Inventor, is a trademark of Silicon Graphics Inc. Solaris is a registered trademark of Sun Microsystems, Inc. OpenWindows, Sun, and SunOS are trademarks of Sun Microsystems, Inc. OSF/Motif and OSF/1 are trademarks of Open Software Foundation, Inc. UNIX is a registered trademark of UNIX Systems Laboratories, Inc. All other products and company names are trademarks or registered trademarks of their respective companies.

Contents

I	User's Guide	5
1	Introduction	7
2	Overview of AmiraMol	9
2.1	Data Types: Molecule - Trajectory - TrajectoryBundle	9
2.2	Topology: Groups - Levels	9
2.3	Modules for the Visualization of Molecules	9
2.4	Coloring	10
2.5	Selection	10
2.6	Alignment	10
2.6.1	Mean squared distance alignment	11
2.6.2	Mean distance alignment	11
2.6.3	Sequence alignment	11
2.7	Modules for Time-dependent Data: ConfVol - MeanMolecule - ClosestTimeStep	11
3	First steps with molecular visualization in Amira	13
3.1	Getting Started with Molecular Visualization	13
3.1.1	Loading data into the system	13
3.1.2	Selecting a molecule from the bundle	14
3.1.3	Displaying the molecule with the MoleculeView	14
3.2	Other Viewing Modules	15
3.2.1	BondAngleView	15
3.2.2	Changing the color scheme	15
3.2.3	Secondary Structure View	16
3.3	Selection Browser	18
3.3.1	Opening the browser and connecting viewing modules	18
3.3.2	Changing the appearance of MoleculeView and BondAngleView	18
3.3.3	Highlighting and masking with the selection browser	19
3.4	Molecular Interface	20
3.4.1	Defining groups	20
3.4.2	Computing a label field	21

3.4.3	Computing the interface surface with the GMC	21
3.4.4	Visualizing the interface	21
3.5	Alignment of molecules	22
3.5.1	Sequence alignment	22
3.5.2	Align molecules by using the mean distance criteria	22
3.6	Measurement	23
II	Reference Manual	25
4	Alphabetic Index of Modules	27
4.1	Align Molecules	27
4.2	Align Sequences	28
4.2.1	Align View	30
4.3	BondAngleView	30
4.4	ClosestTimeStep	31
4.5	Conformation Volume	32
4.6	MeanMolecule	34
4.7	Measurement	35
4.8	Molecular Interface	36
4.9	MolecularSurface	37
4.10	MoleculeView	38
4.11	Observables	40
4.12	SecStructureView	41
5	Alphabetic Index of Components	45
5.1	Molecule Selection Browser	45
6	Alphabetic Index of Data Types	51
6.1	MolTrajectory	51
6.2	Molecular Field	51
6.3	Molecule	52
7	Alphabetic Index of File Formats	55
7.1	MDL	55
7.2	PDB	55
7.3	PSF/DCD (CHARMM)	56
7.4	Tripos	56
7.5	UniChem	56
7.6	ZIB Molecular File Format	56

Part I

User's Guide

Chapter 1

Introduction

This document describes AmiraMol, a molecular visualization extension for Amira.

Amira is a 3D visualization and modelling system. It allows you to visualize scientific data sets from various application areas, e.g. medicine, biology, chemistry, physics, or engineering. 3D objects can be represented as grids suitable for numerical simulations, notably as triangular surface and volumetric tetrahedral grids. Amira provides methods to generate such grids from voxel data representing an image volume, and it includes a general purpose interactive 3D viewer.

The AmiraMol extension adds powerful tools for molecular visualization to the Amira system. AmiraMol contains support for standard molecular file formats, tools for visualisation processing and analysis of static molecules and molecular trajectories (time series).

This document gives a step-by-step introduction to molecular visualization with Amira. The second part contains a reference manual for all AmiraMol modules. The following tutorials do not explain all the Amira basics so you may want to refer to the Amira User's Guide for some details.

Chapter 2

Overview of AmiraMol

In the following we will give an overview of Amira's capabilities for handling molecules and molecular dynamics trajectories. After an explanation of structure and interdependence of the molecular data structures we will illustrate display modules followed by concepts for atom based coloring, masking and alignment. Finally, modules for visualizing configurational densities like molecular conformations are described.

2.1 Data Types: Molecule - Trajectory - TrajectoryBundle

Several molecular file formats including PDB can be read and written.

Single molecular configurations are represented as data objects of type *Molecule*. This kind of object can either be created by loading a file containing a single molecular configuration or otherwise by attaching it to an object of type *MolTrajectory*, in which case it will act as a projector extracting one of the trajectory's time steps.

MolTrajectory represents a trajectory, i.e. a series of molecular configurations. Again two cases of existence are possible. Either the trajectory was loaded directly from a file. Alternatively it can be attached to an object of type *MolTrajectoryBundle* singling out one of the trajectories contained in that bundle.

A trajectory bundle reflects the case of a file holding more than one molecular trajectory. If such a file is loaded into Amira, an object of type *MolTrajectoryBundle* will be created. The single trajectories can be accessed by attaching a *MolTrajectory* object.

2.2 Topology: Groups - Levels

An object of data type *Molecule* contains information about the structure of a molecule and the atom coordinates of one of its configurations. The mandatory part of structural information concerns the number and types of all atoms contained in the molecule. All the topological information is organized in levels which are cliques of groups. Each group contains other groups or atoms. The easiest example is the level *bonds*, which consists of groups of two atoms.

Some levels can build hierarchies. For example a residue consists of a number of atoms, and a couple of residues may form a secondary structure. These levels and groups can be defined by file readers or interactively via Tcl-commands.

2.3 Modules for the Visualization of Molecules

Molecules can be visualized with the *MoleculeView*, the *BondAngleView*, or the *SecStructureView*. Example outputs of these modules are shown in Figure 2.1.

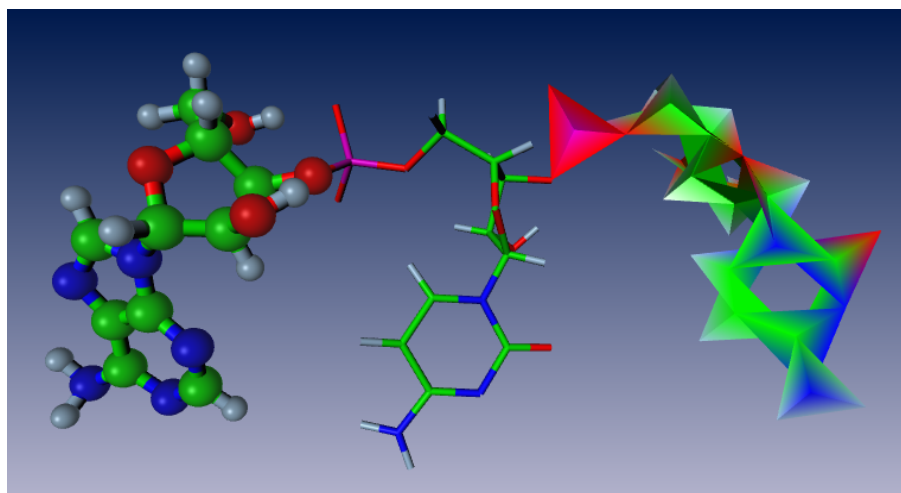


Figure 2.1: MoleculeView and BondAngleView display the molecule simultaneously.

In addition, there are two modules for generating molecular surfaces. The *MolecularInterface* module can be used to generate intramolecular interfaces, such as between single atoms or residues. Also, the interface between two molecules can be computed. The *MolSurface* permits the generation of the *van-der-Waals surface* of a molecule, as well as the generation of the *solvent accessible surface*.

2.4 Coloring

The atom oriented modules *MoleculeView*, *BondAngleView* and *ConformationVolume* allow a coloring on a per atom basis, i.e., each atom is assigned a color. Balls in the *MoleculeView* will have the corresponding colors. Sticks will be split into halves colored according to their incident atoms' colors. In the *BondAngleView* atom colors are assigned to the vertices and color is interpolated over the triangles.

To determine the coloring you can choose a level, like atoms, functional groups, residues, and secondary structures. Furthermore, you can choose one of the level's attributes.

2.5 Selection

Various tasks require the selection of atoms from a molecule. The most prominent are the alignment of molecules and the selective display of molecule parts. Amira offers two ways of selection. The user can either select atoms interactively in the viewer window via *MoleculeView* or *BondAngleView*. Alternatively, selection can be done via the molecule's *selection browser*, which can be opened by pressing the *Show button* of the molecule.

2.6 Alignment

To compare the structures of several molecules it is necessary to bring them into a suitable geometric arrangement to each other, because their relative positions in the common coordinate system are generally meaningless. This provides the opportunity to prescribe a variety of arrangements and thereby to investigate different aspects.

2.6.1 Mean squared distance alignment

Amira offers three main modes of operation, that can be used via *Molecule*, *ConformationVolume* and *MeanMolecule*:

original: Atom coordinates are left as they are. Relative orientations and positions are displayed as prescribed in the data.

center of gravity: For every time step of a trajectory the center of gravity is computed and put to the coordinate center by applying a translation.

align to master: This mode requires a *master* molecule which is used as a reference to which other molecules, the *slaves* are aligned. We consider correspondences between atoms from the reference molecule and atoms from the molecule to be aligned with it. The alignment is done by minimizing the sum of squared distances between all corresponding atoms.

If master and slave have the same structure, every atom exists in both molecules and hence a canonical correspondence exists. The necessary part of user interaction is to determine the set of atoms to be considered. Beyond taking all atoms into account, an explicit subset can be specified by selecting atoms either in the slave or in the master molecule.

Irrespective of a structural identity of master and slave the correspondence can be determined by selecting a sequence of atoms in the master molecule and another sequence of the same length in the slave molecule. The order of selection thereby defines the correspondence. The first atom in the master corresponds to the first in the slave and so forth. If the molecules have different structures, this is the only way to align them.

2.6.2 Mean distance alignment

Apart from the above mentioned alignment procedure, there exists a module that offers the possibility to align two molecules by using the *mean distance* criteria instead of *mean squared* distance. See *AlignMolecules* for more information.

2.6.3 Sequence alignment

Amira allows you to *align the sequences* of two molecules. Three algorithms are available, which can be used depending on the kind of data and your intention: local, semi-global, and global alignment. The algorithms work both for proteins and ribonucleic acids, whereby t-RNA molecules are currently not supported, because they contain modified bases. This module can be very well used in conjunction with the module *AlignMolecules*.

2.7 Modules for Time-dependent Data: ConfVol - MeanMolecule - ClosestTimeStep

A *MolTrajectory* can be visualized via animation of single timestep representations by attaching a *Molecule* module and then attaching a *MoleculeView* or *BondAngleView*. The animation is controlled via the *Molecule*'s port *Time*.

For *MolTrajectories* that represent metastable conformations *MeanMolecule* and *ConformationVolume* can be used. The module *MeanMolecule* aligns all steps of a trajectory and computes a mean molecule by averaging every atom coordinate over all timesteps. Using the *ClosestTimeStep* module the member of the trajectory can be found that is most similar to this mean molecule and can therefore be used as a representative of the metastable conformation.

The module *ConformationVolume* gives an impression of the fuzziness of the conformation by computing some kind of probability density for the positions of atoms and bonds within a molecular trajectory. This density can then be visualized with *Isosurface* or *Voltex*.

Chapter 3

First steps with molecular visualization in Amira

This chapter leads you step by step into the visualization of molecular data sets. Amira is not only able to display a 3D image of a molecule but also provides tools to investigate its distinct parts and properties. After reading the "getting started" introduction you may continue with any of the following independent tutorials.

- *Getting started* - first steps
- *Further viewing modules* - visualizing geometric properties
- *Selection Browser* - exploring the molecule
- *Molecular Interface* - creating partial surfaces
- *Alignment of molecules* - comparing molecules
- *Measurement* - measuring distances and angles within a molecule

Note: If you want to visualize your own data, please refer to the section about data import in the Amira User's Guide. That section contains some general hints on how to import data sets into Amira.

3.1 Getting Started with Molecular Visualization

In this section you will learn how to

1. load a molecular demo data set into the system
2. select a particular molecule from a bundle
3. view the molecule with the *MoleculeView*

3.1.1 Loading data into the system

In the following introduction we will consider a simple example to explain the basic functions of the *MoleculeView*. Our example will be a file containing a bunch of alkanes called `alkane.top`. If you instead use your own data, such as a PDB-File, please continue with Section 3.1.3.

- Load the file `alkane.top` into the object pool from the directory `data/molecules/alkane`.

As result a green icon labelled `alkane.top` will appear in the object pool. The green icon represents a bundle of molecules or molecular trajectories. Click on the green data icon with the right mouse button and choose *MolTrajectory* from the popup menu. Another green icon labelled *MolTrajectory* will appear in the object pool. A blue line connects the two icons.

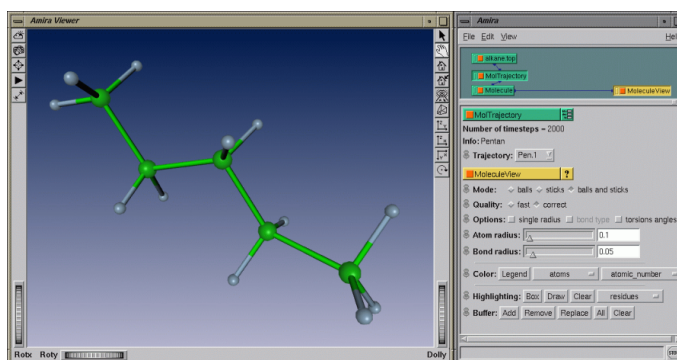


Figure 3.1: A pentane molecule is displayed in the viewer using the MoleculeView.

3.1.2 Selecting a molecule from the bundle

Since the icon **MolTrajectory** is now selected, information about this object is displayed in the working area. There are two information ports, one displaying the number of time steps in the trajectory, the other displaying the name of the molecule. In addition, there is a menu labelled *Trajectory*. It contains three entries: *But.1*, *Pen.1*, and *Hex.1*. Choose the second entry, which is pentane. Now right-click on the green icon **MolTrajectory** in the object pool and choose the entry *Molecule* from the popup menu.

3.1.3 Displaying the molecule with the MoleculeView

As result of the previous steps another green icon labelled **Molecule** has appeared in the object pool. If you right-click on this icon, a menu pops up from which you can select the entry **MoleculeView**. This action creates both a yellow icon in the object pool and an image in the 3D viewer. Furthermore, the GUI of the module appears in the working area.

Once the molecule is displayed in the viewer, you can rotate and zoom the molecule. In order to do so, the viewer needs to be in viewing mode, i.e. the mouse cursor is displayed as a hand, which is the default. If you want to rotate the molecule, left-click in the viewer and move the mouse. Zooming is enabled when you click the left and the middle mouse buttons simultaneously. Moving the mouse up zooms out, moving the mouse down zooms in.

Whenever the **MoleculeView** is active, the little square on the yellow icon **MoleculeView** is orange. You can deactivate the module by clicking on that square with the left mouse button. Now let's explore the following ports in the working area:

Port Mode

Choose another mode to see both atoms and bonds of the molecule, or just atoms. If atoms are shown, use the port *Atom Radius* to adjust the size of the atoms according to your preferences.

Port Quality

This port offers the possibility to display a correct image of the molecule. Consider the trade-off between correct representation and a slower rendering performance. Use the *fast* mode, if you want to display a large molecule. If your graphics hardware is fast enough, use the *correct* mode. You can increase the complexity of the scene by typing

```
MoleculeView complexity setValue 0.5, followed by
MoleculeView fire
```

Port Color

Click on the button *Legend* to display a small window decoding the colors you see in the viewer window. Choose *residues* from the first popup menu and *type-id* from the second to investigate different components of your molecule. Change the color map by right-clicking on the color bar of the port *Discrete CM* that appears below the Color port.

Port Highlighting

Press the *Draw* button and draw a line in the viewer window around a group of atoms you want to select. You will see a red box around each of the selected atoms.

Change to interaction mode by clicking the icon displaying a small black arrow in the upper right corner of the Viewer window. Your cursor changes its appearance from a hand to an arrow. You can now pick one of the atoms to get some information about the atom listed in the Amira console window. There is also the possibility to select a group of atoms, such as a residue, or a chain (see the description on *MoleculeView*). In order to unhighlight all atoms, press the button *Clear* of the *Highlighting* port.

Port Buffer

This port allows you to hide parts of the molecule in the viewer window. Highlight a few atoms as described in the previous section, and then press the button *Replace*. All atoms, that were not highlighted will now have disappeared. Play around with this port in conjunction with the *Highlighting* port.

3.2 Other Viewing Modules

In Section 3.1 we have seen how to visualize molecules with the *MoleculeView*.

In this section you will learn how to

1. view a molecule with the *BondAngleView*
2. apply different coloring schemes to the molecule
3. use the *SecStructureView*

We start by loading the data file `2JEL.pdb` from the directory `data/molecules/pdb`.

3.2.1 BondAngleView

As the name suggests, the *BondAngleView* displays all bond angles in a molecule. For each triple of atoms connected by two bonds, a triangle will be drawn. Create the module by right-clicking on the green icon labelled `2JEL.pdb`, and choose *BondAngleView* from the submenu *Display*. In order to display the molecule as depicted in Figure 3.2, you need to rotate and zoom the molecule, and change the *Draw Style* from *outlined* to *shaded*.

3.2.2 Changing the color scheme

Currently the triangles of the *BondAngleView* are colored according to the atom types, i.e. the *atomic number*. We will now explore, how to use different color schemes.

- Click on the second menu of the port *Color* and select the entry *index*.

As result, the whole molecule will now appear blue. This is because, the default color map is a constant one.

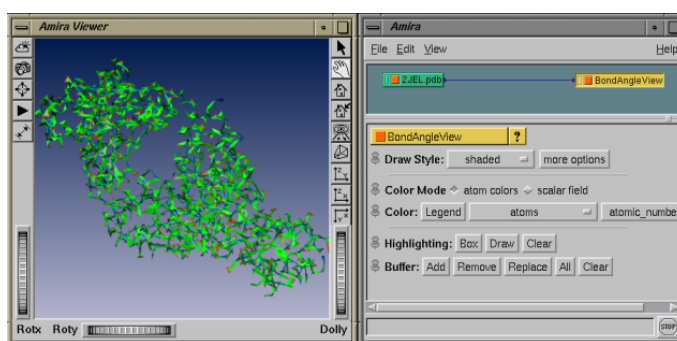


Figure 3.2: Creating the Bond Angle View.

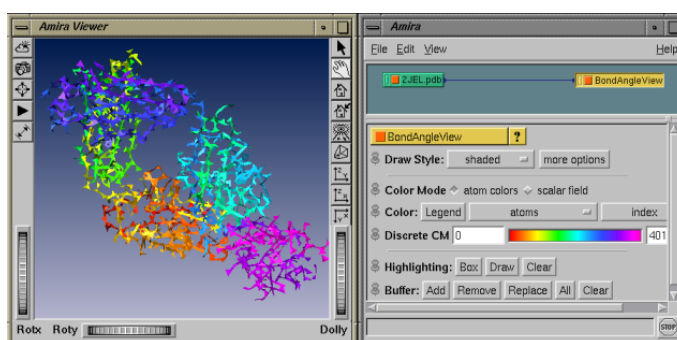


Figure 3.3: Changing the color scheme. One can easily follow the sequence of atoms in the molecule by means of the colormap.

- Now, right-click on the color bar of the port *Discrete CM*.

A list of pre-defined color maps appears.

- Select the entry *standard.icol* from the list.

The molecule will now be displayed in pink color. This is because the color map range currently is [0,1], which means, that the atom index 0 is mapped to red and the index 1 as well as all indices larger than 1 are mapped to pink. If you have selected the *standard* color map, the optimal range will be printed in the console window. For this molecule and the specified color scheme, the optimal range is [0, 4013].

- So you only need to change the value in the second text field to 4013.

The result is depicted in Figure 3.3.

Now let's try out yet another color scheme.

- Change the entry of the first menu to *residues* and of the second menu to *type*.
- Change the color map to *physics.icol*.
- Set the range according to what is printed in the console window.

The result is shown in Figure 3.4.

3.2.3 Secondary Structure View

In order to create the module *SecStructureView*, right-click on the icon *2JEL.pdb* and choose the entry *SecStructureView* from the submenu *Display*. Now both modules, the *BondAngleView* and the *SecStructureView* display the molecule in the viewer. Since in this section we want to concentrate on the latter, we

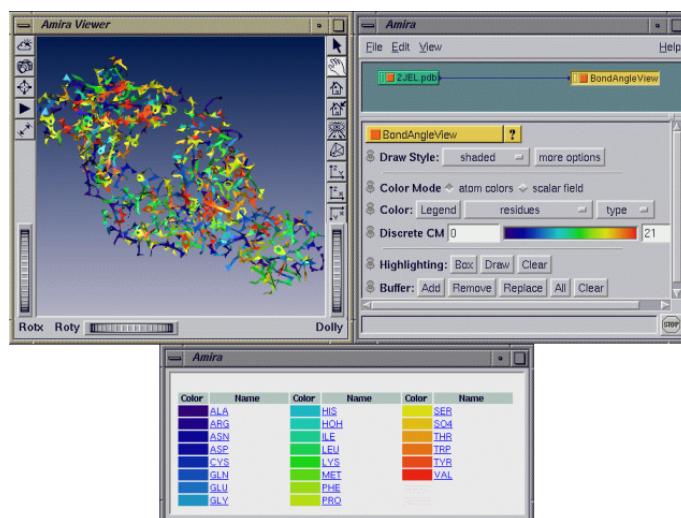


Figure 3.4: Coloring the atoms according to residue type. The window at the bottom decodes the colors.

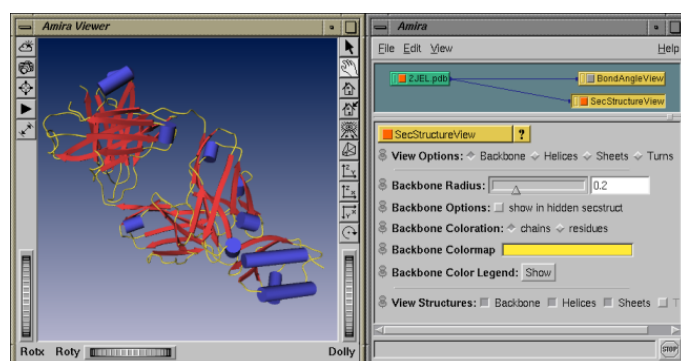


Figure 3.5: View the molecule with the SecStructureView.

want to switch off the *BondAngleView*. There are two possibilities to do this. Either we select the icon *BondAngleView* in the object pool by clicking on it, and afterwards press **Ctrl+X**, which removes the *BondAngleView*. Or we click on the orange square inside the icon *BondAngleView*, which does not remove the object from the object pool, but hides it in the viewer. We prefer the second procedure, since it allows us to quickly change between the viewing modules.

If you have done the previously described steps, you will observe that your backbone is colored grey instead of yellow as in Figure 3.5. In order to change this,

- double-click on the color bar of port *Backbone Colormap*.

Immediately, a *Color Dialog* will appear.

- Pick yellow from the *Custom Colors* and pull it to the color field labelled *Current Color* in the color dialog.
- Press **OK** and the backbone should now be yellow.

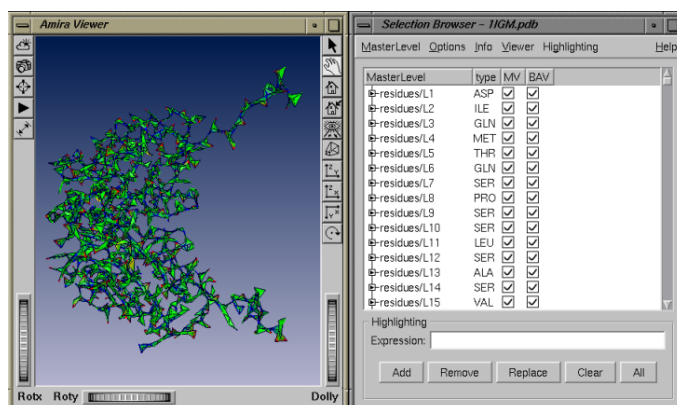


Figure 3.6: Left: MoleculeView and BondAngleView display the molecule simultaneously. Right: Selection browser after connecting two viewing modules to the molecule.

3.3 Selection Browser

You already know how to load files and how to use the *MoleculeView*. In this section you will learn the basics about the *Molecule Selection Browser*, which is a very important feature for the investigation of a molecule. The *selection browser* is a flexible tool for selecting those parts of the molecule the user is interested in. Moreover, it allows to easily combine different viewing modules into one viewer.

3.3.1 Opening the browser and connecting viewing modules

To get started, load the file `1IGM.pdb` from the directory `data/molecules/pdb`. In the working area below the object pool, information about the loaded molecule will be displayed, such as the number of atoms, bonds, residues, and secondary structure elements. Furthermore, you will find two ports, *Alignment* and *Selection Browser*. The first port is currently not of interest to us.

- Press the button *Show* of the second port to open the selection browser which appears in the bottom left corner of the screen.

The scroll view of the browser currently contains two columns, one with the heading *MasterLevel*, the other one with the heading *type*. In the first, all residues of the molecule are displayed. The second column shows the residue type.

- Connect a *MoleculeView* to the icon `1IGM.pdb`.
- Connect a *BondAngleView* to `1IGM.pdb` by right-clicking the icon and choosing *BondAngleView* from the submenu *Display*.

You will observe two new columns in the selection browser, representing the *MoleculeView* (*MV*) and the *BondAngleView* (*BAV*), respectively (Figure 3.6).

3.3.2 Changing the appearance of MoleculeView and BondAngleView

In order to get a similar image as in Figure 3.8, we first need to set the ports in the *MoleculeView* and the *BondAngleView* correctly (Figure 3.7).

Let's start with the *MoleculeView*.

- Set the port *Mode* to *balls*.
- Set the port *Quality* to *correct*.

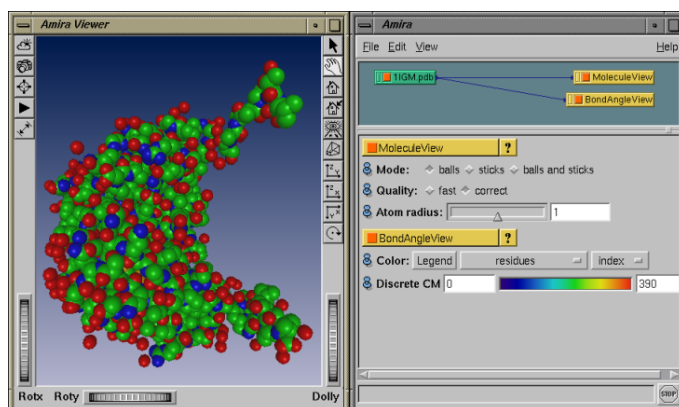


Figure 3.7: Setting the ports in the viewing modules correctly.

- Set the *Atom Radius* to 1.0.

For the *BondAngleView* continue as follows:

- Select the entry *residues* from the first menu of port *Color*.
- From the second menu of port *Color* select the entry *index*.
- Right-click on the color bar of the port *Continues CM* and select the color map *physics.icol*.
- Set the max range value of the color map port to 390.

We now only see the *MoleculeView*, since the triangles of the *BondAngleView* are hidden by the van der Waals spheres. In order to combine the two viewing modules to one image, continue with the next section.

3.3.3 Highlighting and masking with the selection browser

We now want to use the selection browser to display parts of the molecule with the *MoleculeView* and the rest with the *BondAngleView*.

- Select *chains* as new master level from the *MasterLevel* menu.

As result you will now see three entries in the column *MasterLevel*: *chains/L*, *chains/H*, and *chains/W*. This means that the level *chains* contains three groups. In order to view the group *chains/L* with the *MoleculeView* and the other two groups with the *BondAngleView*,

- remove the ticks for *chains/H* and *chains/W* from the column with heading *MV* by clicking on them.

To deselect the group *chains/L* for the *BondAngleView*,

- click on the respective box in the column *BAV*.

Now, you should see an image that is pretty close to that in Figure 3.8. However, the *BondAngleView* displays more triangles than in the Figure. The *BondAngleView* in Figure 3.8 only displays *backbone* atoms. In order to achieve this

- right-click on heading labelled *BAV*.

A popup menu as in Figure 3.8 should appear.

- Select *Backbone* from the submenu *Restrict to*.

The side-chain atoms should not be visible anymore.

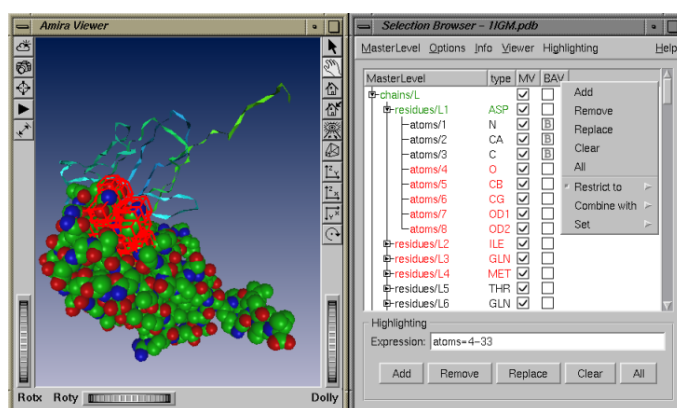


Figure 3.8: Using the Selection Dialog window.

Next, you should explore the group *chains/L* a bit further.

- Click on the little icon left to it.

After this action the residues contained in chain *L* will have appeared.

- Type `atoms=4-33` in the text field labelled *Expression*, below the scroll window, and press the *Replace* button.

The atoms 4 to 8 and the residues *L2*, *L3*, and *L4* are highlighted in red. The residue *L1* and the chain *L* are highlighted in green. The color *red* means, the whole group is selected, i.e. all its elements. *Green* denotes the group is partially selected.

To get accustomed to the browser, play around with it. If you need further information, just press the button *F1* in the browser window or see [here](#).

3.4 Molecular Interface

This tool is particularly interesting to visualize contact areas between parts of a single molecule, or between different molecules, e.g. enzymes and their ligands. In this example we will consider the second case.

- Please load the file `t1gg.mol` from the directory `data/molecules/r_nuclease`.

3.4.1 Defining groups

In this section we will create a new level, called *interface*, for the molecule and define two groups of the level, *substrate* and *nuclease*, respectively.

- Select the icon `t1gg.mol` in the object pool by clicking on it.
- Now click into the console window to activate it and press the *TAB*-key.

The name of the selected icon, i.e. `t1gg.mol` should appear in the console window. If it does not,

- please type it.
- Behind `t1gg.mol` type `defregexp interface/substrate atoms=1460-1526`.

The following should now be written in the console window

```
t1gg.mol defregexp interface/substrate atoms=1460-1526.
```

-
- Press the ENTER-key.

You have now defined the group *substrate* in the level *interface*. The group consists of the atoms 1460-1526. We will now define the group *nuclease*, containing the atoms 0-1459.

- Press the TAB-key again followed by typing `defregexp interface/nuclease atoms=0-1459`

If you now list all levels by typing

- `t1gg.mol list`

you will find a level *interface* containing two groups.

- Type `t1gg.mol list interface`

and all groups of the level *interface* will be printed in the console window.

3.4.2 Computing a label field

We will now create a label field, containing two labels, *substrate* and *nuclease*, which will then be used for the computation of the interface surface. You will learn, how to use the module *MolecularInterface*.

- Right-click the icon `t1gg.mol` and select *Molecular Interface* from the submenu *Compute*.

The GUI of the module *Molecular Interface* should now have appeared in the working area. If it has not,

- select the icon labelled `MolecularInterface`.

Continue by

- choosing *interface* from the menu port *Level* of the *MolecularInterface*.
- Press the button *Generate Grid*.

The result is an object *Interface* of type *UniformScalarField*.

3.4.3 Computing the interface surface with the GMC

With the module *GMC* we will now create an interface surface from the label field.

- Right-click the icon *Interface* and select the entry *GMC* from the submenu *Compute*.
- Press the button *Triangulate* of the *Action* port of the module *GMC* in the working area.

A green icon labelled `Interface.gmc-surf` has appeared in the object pool.

3.4.4 Visualizing the interface

Finally, we will view the computed interface in the viewer window via the *SurfaceView*.

- Right-click on the icon and select *SurfaceView*.

The GUI of the *SurfaceView* module should now be seen in the working area.

- Choose the *Colors*-option *twisted*.
- Select the entry *SubUnit2* from the first menu of the port *Materials*.
- Press the *Clear* button of port *Buffer* followed by the *Add* button of the same port.

You should now see a grey surface in the viewer with some colored patches. The grey surface wraps the nuclease, the colored patches depict the interface between the nuclease and the substrate.

What you see should be similar to the demo Nuclease on the demo pages.

3.5 Alignment of molecules

In this tutorial you will get to know the modules *AlignSequences* and *AlignMolecules*.

The *AlignSequences* tool facilitates the comparison of two sequences. It can be applied to both proteins and nucleic acids except for t-rna molecules containing modified bases. It is planned to extend this module.

Having done the sequential alignment you can use the correspondence gained by the sequence alignment to do a structural alignment of the molecules.

- Please load the files `1IGM.pdb` and `2JEL.pdb` from the directory `data/molecules/pdb/` and connect a *MoleculeView* to each of them.

You should know how to do this from the tutorial *Getting started*. For both *MoleculeViews*

- select *residues* from the first menu of the port *Color*.

A *Discrete ColorMap* with constant color will appear. To distinguish the molecules, choose a different color for one of them. If you don't know how to do this, see the tutorial *Further viewing modules*.

3.5.1 Sequence alignment

- Right-click on the icon `1IGM.pdb` and select *AlignSequences* from the submenu *Compute*.
- Right-click the little rectangle inside the icon *AlignSequences* on the left side.

A popup displaying all connection ports opens.

- Select *MoleculeB* from it and connect the module to the icon `2JEL.pdb` by clicking on it.

There should now be two blue lines connecting the icon *AlignSequences* with the icons `1IGM.pdb` and `2JEL.pdb`, respectively.

- Select the module *AlignSequences* by clicking on its icon in the object pool.

As result, the user interface of the module appears in the working area.

- Choose *semiglobal* from the second menu of port *Align Type*, and press the button *Align Sequences*.

If the alignment has been successful, a window displaying possibly several slightly different alignments will appear.

- Press the *AcceptAll* button.

This action results in the alignments being written to the molecules as levels.

- Type, e.g. `1IGM.pdb list`.

Apart from levels such as *atoms*, *bonds*, *residues*, etc. you will also see levels named *seqAlign1* to *seqAlign10*. `2JEL.pdb` has the same levels with an equal number of groups.

3.5.2 Align molecules by using the mean distance criteria

We can now proceed to use the levels *seqAlign** for a structural alignment. In order to do so,

-
- right-click the icon `1IGM.pdb` and select *AlignMolecules* from the submenu *Compute*.
 - Right-click the little rectangle inside the icon `AlignMolecules` on the left side and connect the module to `2JEL.pdb`.
 - Select the icon `AlignMolecules` to make it appear in the working area.
 - Select any of the levels named *seqAlign** in the port *AlignLevel* and press the *DoIt*-button.

If you want to follow the alignment process

- click the option *show alignment*

before pressing the *DoIt*-button.

Compare your result to the demo.

3.6 Measurement

In this tutorial you will learn how to measure distances and angles between atoms in a molecule. For this tutorial it is necessary that you have done the *Getting started*-tutorial previously. You need to have a time step, i.e. a *Molecule*, loaded from the butane-trajectory, and a *MoleculeView* should be connected to it.

- Select the *MoleculeView* by clicking on its icon in the object pool.

You will now see the *MoleculeView*'s user interface in the working area.

- Select *ball and sticks Mode* and *fast Quality*.
- Next, right-click on the icon `MoleculeView`, and select *Measurement* from the popup menu.

The user interface of the *Measurement* tool should now be visible in the working area. An *Info* port tells you that you need to select 2, 3, or 4 atoms. In order to do so, change into interaction mode by

- pressing the `ESC`-key or by pressing the arrow-icon in the right upper corner of the viewer window.

You can now select single atoms in the viewer by clicking on them. If you click on one atom, the previously selected atom will be deselected. By using the `Ctrl`-key you can select more than one atom.

- Select 2, 3, or 4 atoms and observe what is being displaying in the user interface of the module *Measurement*.

The `Ctrl`-key is also used to deselect atoms.

Part II

Reference Manual

Chapter 4

Alphabetic Index of Modules

4.1 Align Molecules

This module enables you to align two arbitrary molecules to each other. However, the molecules need to fulfil one precondition: they both need to have (at least) one level with the same name and an equal number of groups. In the algorithm the atoms in the corresponding groups will attract each other, whereby the correspondence of the groups is established by the index of the group in the level. You can easily create such levels by using the module *Align Sequences*.

The algorithm works iteratively. Starting from an arbitrary position, a force and a rotation will be calculated at each step resulting from the attraction forces of the atoms. The algorithm stops if the transformation from one step to another is smaller than some epsilon. The alignment found is not necessarily the global minimum of the sum of distances between the corresponding atoms. However, it will always be a local minimum.

If the global minimum is outstanding, it can be assumed that this minimum will be found from each starting position. In order to find out whether there exist other equally good local alignments, there exists the possibility to compute alignments starting from 20 different positions. The different alignments will be stored in a list. You can then compare the alignments visually as well as by the distance between the two molecules.

Connections

MoleculeA [required]

First molecule to which the second molecule will be aligned.

MoleculeB [required]

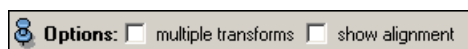
Second molecule to be aligned to the first one.

Slave [optional]

You might connect any data object to this port, if you want to apply the same transformation to it that is applied to the second molecule.

Ports

Options



- *multiple transforms*: If this toggle is not selected, a local minimum will be found starting from the current positions of the molecules. If the toggle is selected, the molecule B will be translated such that the centers of the molecules according to the groups found in the *Align Level* are at the same

position. Starting from this position, 20 rotations will be applied to the molecule B, and these new positions will be used as starting points for the alignment.

- *show alignment*: If this option is set, you can observe the alignment process visually. Otherwise, only the final alignment will be shown.

Transforms



If you compute the alignment with the *multiple transform* toggle switched on, this slider appears. Once the computation is finished, i.e. for all 20 starting positions the alignment was computed, you can step through all alignments that differ by an epsilon. The epsilon can be set by the command *setEpsilon*. Notice, that if there is only one local alignment, it can be assumed that this is also the global one. However, there is no guarantee.

Align Level



Select the level which is used for alignment.

Action



Compute alignment and transform second molecule.

Commands

getEpsilon

Prints the current epsilon that is used for comparison of the transformations.

setEpsilon <value>

Set epsilon to value, which should be smaller than 1.0. Check the current epsilon before setting a new one with the command *getEpsilon*.

4.2 Align Sequences

This module aligns the sequences of two molecules to each other. Proteins as well as nucleic acids can be aligned. Currently, it is not possible to align t-RNA sequences because they contain modified bases deviating from the standard data format. The module will be adjusted and extended in the near future. The output of the alignment is displayed in a separate window. Currently you may choose between *local*, *semiglobal* and *global* alignment. The algorithms are based on the Waterman algorithms for pairwise sequence alignments using a Blossum weighting matrix for protein sequences and a Transition/Transversion weighting matrix for nucleic acids. Consider the following three cases to decide which algorithm to use:

- *local alignment*
The local alignment algorithm is used best if you want to match a short sequence, possibly a sub-sequence or motif of some molecule, against a long one. You may specify a motif (e.g. a promoter sequence) and search for it in the molecule of interest. Beware, the local alignment algorithm is not useful for aligning two long sequences to each other, e.g. two related proteins. You might end up with an incredibly lot of equally evaluated sequences, scattering the molecules in too many pieces. For alignment of two long sequences, you should rather use one of the next two algorithms instead.
- *global alignment with gap function*
In this case, you want to match two related long sequences to each other, e.g. two related enzymes. The algorithm matches the whole sequences punishing long gaps proportionally less than a sheer accumulation of deletions/insertions.

- *semiglobal alignment*

If you have no clue about the relationship of your two test molecules, it is best to use the semiglobal algorithm first because it will scan for the possibly best matching parts of your molecules without having to align the whole sequences as with the global algorithm.

Connections

MoleculeA [required]

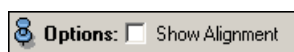
First molecule to be aligned.

MoleculeB [required]

Second molecule to be aligned.

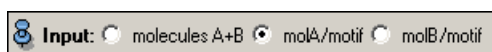
Ports

Options



Currently the only option available is whether to show the alignment or not. This toggle is sensitive if an alignment exists, i.e. has previously been computed.

Input



Which sequences should be taken as input for the alignment. Three options exist:

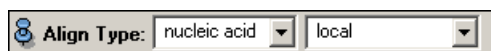
- Molecule A and B.
- Molecule A and the specified motif (see below).
- Molecule B and the motif.

Motif



You may specify a motif which shall be searched for in a longer sequence. This port appears if one of the last two options is chosen in the *Input* port.

Align Type



You may align proteins as well as ribonucleic acids. Specify the type of molecule in the first popup menu. In the second menu you can select one of three algorithms as explained above: *local*, *semiglobal*, or *global* alignment.

Align Sequences



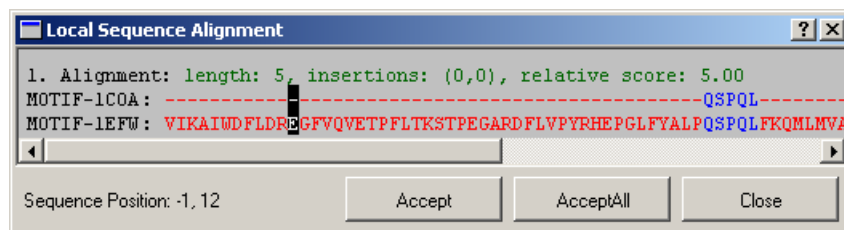
Compute alignment and display it.

Commands

`setLimit <value>`

With this command you restrict the number of alignments gained from one start position of the alignment matrix. The default is 10.

4.2.1 Align View



For each alignment its number and additional information are displayed in the first line. Currently the information includes the length of the alignment, the number of insertions for both sequences and the relative score, i.e. the score divided by the length of the alignment. The last might be interesting when comparing alignments between homologous sequences.

The align view displays corresponding residues in blue color. To get an idea about where in the sequences similarities have been found, the rest of the sequence is displayed as well (in red color). In order to find out where in the sequence you are, just click at the position within the sequence you are interested in. Immediately, the positions for both sequences will be shown in the lower left corner of the window. The clicked position will be marked by a black rectangle.

Apart from just showing alignments, the module *Align Sequences* allows to add alignments as levels to the aligned molecules. However, since there might be a large number of alignments and one might only be interested in a few, the window offers the functionality of selecting alignments. In order to select a single alignment left-click into the row with the label *i. Alignment*. The label will turn black. If you want to select more than one alignment, use the **Ctrl**-key. An alignment can be unselected by just clicking on it again. All currently selected alignments are highlighted by a black frame around the label. Those alignments can be written to the molecules by pressing the *Accept*-button. If all alignments are wished, press the *AcceptAll*-button.

When an alignment is written to a molecule as a new level, all letters of the sequence that are not matched, i.e. that correspond to a blank, are left out. Both molecules will have a new level with exactly the same name and length. Thus, those levels can be used by the module *AlignMolecules*.

4.3 BondAngleView

The *BondAngle View* offers an alternative to the *Molecule View*. This viewing module, however, does not display atoms and bonds but, as the name suggests, bond angles, i.e. for every three atoms connected by two bonds a triangle will be shown. The vertex colors are linearly interpolated across the triangle. Apart from coloring the molecule in analogy to the *Molecule View* the bond angles can be colored according to a specified color field, e.g. the electrostatic potential, and a colormap. Here, apart from interpolating the colors, texture mapping can be applied which interpolates the texture coordinates.

Connections

Data [required]

Molecule to be visualized.

ColorField [optional]

Scalar field, that can be displayed on the bond angle by pseudo-coloring.

Colormap [optional]

Color map needed in conjunction with the color field.

Continuous CM [optional]

The colormap that is used to map values of float attributes to colors.

Discrete CM [optional]

The colormap that is used to map values of discrete attributes, like integers and strings, to colors.

Ports

Draw Style



This port is inherited from the module *ViewBase*.

Colormap



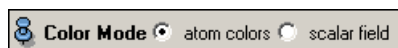
This port becomes visible only if a scalar field has been connected to the color field port.

Transparency



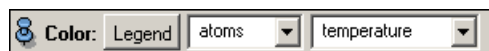
Degree of transparency.

Color Mode



Allows you to switch between "normal" coloring mode and coloring according to a scalar field.

Color



For a detailed explanation see the description of *MoleculeView*.

Continuous CM



See explanation in *MoleculeView*.

Discrete CM



See explanation in *MoleculeView*.

Commands

getOptimalColormapRange

When displaying all atoms the range printed by this command is a good choice as range for the colormap.

4.4 ClosestTimeStep

For a given molecule, this module searches for the molecule within a molecular trajectory with the smallest mean squared distance. The reference molecule and the molecules in the trajectory need to have the same number of atoms.

The module is especially helpful in conjunction with the module for the computation of the *mean molecule*. It allows you to find the time step in the trajectory that best approximates the mean molecule.

Connections

Data [required]

Molecular trajectory for which the mean molecule should be computed.

ReferenceMolecule [required]

The molecule for which the closest time step is searched.

Ports

Action



Start search for the closest time step in the trajectory. The index of the closest time step together with the distance will be printed in the *console window*.

4.5 Conformation Volume

This module enables you to compute some kind of probability density for the positions of atoms and bonds within a molecular trajectory. The input is an object of *Molecule Trajectory*. As output either a scalar field or a color field is generated. In order to visualize the computed density you can, e.g., apply the volume rendering module *Voltex*.

Since the time steps of the molecular trajectory can be arbitrarily rotated and translated, we need to perform an alignment for each time step to fit it best to some chosen reference. This is done internally, but you need to specify how the molecules should be aligned to each other. There are four options among which you can choose. The first uses all atoms for alignment, which is the recommended option. In the second you can select a few atoms (see below for more information). The third and fourth use none and 'center of gravity' alignment, respectively.

For the representation of the molecule two kinds of geometric objects can be used: spheres and sticks. Here, spheres represent the positions of the atomic nuclei and the sticks the existing bonds within the molecule. You can compute the density for sticks, atoms, or sticks and atoms.

Connections

Data [required]

The molecular trajectory for which the density should be computed.

AlignMolecule [optional]

A molecule to which each time step of the trajectory should be aligned.

Continuous and DiscreteColormap [optional]

Those two colormaps are used for the color management of the computed volume, in case a color field should be generated.

Ports

Voxel Size



This value determines the size of a voxel, i.e. its height, width, and depth, of the field storing the probability density.

Grid Dimension

 Grid Dimension: 125 x 116 x 108

The dimension of the field grid resulting from the specified voxel size.

Time Steps

 Time Steps: from 1 to 100

Specifies range of time steps for which the density should be computed.

Align all molecules to time step

 Align all molecules to time step 1

All time steps of the trajectory will be aligned to this time step. This port is invisible if connection *AlignMolecule* is used.

Align Options

 Align ☒ all atoms ☐ selected atoms ☐ cog ☐ no atoms

The default alignment option is to use all atoms of the molecule. If you connect the connection *AlignMolecule* to a molecule, usually a time step of this trajectory too, you may as well align the time steps by using only a few atoms. For this you must view the align molecule with the *Molecule View* module, which allows you to select atoms. The selected atoms are as yet only stored in the align molecule. In order to transfer the selections to this module you need to press the *Get* button of the *Selection* port. There is also the possibility to align the molecule not at all or only to align the 'center of gravity'. This may sometimes be useful but in general is not.

Selection

 Selection:

This port is visible only if *selected atoms* is chosen as alignment option. With the first button you *get* the currently highlighted atoms from the *AlignMolecule*, which will be added to the previous selection, so press *Clear* first to make sure that only the current selection is stored.

Shape

 Shape: ☐ atom spheres ☒ bond cylinders

Choose the geometry used to represent the molecule. If you have a highly varying trajectory the recommended shape is *cylinders*.

Radius Options

 Radius Options: ☒ fixed ☐ atom class depending

Use a unique radius for all atom spheres (*fixed*) or take for each sphere the van der Waals radius scaled by the value of port *Atom Radius* (*atom class depending*).

Atom Radius

 Atom Radius:

Radius of the atom spheres.

Bond Radius

 Bond Radius:

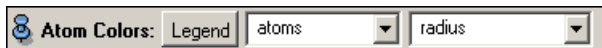
Radius of cylinders representing bonds.

Field

 Field:

This menu contains two options, *Scalar Field* and *Color Field*. Choose the first if you only want the density. If the second option is selected, in addition to the density, the color information will be stored according to the color scheme (see port *Atom Colors*).

Atom Colors



Determines how the field regions will be colored. See the port *Color* in Molecule View.

Colormaps



Those two colormaps are used to color the atoms and sticks according to the selected colorscheme in *Auatom Colors*. See the description in *Molecule View* for a detailed explanation.

Compute



Invoke actions.

- Since for the computation of the grid dimension all molecules need to be aligned, for a long trajectory this might take some time. So it is not desired to recompute the grid dimension all the time. If the user is interested in the size of the grid before invoking the computation of the field, the button *Grid Dimension* must be pressed.
- Press the button *Field* to start the computation of the field. If the grid dimension is not up to date, it will be computed first.

4.6 MeanMolecule

This module computes the 'mean' molecule of a molecular trajectory by averaging the atom positions. In order to accomplish this the molecules need to be aligned to some reference molecule which can either be a molecule from the trajectory or some other molecule which can be connected to the module via the *AlignMolecule* connection port.

By computing the mean molecule several times, each time taking the previous mean molecule as *AlignMolecule* you get a better and better mean molecule.

Connections

Data [required]

Molecular trajectory for which the mean molecule should be computed.

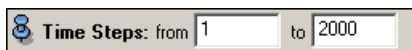
AlignMolecule [optional]

In order to compute the mean molecule, all molecules of the trajectory need to be aligned with respect to a certain molecule. If you don't want to align the molecules to a time step of the trajectory, you need to connect this port to some other molecule. In order to do so, right-click on the little rectangle of the module icon in the *object pool*. As result a popup appears. Select *AlignMolecule* and connect it to a molecule in the object pool.

This connection is also needed if you want to use *selected atoms* for alignment instead of *all atoms*.

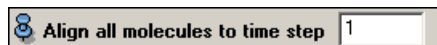
Ports

Time Steps



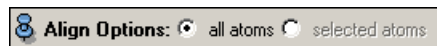
Specify the range of the molecules that are used for the computation.

Align all molecules to time step



If no *AlignMolecule* is connected the specified time step is used for alignment. With this method, however, no atoms can be selected.

Align Options



Choose between these two options for aligning the molecules. Note, that the second option is disabled if the connection port *AlignMolecule* is empty.

Selection



Get selection from *AlignMolecule*. For a description on how atoms can be selected refer to *Molecule View*.

Action



Start computation of mean molecule.

4.7 Measurement

This module lets you determine distances, angles, and dihedral angles between atoms of the molecule.

To obtain a measurement you need to select the atoms that are to be examined. In order to select objects in the viewer you need to switch into the interaction mode of the viewer (for example by typing the ESC key as a toggle) and **Ctrl**-clicking on the desired atoms. You can read more about selecting structures in the documentation of the highlighting port of the *Molecule View*.

Connections

Module [required]

A module of type *Molecule View* which is used for the selection of atoms.

Ports

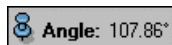
The following ports will only be visible if you have selected two atoms for distance measurement, three atoms for angle measurement, or four atoms for measurement of dihedral angles.

Selected atoms



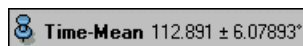
This port shows the names of the atoms that form the measurement.

Angle



This port shows the value of the measurement.

Time-Mean



This port is visible if your base molecule is derived from a trajectory. It will show the mean value of

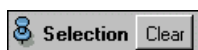
the measurement by determining the arithmetic mean over all molecules that are part of the trajectory. Besides, the standard deviation of the time-mean will be shown.

Time Plot



Like the time-mean port this port is only visible for trajectories. By pushing the show button you can see the plot of the measurement against the time steps of the trajectory. The vertical line of the plot indicates the position of the currently shown time step. The horizontal lines show the mean and the standard deviation.

Selection



With the clear button you can deselect all atoms.

4.8 Molecular Interface

This computational module provides a means to generate intramolecular and intermolecular interfaces. This is done by creating an object of type *Label Field*, which is a three-dimensional grid holding at each node a label, which represents an area of the molecule, plus additional information. The usual way to visualize a label field in order to see molecular interfaces is to use the Generalized Marching Cubes (*GMC*) to compute a surface representing the interfaces. This surface can then be viewed with the *Surface View*.

Connections

PrimaryMolecule [required]

Molecule for which intramolecular interfaces should be computed. In case of the computation of intermolecular interfaces, this is one of the molecules for which the interfaces should be generated.

SecondaryMolecule [optional]

Second molecule for the computation of intermolecular interfaces.

Ports

Resolution



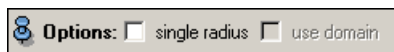
Resolution of the label field.

Level



This option menu allows you to choose the level of molecular structure for which intramolecular interfaces should be extracted, e.g. you might want to compute atomic interfaces on a very low level, or you might just want to generate interfaces between functional groups on a higher level or even secondary structures. This port disappears if the *SecondaryMolecule* connection is used.

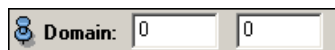
Options



The *Options* toggle list enables you to select two options. By selecting *single radius* all atoms will be considered as having the same radius, which will be equal to the *Scale Atoms* value (see below). This option will probably be rarely used. The second option, *use domain*, allows you to specify a range of

atoms, or residues, depending on the specified level. All atoms belonging to this range will be assigned to one subunit, the rest to a second subunit. Currently only two regions can thus be defined.

Domain

A control panel for the Domain parameter. It features a blue icon on the left, followed by the text 'Domain:'. To the right of the text are two adjacent input fields, both containing the number '0'.

Visible only if *use domain* is selected in *Options* list. In conjunction with the chosen level, this range specifies which atom will be assigned to *Subunit1*. All other atoms are assigned to *Subunit2*.

Scale Atoms

A control panel for the Scale Atoms parameter. It features a blue icon on the left, followed by the text 'Scale Atoms:'. To the right is a slider control with a triangle marker, and a numeric input field containing the value '1'.

Multiply all atom radii by this value.

Enlarge Atoms

A control panel for the Enlarge Atoms parameter. It features a blue icon on the left, followed by the text 'Enlarge Atoms:'. To the right is a slider control with a triangle marker, and a numeric input field containing the value '0'.

Enlarge all atom radii by this value.

Action

A control panel for the Action parameter. It features a blue icon on the left, followed by the text 'Action:'. To the right is a button with a red square icon and the text 'Generate Grid'.

Start computation of label field.

4.9 MolecularSurface

With this module you are able to compute the *van der Waals surface* and the *solvent accessible surface* of a molecule. You can specify the radius of the solvent sphere for the latter surface, and also the approximate edge length of the surface triangles.

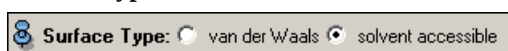
Connections

Data [required]

This port needs to be connected to the molecule for which the surface should be generated.

Ports

Surface Type

A control panel for the Surface Type parameter. It features a blue icon on the left, followed by the text 'Surface Type:'. To the right are two radio buttons. The first is labeled 'van der Waals' and is unselected. The second is labeled 'solvent accessible' and is selected.

Choose between van der Waals and solvent accessible surface.

Edge Length

A control panel for the Edge Length parameter. It features a blue icon on the left, followed by the text 'Edge Length:'. To the right is a slider control with a triangle marker, and a numeric input field containing the value '0.5'.

Specify the surface resolution by giving the approximate edge length.

Solvent Radius

A control panel for the Solvent Radius parameter. It features a blue icon on the left, followed by the text 'Solvent Radius:'. To the right is a slider control with a triangle marker, and a numeric input field containing the value '1.5'.

Choose a radius for the solvent molecule. This port is only visible, if you want to compute the solvent accessible surface.

Action

A control panel for the Action parameter. It features a blue icon on the left, followed by the text 'Action:'. To the right is a button with a red square icon and the text 'Do It'.

Start computation of surface. As result of the computation, an object of type *Surface* will be generated.

4.10 MoleculeView

The *Molecule View* allows you to display molecules in three different representations: atom spheres, wire frame, and ball-and-stick. For each of these representations there exist two *quality* modes, a fast and a correct mode.

Atoms can be colored according to the *groupings* defined on the molecule, like atoms, residues, and chains. Moreover, the displayed part of the molecule can be restricted by using the *selection browser* as well as by interaction in the viewer.

In addition to just visualizing molecules this module provides tools that facilitate interactive working with the molecule. For example you can select atoms to find out which index they have and which group they belong to. The ability to select atoms is also a precondition for the alignment procedure described in the section *Molecule*.

In order to measure distances and angles in the molecule, right-click the MoleculeView in the *object pool*, which will open a popup menu with the entry *Measurement*.

Connections

Data [required]

The molecule to be visualized.

ReferenceMolecule [optional]

A reference molecule must be connected if you want to study the alteration of the torsion angles from one time step of a molecular trajectory to another (see the *Options* port below for more information).

Continuous CM [optional]

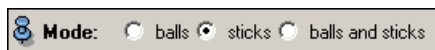
The colormap that is used to map values of float attributes to colors.

Discrete CM [optional]

The colormap that is used to map values of discrete attributes, like integers and strings, to colors.

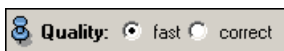
Ports

Mode



Select one of three main display modes: balls, sticks, or balls and sticks. The default is sticks.

Quality



The appearance of the balls and sticks can be altered by selection of one of two different qualities, namely a *fast* and a *correct* mode.

The fast mode shows sticks as lines and spheres by rendering a square with the image of the sphere. This permits to display even large numbers of spheres at interactive speed. However, showing the image of a sphere means the sterical expansion of the sphere in only two directions is taken into account. As long as you don't have intersecting spheres you will not see a difference to a perfectly rendered three dimensional model of a sphere.

In the correct mode spheres are represented by textured triangular meshes and sticks by cylinders.

Options



Single radius: display all atoms with the same radius.

Bond type (only in fast quality mode): display correct bond type, e.g. single, double, triple, or aromatic.

Torsion angles: display torsion angles by the use of a coil around the stick.

Atom radius

A control interface for atom radius. It features a blue icon on the left, followed by the text "Atom radius:". To the right is a slider bar with a triangle marker, and a text box on the far right containing the value "0.1".

If the option *single radius* is selected, the value given here will be interpreted as sphere radius for all atom spheres. Otherwise the value scales the van der Waals radius of the atom.

Bond radius

A control interface for bond radius. It features a blue icon on the left, followed by the text "Bond radius:". To the right is a slider bar with a triangle marker, and a text box on the far right containing the value "0.05".

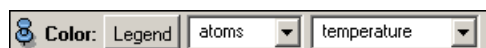
Radius of cylinders representing bonds (only visible in *correct* display mode).

Twist factor

A control interface for twist factor. It features a blue icon on the left, followed by the text "Twist factor:". To the right is a slider bar with a triangle marker, and a text box on the far right containing the value "1".

Allows to change the twisting degree of the coil.

Color

A control interface for coloring. It features a blue icon on the left, followed by the text "Color:". To the right are two dropdown menus. The first menu is labeled "Legend" and has "atoms" selected. The second menu is labeled "temperature" and has a downward arrow.

A molecule may be colored according to different schemes. For example, each atom of a specific type may get the same color. Another possibility is to give all atoms belonging to a certain group the same color. The first menu specifies the group level, such as atoms, functional groups, residues, and secondary structures. The second menu allows the user to decide which attribute of the specified level should be considered to color its groups.

If you press the button *Legend*, a separate window will be opened, which displays a legend of the coloring, i.e. a table associating colors with attribute values according to the current coloring. Clicking on a text to the right of a color will select all atoms with this color.

Continuous CM

A control interface for continuous colormap. It features a blue icon on the left, followed by the text "Continuous CM". To the right is a slider bar with a color gradient from blue to red. The left end of the slider is labeled "0" and the right end is labeled "45.66".

The colormap that is used to map values of float attributes to colors. To get the optimal mapping you need to set the range correctly. The range of values corresponding to the chosen coloring will be printed in the console window whenever the color scheme changes.

The default colormap has a constant color over the whole range. By leaving it constant you give all atoms the same color, which may be important when comparing different molecules.

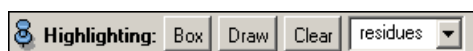
For more information see the section about *colormaps*.

Discrete CM

A control interface for discrete colormap. It features a blue icon on the left, followed by the text "Discrete CM". To the right is a slider bar with a color gradient from blue to red. The left end of the slider is labeled "0" and the right end is labeled "60".

The colormap that is used to map values of discrete attributes, like integers and strings, to colors. Again, you need to set the range correctly.

Highlighting

A control interface for highlighting. It features a blue icon on the left, followed by the text "Highlighting:". To the right are three buttons: "Box", "Draw", and "Clear". Further right is a dropdown menu with "residues" selected.

This port provides functionality for highlighting atoms in the viewer window.

- When pressing the *Box* button a box with the size of the bounding box of the molecule will appear in the viewer. All atoms contained in the box will be highlighted. You can change the size of the box in *interaction mode* by clicking on one of the highlighted handles of the box. Keep the mouse button pressed and pull in the desired direction. Notice, that the box will not exceed the molecule's bounding box. You can also move the box within the bounding box by clicking on one of the boxes' sides. Pressing the box button a second time hides it.
- After pressing the *Draw* button, you can draw a line in the viewer window which selects all atoms that are inside the line.

- The *Clear* button unhighlights all atoms and hides the box, if it is visible.
- Apart from the previously described methods for the selection of atoms you may also select parts of the molecule by picking in the viewer window. In order to do so you need to be in interaction mode. Also, the atoms need to be displayed. If you now pick an atom, it will be highlighted. If you pick another atom the previously highlighted atoms will be unhighlighted and the new atom is highlighted. Ctrl-clicking an atom that is already highlighted unhighlights it. Otherwise it will be added to the selection.
If you Shift-click an atom, the whole group of the level specified in the menu will be highlighted. The Shift key can also be used in conjunction with the Ctrl key.

Commands

`complexity setValue <value>`

Using this command you can set the complexity of the scene displayed by this viewing module. The complexity controls the number of triangles used for visualizing atoms and bonds. In fast mode it also regulates the size of the texture used for displaying the atoms. The value needs to be in the range [0.0-1.0].

`getOptimalColormapRange`

When displaying all atoms the range printed by this command is a good choice as range for the colormap.

`linewidth setValue <value>`

Sets the width (integer value ≥ 0) of the lines in wire frame mode.

`setTextureAntiAliasing <value>`

This command enables (1) or disables (0) antialiasing for the textures used in fast sphere mode. Antialiasing has no impact on the rendering speed. However, if antialiasing is enabled and you have many different colors, the generation of the textures might be considerably slower.

4.11 Observables

This module allows to display some statistics about a molecular trajectory. All properties stored in the trajectory may be displayed, such as *potential energy*, *kinetic energy*, *total energy* etc.

Connections

Data [required]

The molecular trajectory for which properties should be displayed.

Time [optional]

A time object that may be connected to the module in order to synchronize, e.g., a viewing module with the line in the plot window, indicating the position of the time step within the trajectory.

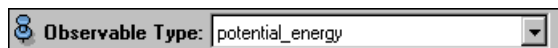
Ports

EditTime



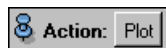
This port is most useful in conjunction with a viewing module in order to see the property of the displayed time step. In order to create a time object click the time slider with the right mouse button and select *Create time* from the menu. As a result a time object will appear in the object pool, which should be connected with an object of type *Molecule* which can be visualized with, e.g., the *Molecule View*.

Observable Type

A control box with a blue icon on the left, followed by the text "Observable Type:" and a dropdown menu showing "potential_energy".

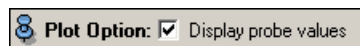
Choose the type of property which should be plotted.

Action

A control box with a blue icon on the left, followed by the text "Action:" and a button labeled "Plot".

Plot the *Observable Type* within a new window.

Plot Option

A control box with a blue icon on the left, followed by the text "Plot Option:" and a checked checkbox labeled "Display probe values".

Select this toggle if the values of the plotted properties at the current time are shown in the plot window.

4.12 SecStructureView

This module provides the functionality for viewing the secondary structure of a molecule. It relies on the information given in the data entry of the connected molecule. You can only view helices, sheets and turns if information about these structures has been read in (for example from a *pdb* file).

The backbone will be interpolated by using the coordinates of those atoms whose comment is set to "N" and "C" (if the structure is a polypeptide) or respectively "O5*" and "O3*" (if the structure is a DNA or RNA) If the atom-comment attribute does not exist (for example molecules that have been imported from unichem files) no backbone can be viewed.

Connections

Data [required]

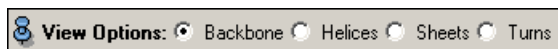
The molecule that is to be viewed.

Backbone Colormap [optional]

This port lets you connect the colormap of the backbone to a user defined colormap. A similar port exists for helices, sheets, and turns.

Ports

View Options

A control box with a blue icon on the left, followed by the text "View Options:" and four radio buttons labeled "Backbone", "Helices", "Sheets", and "Turns".

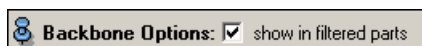
To switch between the options of the different structure types you can use these radio-buttons. The option ports for the desired structure will then be visible.

Backbone Radius

A control box with a blue icon on the left, followed by the text "Backbone Radius:" and a slider control with a value of 0.2.

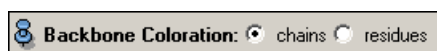
This port lets you change the backbone radius.

Backbone Options

A control box with a blue icon on the left, followed by the text "Backbone Options:" and a checked checkbox labeled "show in filtered parts".

With the toggle *show in filtered parts* you can choose whether the backbone shall be viewed in parts of the molecule that belong to any structures that have been hidden using the filters.

Backbone Coloration

A control box with a blue icon on the left, followed by the text "Backbone Coloration:" and two radio buttons labeled "chains" and "residues".

The setting of this port determines in which way the backbone will be colorized. You can colorize the *chains* or the *residues* in the colors defined by the colormap port.

Backbone Colormap



By right-clicking on this port you can choose between different colormaps.

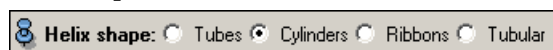
If a constant color is selected a left-click on the port will open a color editor. If a colormap is selected then a left-click will create a colormap object in the object-pool. A similar port is available for helices, sheets and turns.

Backbone Color Legend



The color legend helps you to associate the colors to the names or types of the colored structures. A similar port exists for helices, sheets and turns.

Helix shape



This port lets you choose between different methods for viewing helices.

The *tube* view will show the helix by drawing a tube connecting the interpolated backbone coordinates.

The *cylinder* view will approximate axis using a least square error minimization for the sum of the distances between the backbone coordinates of the helix and the axis. These axis will be used to center the cylinders for the helices.

The *ribbon* view is similar to the tube representation except that the backbone is shown using a surface whose normals are set perpendicular to a line fitting the helix.

The *tubular* view will show a tube fitting the helix.

Helix Radius



With this port you can adjust the radius of the shown tubes, cylinders or tubulars.

Helix Ribbon Width



This port can be used to adjust the width of ribbon.

Helix Ribbon Options



The option *3D* enables a three dimensional representation of the ribbon surface.

Helix Colormap



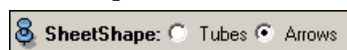
The use of this port is similar to the *backbone colormap*.

Helix Color Legend



The use of this port is similar to the *backbone legend*.

SheetShape



This port lets you choose between different methods for viewing sheets.

The *tube* view will show the sheet using a tube connecting the interpolated backbone coordinates.

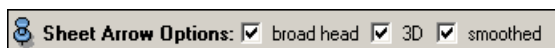
The *arrow* view will show arrows instead which show the orientation of the planes that are formed by the adjacent strands and the hydrogen bridges between them.

Sheet Arrow Width

A control panel for 'Sheet Arrow Width'. It features a small blue icon with a plus sign, followed by the text 'Sheet Arrow Width:'. To the right is a horizontal slider with a triangle marker, and a numeric input field containing the value '1.5'.

This port is visible if you view the sheets as arrows. It determines the width of the arrows.

Sheet Arrow Options

A control panel for 'Sheet Arrow Options'. It features a small blue icon with a plus sign, followed by the text 'Sheet Arrow Options:'. To the right are three checkboxes: 'broad head' (checked), '3D' (checked), and 'smoothed' (checked).

The options *broad head* and *3D* configure the style of the arrows further.

The option *smooth* will bezier interpolate the backbone coordinates of the sheet.

Sheet Radius

A control panel for 'Sheet Radius'. It features a small blue icon with a plus sign, followed by the text 'Sheet Radius:'. To the right is a horizontal slider with a triangle marker, and a numeric input field containing the value '0.3'.

If you view the sheets as tubes this port lets you configure the radius of the tubes.

Sheet Colormap

A control panel for 'Sheet Colormap'. It features a small blue icon with a plus sign, followed by the text 'Sheet Colormap'. To the right is a color gradient bar transitioning from blue to red, with numeric input fields at each end containing the values '0' and '1'.

The use of this port is similar to the *backbone colormap*.

Sheet Color Legend

A control panel for 'Sheet Color Legend'. It features a small blue icon with a plus sign, followed by the text 'Sheet Color Legend:'. To the right is a button labeled 'Show'.

The use of this port is similar to the *backbone legend*.

Turn radius

A control panel for 'Turn radius'. It features a small blue icon with a plus sign, followed by the text 'Turn radius:'. To the right is a horizontal slider with a triangle marker, and a numeric input field containing the value '0.3'.

This port lets you determine the radius of the turn tubes.

Turn Colormap

A control panel for 'Turn Colormap'. It features a small blue icon with a plus sign, followed by the text 'Turn Colormap'. To the right is a color gradient bar transitioning from dark red to yellow, with numeric input fields at each end containing the values '0' and '255'.

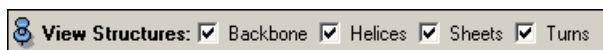
The use of this port is similar to the *backbone colormap*.

Turn Color Legend

A control panel for 'Turn Color Legend'. It features a small blue icon with a plus sign, followed by the text 'Sheet Color Legend:'. To the right is a button labeled 'Show'.

The use of this port is similar to the *backbone legend*.

View Structures

A control panel for 'View Structures'. It features a small blue icon with a plus sign, followed by the text 'View Structures:'. To the right are four checkboxes: 'Backbone' (checked), 'Helices' (checked), 'Sheets' (checked), and 'Turns' (checked).

This port can be used to filter out the backbone or all sheets helices and turns.

Chapter 5

Alphabetic Index of Components

5.1 Molecule Selection Browser

The selection browser is a handy tool for the easy investigation of the two-dimensional structure of a molecule. It consists of three main parts: the menu bar, a scroll window, and a highlighting section. In the scroll window all items of a chosen level will be displayed. Most of the popup menus in the menu bar allow the user to control what is being shown in the scroll window. Finally, the highlighting section provides further functionality for the selection of items in the scroll window.

The selection browser enables the user to quickly select parts of the molecule, which will then be highlighted in the browser as well as in the viewer by the viewing modules connected to the molecule. The highlighting allows to establish a correlation between the two- and the three-dimensional molecule's structure. Highlighted atoms can be used for masking. Thus, the user can concentrate on the regions of interest. Furthermore, the masking allows to combine different representations in an easy way.

The scroll window

Displayed information

In this window all items of some chosen level are displayed hierarchically in the first column. An item might be a single atom or a group of items.

If there exists additional information for an item, this will be displayed in the following columns. If the information is not available, the entry will be empty. Additional information typically includes the item's type, and the membership of the item in some level. Note, that only the direct membership will be considered, i.e. if an item is not a member of some level, but a member of some group, which is an item of the level, the membership information will not be displayed. An exception is being made, if 'atoms' is chosen as master level. In this case, we display all groups of the information level the atom somehow belongs to.

If there are any viewing modules connected to the molecule, there will be more columns (let's call them viewer columns) to the right of the info columns - one per module. These columns are used to display whether an item is shown by the viewing modules. For each item a rectangle will be drawn in each viewer column. If the item is fully shown by the viewing module, the rectangle contains a tick. If it is partially shown, i.e. some but not all of its group items are shown, the tick will be greyish. There is one other symbol, which might get displayed in the rectangle, a 'B'. The 'B' stands for backbone and indicates that the atom belongs to those atoms currently defined as backbone. This will be described in the following subsections. The 'B' might be black or grey. Again, grey means the atom is not displayed by the viewing module. A black letter indicates that the atom is shown, but simply because it belongs to the backbone.

Item and information columns

If an item is a group, i.e. it contains items, a little icon will be placed to the left of the item name. You can click on the icon to open/close the item, i.e. to show or hide the contained items, respectively.

In order to select an item in the scroll window the user can click on the respective item, which will be highlighted immediately. To toggle the selection of an item you must CTRL-click the item. When SHIFT-clicking an item a whole range will be selected. You can also pull the mouse with the left mouse button pressed over a couple of items which selects all of them. CTRL-click and pull again toggles the selection. If an item is fully selected, it will be highlighted in red. If at least one but not all of its subitems are selected, we call the item partially selected and highlight it in green.

There is also the possibility to select items by its type or some other information in one of the information columns. If you click on the displayed information for some item, all items with the same information will also be selected.

Viewer columns

If there are any viewing modules connected to the molecule, such as *MoleculeView*, *BondAngleView*, or *SecStructureView*, there will be viewer columns in addition to the master and information columns - one per viewing module. Each of these columns contains a tick box for each item displayed. The tick box contains information about whether an item is shown by a viewing module or not. Before the different states of a tick box will be explained we make a short excursion. For convenience, all atoms are classified as belonging to the backbone or not, whereby we use predefined backbone classes. You may restrict your selection to only backbone atoms, which means that an atom will only be displayed by the viewing module if it belongs to the backbone. Apart from that you may also want to show at least all backbone atoms plus additional atoms. Therefore, internally there exist two arrays, one containing all backbone atoms and one containing so-called 'selected' atoms. Depending on the backbone mode there are three choices:

- all atoms contained in either array will be displayed, or
- all atoms contained in both arrays will be displayed, or
- only those atoms contained in the 'selected'-array will be displayed.

Using those two arrays allows the user to quickly change between backbone atoms and all others without loosing the previous atom masking.

Now let's go back to the tick box' states. There are five different states of a tick box, whereby there is one state with two different meanings.

- The box is empty - the complete item is not shown by the viewer.
- The box contains a black tick - the complete item is shown.
- The box contains a grey tick and the item is an atom - the atom is a member of the 'selected'-array, but is still not shown due to the backbone mode, i.e. it does not belong to the backbone.
- The box contains a grey tick and the item is not an atom - the item is only partially shown in the viewer.
- The box contains a black 'B' (only for atoms) - the atom is shown simply because it belongs to the backbone.
- The box contains a grey 'B' (only for atoms) - the atom is not shown, but belongs to the backbone.

Clicking on a box toggles its state. You may also click on one item and then pull the mouse pointer over a number of items. All of these items will be set to the same state as the first item.

Apart from directly clicking on a box there is yet another possibility to change the state of the tick boxes. If you click on the header of a viewer column with the right mouse button a popup menu will appear containing the following entries:

- Add - adds the currently highlighted atoms to the 'selected'-array.
- Remove - removes the highlighted atoms from the selection.
- Replace - replaces selection by highlighted atoms.
- Clear - clears the 'selected'-array.
- All - selects all atoms.
- Restrict to - this popup menu allows the user to restrict the set of displayed atoms to some predefined set.
- Combine with - if one of the entries in this popup menu is selected, the chosen set will be the minimum set that is being displayed by the respective viewing module.
- Set - the 'selected'-array is set to the chosen predefined set of atoms. Starting from this selection you can add or remove parts of the molecule.

Highlighting section

This browser section consists of a text field and a button row. In the text field you can type atom expressions, which will be explained next.

Atom Expressions

Syntax

```
atomExpr →      ( atomExpr )
                | NOT atomExpr
                | atomExpr AND atomExpr
                | atomExpr OR atomExpr
                | WITHIN (atomExpr,float)
                | BONDED (atomExpr[,int])
                | CS
                | atomSpecifier
atomSpecifier → hierName[/attrName]=ID
```

hierName stands for a name of a hierarchy level. The following abbreviations can be used for the most common levels:

a=atoms, r=residues, b=bonds, s=secondary structure

attrName is optional and defines the name of an attribute (e.g. temperature, occupancy, type ...) of the given level. If it is omitted the ID is supposed to specify the internal name used by Amira as shown by the list command. If an attribute name is given, the ID is supposed to stand for values of the attribute.

To see which hierarchy levels and respective attributes are defined for the given molecule take a look at the *Color-Scheme-Port* of the Molecule View. The right pull down menu will show all available attributes for the level chosen in the left pull down menu.

ID specifies an identifier of a member of the given hierarchy and attribute. It can be the name/index of the group (eg. L112 for the residue 112 on the L chain) or the type (eg. ASP for all asp-residues). Wildcards such as * and ? may be used or ranges (e.g. L112-L115) may be given.

Several *atomSpecifier* combinations can be used in one expression by linking them logically via the operators AND, OR and NOT (short &, | and !). Priorities can be specified using usual brackets ((and)).

The WITHIN(selExpr,float) operator selects all atoms which are nearer than *float* Angstrom to any of the atoms specified by *atomExpr*.

With the BONDED(selExpr[,int]) operator all atoms which are recursively connected to any atoms specified by *selExpr* will be chosen. You can optionally give an integer value defining the maximal bond-steps. If it is omitted there will be no limit.

CS lets you specify all currently selected (highlighted) atoms.

Shortcuts

Amira also offers shortcuts that have been assembled by using the previously mentioned syntactical elements. The shortcuts can be found in your local Amira directory in `share/molecules/atomExpr.cfg` and can be edited and supplemented.

The standard aliases included in the current Amira release are:

acidic	acidic amino acids
acyclic	acyclic amino acids
aliphatic	aliphatic amino acids
alkali	atoms which are alkali metals
alkaliesarth	atoms which are earth alkali metals
amino	amino acids
aromatic	aromatic amino acids
at	adenine or thymine
backbone	atoms of protein or DNA/RNA
basic	basic amino acids
buried	amino acids which are usually found inside the protein
cg	cytosine or guanine
charged	charged amino acids
cyclic	cyclic amino acids
h2o	water molecules
helix	helices
halfmetallic	atoms which have halfmetallic properties
halogens	halogenic atoms
hetero	heterogenic atoms
hydrophobic	hydrophobic amino acids
ions	charged heterogenic atoms
metallic	atoms which have metallic properties
neutral	neutral amino acids
noblegas	atoms which have noblegas properties
nonmetallic	atoms which have nonmetallic properties
nucleic	nucleic acids
nucleicbackbone	backbone atoms of RNA/DNA
polar	polar amino acids
proteinbackbone	backbone of a polypeptide
purine	adenine or guanine
pyrimidine	cytosine or thymine
sheet	sheets
sidechain	atoms of the protein/RNA/DNA which do not belong to the backbone
site	
surface	amino acids which tend to be found on the surface of molecules
turn	

Examples

- `atoms=5-8`
all atoms whose index is between 5 to 8.
- `s/type=helix AND NOT (a=C OR a=N)`
all atoms which belong to helices except C and N atoms.
- `r/type=A*`
all atoms which belong to residues whose type name begins with the letter A.
- `BONDED(a=4 OR a=100, 6)`
all atoms which are connected over 6 or less steps to the two specified atoms
- `WITHIN(r=A11, 3.1) AND C`

all carbon atoms which are not away further than 3.1 Angstrom from atoms of the residue 11 on chain A

- **acidic AND helix**

all carbon atoms of acidic amino acids which belong to helices

Buttons

The buttons work in conjunction with the expression in the text field. When pressing any of the first three buttons, the atom expression will be evaluated, and the atoms corresponding to the expression will be *added to*, *removed from*, or *replace* the highlighting, respectively. The last two buttons clear the highlighting or highlight all items, respectively.

Menu bar

The menu bar contains a couple of popup menus which will be explained below.

MasterLevel

The *MasterLevel* menu contains all levels currently defined in the molecule's *topology*. All groups of the current master level will be displayed in the scroll window.

Options

The *Options* menu allows the user to quickly open and collapse items, i.e. to show and hide the items contained in other items.

Info

With the help of the *Info* menu you can display additional information columns, such as type, or the item's membership in some group of some level.

Viewer

If you have many viewing modules connected to the molecule, the number of columns might confuse you. If you are only working with one or a few of the viewing modules, you might want to hide the other columns. This can be done with the help of this menu.

Highlighting

By default, all viewing modules that allow highlighting will highlight all selected atoms. The *Highlighting* menu enables the user to turn highlighting for each viewing module on and off.

Chapter 6

Alphabetic Index of Data Types

6.1 MolTrajectory

An object of type *MolTrajectory* represents a trajectory, i.e. a series of molecular configurations. Either the trajectory was loaded directly from a file. Alternatively it can be attached to an object of type *MolTrajectoryBundle* singling out one of the trajectories contained in that bundle.

Connections

Bundle [optional]

The trajectory may be connected to a *trajectory bundle*. If so, the *Trajectory* port will appear in the *working area* which enables the user to control which member of the bundle this trajectory represents.

Ports

Trajectory



The *Trajectory* port is visible only if the trajectory object is connected to a trajectory bundle. The menu offers you the choice between the trajectories contained in the bundle.

6.2 Molecular Field

An object of this data class extracts molecular properties, represented as uniform scalar fields, from an object of type *Molecule*. These scalar fields can either be visualized directly with several viewing modules typical for visualizing scalar fields such as *OrthoSlice* or *Voltex*. Furthermore, the scalar fields can be displayed on a surface in pseudo-color mode, e.g. on a *molecular surface*. Finally, you can compute the *gradient* of the scalar field and visualize the resulting vector field with the *DisplayISL* module.

Connections

Molecule [optional]

Connection to molecule from which fields are extracted.

Ports

Field Type



Field, that should be extracted from molecule, e.g. electrostatic potential.

6.3 Molecule

An object of data type *Molecule* contains information about the structure of a molecule. Typical information that are stored are the types and positions of the molecule's atoms and the bonds between the atoms plus the type of each bond. Furthermore, the data object stores information about groups of atoms in a hierarchical way. For example a functional group consists of a number of atoms, several functional groups may form a residue, and a couple of residues may form a secondary structure. This allows to quickly traverse the structure of a molecule.

In addition to structural information, an object of type *Molecule* may contain property fields, such as the electrostatic potential or hydrophobicity, which can be extracted from the molecule by clicking the molecule icon in the *object pool* and selecting *MolecularField*, which results in an object of type *MolecularField* appearing in the object pool.

Molecules can be visualized with one of the following viewers: *MoleculeView*, *BondAngleView*, or *Sec-StructureView*. In addition to that there are two modules by which some kind of molecular surface can be generated. The *MolecularInterface* module can be used to generate intramolecular interfaces, such as between single atoms or residues. Beyond that this module also permits the generation of the *van-der-Waals surface* of a molecule. Also, the interface between two molecules can be computed. The *MolecularSurface* module provides the functionality of computing the *Solvent Excluded Surface*.

When comparing the structures of several molecules with each other one is faced with the problem of aligning molecules to each other. This can be easily done by connecting the *AlignMaster* connection port to a second molecule which will serve as reference. There are several alignment modes. If the molecules have the same number of atoms you can align the two molecules to each other by using all atoms, whereby the *i*'th atom of the first molecule corresponds to the *i*'th atom of the second molecule. You may also select atoms in either the *Slave* or the *Master* molecule. The *Slave* is the molecule to be aligned. If the molecule have different numbers of atoms, the only way to align molecules is to select atoms in both molecules.

Connections

Data [optional]

The molecule may be connected to a *molecular dynamics trajectory*. If so, a port *Time* will appear in the *working area* which enables the user to load new time steps into the molecule.

AlignMaster [optional]

In order to align a molecule to another one, this port must be connected to a second molecule.

Time [optional]

Sometimes it may be required to synchronize two or more time dependent data objects, such as a molecule from a trajectory and the corresponding electrostatic field. In this case the time port can be connected to a time object.

Ports

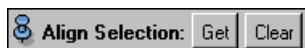
Alignment



This port consists of two menus. The first one allows you to specify how the molecule should be aligned.

Possible options are *original*, *center of gravity* and *to alignmaster*. The first option means that the molecule will not be aligned at all, but remains at its original position. If the second option is chosen, the center of the molecules atoms is computed and the molecule is translated such that this center is at the origin of the coordinate system. The third option, finally, allows the user to really align two molecules to each other taking the molecules structure into account. However, this option is only enabled if the connection port *AlignMaster* is connected to a second molecule. If *to alignmaster* is selected, the second menu becomes important, which specifies how the alignment is carried out. The easiest way is to *use all atoms*. This, however, is only allowed if the two molecules have the same number of atoms. The last option, *select atoms in 2 molecules*, is useful if you have two molecules that do not agree in the atom sequence. Thus, you need to specify the corresponding atoms of the two molecules. You do this by the sequence of selection. If you have two different structures of the same molecule, however, you might not want to select the same atoms in both molecules. As convenience options for this serve the second option, *select atoms in slave molecule*, and the last one, *select atoms in master molecule*. The selection is done via the *Molecule View*.

Align Selection



This port appears if you want to align the molecule *to alignmaster* and the specified option in the second menu is not *use all atoms*. The selections that you do in the *Molecule View* are not automatically handed over to the alignment procedure, but must be explicitly loaded by clicking the *Get* button. As result the current selection will be appended to the previous selections. If you want to make sure, that only the current selections are used, click the *Clear* button before you get the selection.

Action



Press this button if you wish to carry out the alignment.

Time



The *Time* port is visible only if the molecule object is time dependent, i.e. it is connected to a molecular dynamics trajectory. In the slider you can select a specific time step. If you wish to step through the trajectory use the buttons next to the slider. For continuous animation use the outer buttons.

Selection Browser



Press this button if you wish to open the selection browser for this molecule.

Commands

The *console window* offers some additional commands for molecules.

The easiest way to use the console for a molecule is to click on the molecule object in the work area and then press the **TAB**-key in console window to prompt for the name of the selected object. Behind the name you can use the following selection commands:

```
list
```

Lists all levels defined for the molecule.

```
list <levelName>
```

Lists all groups of level *levelName*.

```
list <levelName/[groupName|groupRange]>
```

Prints information about a single group or a number of groups specified by a group range.

define <levelName/groupName> <groups>

This command defines a new group *groupName* in the level *levelName*. If *levelName* does not yet exist, it will be added as a new level. *groups* is a list of groups of possibly different levels. Groups can be specified using ranges.

Example: The following command defines a group *one* in level *test* consisting of atoms 1 and 3 and residues 4 to 6.

- `define test/one atoms/1 atoms/3 residues/4-6`

defregexp <levelName/groupName> <atomExpr>

Define a new group by using *atom expressions*. The new group will only contain atoms, i.e. no higher level groups.

sel <atomExpr>

Lets you add atoms specified by *atomExpr* to the current selection.

Example: The following command will select all carbon-atoms which are located in helices.

- `sel a=C AND s/type=helix`

desel <atomExpr>

Lets you remove atoms from the selection.

applyTransform

The current transform is applied to the molecule, i.e. the original atomic positions are replaced by the transformed one's.

Chapter 7

Alphabetic Index of File Formats

7.1 MDL

The file format *sdf* is part of the set of MDL file formats to save chemical structures.

Amira uses and writes the information of the atom and bond blocks.

7.2 PDB

PDB is a file format for storing structures of biomolecules. The format is used for archiving molecules in the online protein data base. [<http://www.rcsb.org/pdb/>]

Supported records

Amira uses parts of the information that is given in a pdb file. All following record types will be parsed:

HELIX SHEET TURN SITE MODEL ENDMDL ATOM HETATM TER CONECT END

The rest will be skipped.

When saving a structure the following record types will be written:

HEADER TITLE AUTHOR

SEQRES HELIX SHEET TURN SITE SSBOND

ATOM HETATM TER CONECT MASTER END

Amira supports pdb files that follow the conventions of the format version 2.0 or higher.

How to identify pdb groups in Amira

In Amira all levels (atoms, residues, bonds , ...) contain a set of information fields or attributes. Every level contains the fields 'name' and 'index' with the latter one being an internal identifier that does not necessarily correspond to the succession of the groups in the file.

If the molecule has been read from a pdb file then the **atom** name corresponds to the serial number (columns 7 - 11) of the respective ATOM records.

The 'comment' field contains the information of the field named 'atom name' (columns 13 - 16) in the PDB file.

The 'name' field of **chains** contains the chainID (column 22) of the respective ATOM record. HETATM records whose chainID field is empty will not belong to any chain in Amira. The chain name of atoms belonging to ATOM records whose chainID field is empty will be set to 'NIL'.

The 'name' fields of **residues** are composed of the chain name and the residue sequence number (columns 23 - 26). If the residue is a het-group the name will start with 'HET'.

The field 'type' contains the residue names (columns 18 - 20). If none is given the field will be set to 'UNK'

(for unknown).

Example:

```
ATOM      441  CB  VAL L  58      21.025  37.362  -2.515   1.00   8.15   1IGM 562
```

The atom will be known with name '441' and comment 'CB'. It will belong to the residue with name 'L58' and of type 'VAL' which is sited on the chain with name 'L'.

Possible problems

Amira can read several models of one molecule if the different versions are separately declared between MODEL and ENDMDL records. If there are several possible conformations in one model (by using alternate location identifiers) Amira will read only the first version (that one with the altLocId 'A').

Columns 77-78 should contain the element name of the atom. In some PDB files this field is empty or contains other information. In this case Amira uses the first two letters of the atom name to determine the atom type. However in special cases this is not unequivocal. It is not allowed that the field contains some other information that is not numerical.

All information in column 79-80 is interpreted as the charge of the atom. Thus linenumbers in this field will lead to strange charge values.

All bonds of amino acids and nucleotides will be created using an internal database. Bonds of non standard residues have to be declared in the connect records.

You can get a complete documentation of the fileformat at <http://www.rcsb.org/pdb/>.

7.3 PSF/DCD (CHARMM)

This is a format used by CHARMM. It consists of two files, one holding the structural information (suffix .psf) and one containing the trajectory, i.e. the atom coordinates (suffix .dcd).

Amira can read this format and translates the sections of the psf-file into grouping levels.

7.4 Tripos

The file format *mol2* is used to save Tripos Sybyl molecule information. See more at <http://www.tripos.com/services.old/mol2/index.html>.

Amira uses and writes the information given in the molecule atom and bond records.

7.5 UniChem

The *uni* fileformat is used by the molecular modelling software UniChem to save the structure of molecules and computational results. See more at <http://www.oxmol.com/software/unichem/>.

Amira uses all structural information which is given in the COORDINATES and BONDS blocks. Files that were exported by Amira follow the file format version 4.0 and can be used with the UniChem software.

7.6 ZIB Molecular File Format

ZIBMolFile is a structured file format developed for exchanging molecular data.

This description is divided into two parts. First we describe the general and context independent file structure. In a second step we explain our structuring conventions for molecular data.

Structured files

A structured file consists of a header and two sections: a plain text section describing the file's structure, and a binary section containing the actual data:

```
STRUCTURED FILE V0.1 BINARY_BE MOLECULE_TRAJECTORY
TYPE
struct {
    atoms : array of struct {
        id      : integer*4;
        name     : string;
        mass     : real*4;
        properties : integer*4[8];
    };
};
DATA;
..... binary data .....
```

The file structure is similar to a structured data type in C or PASCAL. There are two composite data types, (array and struct), and three elementary data types, (integer, real, and string). Finally there exists a special type (extern) for including other structured files.

struct contains named fields of arbitrary heterogeneous type.

array contains numbered fields of the same type. Arrays are one-dimensional only. Their size can either be specified in the type declaration (Example: `array[3] of integer`) or in the data section.

real and **integer** contain numerical, possibly multi-dimensional, data. The number of bytes in an element may be specified using `"*n"`. At the moment only 4 byte types are allowed. The rank (number of dimensions) must be specified in the type declaration. The number of elements in each dimension may be specified in the type declaration or in the data section (indicated by `"."`). Example: `integer*4[4, ., 3]`.

string contains ASCII data of arbitrary length. Again the length can be specified in the type declaration (Example: `string*22`) or in the data section.

extern is a special kind of a **string**. The length must be specified in the data section. The string is interpreted as a structured file whose contents should appear as part of the current files structure.

The data section is written as follows:

struct: The individual members are written in the given order.

array: The elements are simply written one after another. If the array size is unspecified, it is written in front of the array as a 4-byte integer.

string and **extern**: The data is written byte by byte. If the size is unspecified in the data type, it is prepended as a 4-byte integer.

real and **integer**: Individual elements are written in the format specified in the file header (big endian or little endian). Floating-point numbers are in IEEE format. Matrices are written in FORTRAN order. If any dimension is unspecified in the data type, it is written as a 4-byte integer before the data. If more than one dimension is unspecified, they (and only they) are written before the data in their natural order. If a dimension is specified in the data type, it is never written in the data. Example:

```
integer[2, ., 4] = [ [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ],
                    [ [13,14,15,16], [17,18,19,20], [21,22,23,24]] ]
```

is written as a sequence of 4-byte integers.

```
3 1 13 5 17 9 21 2 14 6 18 10 22 3 15 7 19 11 23 4 16 8 20 12 24
```

The leading 3 specifies the dimension unspecified in the data type. The following 24 numbers are the entries of the array.

Molecular data - a minimal file

Reading a structured file containing molecular data Amira expects the file structure to have certain elements. Moreover the file may have arbitrary additional elements, which will be ignored. The minimal structure a molecular data file must supply is:

```
struct {
    molecules : array of struct{
        name : string;
        groupings : struct {
            atoms : array of struct{
                id : integer*4;
                type : integer*4;
                name : string;
            };
        };
        trajectory : array of struct{
            coordinates : real*4[3,.];
        };
    };
    typbase : struct {
        atoms : array of struct{
            id : integer*4;
            name : string;
            number : integer*4;
            radius : real*4;
        };
    };
};
```

In the following we will use a notation inspired by C to denote elements of the structure. A **struct**'s field will be specified as the **struct** followed by a “.” and the name of the field. For array elements we will write the array followed by brackets (“[]”). For example “**molecules**[**groupings**.**atoms**].**id**” means the field “**id**” in an element of the array “**atoms**” which is a field of the **struct** “**groupings**” which is a field of an element of the array “**molecules**”.

The overall type of the file is a **struct** containing the fields **molecules** and **typbase**. **molecules** is an array of which each element describes a molecule. **molecules**[**name**] contains the name of the molecule.

molecules[**groupings**] specifies the topology of the molecule, i.e. which atoms it contains. Every element of **molecules**[**groupings**.**atoms**] describes one atom of the molecule. **atoms**[**id**] is a unique numerical identifier. **atoms**[**type**] is a numerical reference to a field in the **typbase** described below. **atoms**[**name**] is a textual identifier unique with respect to all atoms. It is used to specify this atom on the command line.

molecules[**trajectory**] is an array with every element describing one timestep of a molecule trajectory. **trajectory**[**coordinates**] has the same number of elements as **molecules**[**groupings**.**atoms**] and every element specifies a position for the corresponding atom.

typbase is structured analogously to **molecules**[**groupings**]. Corresponding to **molecules**[**groupings**.**atoms**] the array **typbase**.**atoms** holds information common to atoms of the same type. Precisely the type information of an atom is stored in the **molecules**[**groupings**.**atoms**].**type**'th element of **typbase**.**atoms**.

typbase.atoms[].id holds a numerical identifier which should be identical to the element index. **typbase.atoms[].name** is a textual label for making the typefication transparent for the user. **typbase.atoms[].number** and **typbase.atoms[].radius** are the atomic number and radius of the typed atom.

Bonds

Bonds can be specified by adding an array named **bonds** to **molecules[].groupings**:

```
bonds : array of struct{
    id : integer*4;
    type : integer*4;
    components : integer*4[2];
    index : integer*4;
};
```

bonds[].id is a unique numerical identifier which should be greater than any value specified in **atoms[].id**. Analogous to **atoms[].type** for atoms **bonds[].type** specifies a numerical type for every bond, although this is not yet interpreted by Amira. The array **bonds[].components** holds the numerical identifiers of the two atoms connected by the bond. **bonds[].index** finally indicates if it is a single (1), double (2), triple (3), or an aromatic bond (4). The value 8 indicates a bond of unknown type.

A big example

The information stored in the described file format can be divided into different degrees of generality. Information stored in **typbase** is specific to the molecular force field used in the calculations leading to a molecular trajectory. It would be redundant to store this information in every file for which the same molecular force field was used. Thus it can be stored in an extra file, which is then referenced by an **extern** declaration.

A similar mechanism applies if more than one trajectory is calculated for the same molecule. The topological information about atoms making up the molecule and bonds connecting them will be the same for all the trajectories. On the other hand we do not want a file for every single molecule topology. Thus our trajectory can reference another file containing a multitude of molecular topologies and just supply the timesteps itself.

The following is an example realising these concepts:

trajectory file:

```
struct {
    typbase : extern;
    molecules : array of struct{
        id : integer*4;
        observations : struct {
            global_obs : string[.];
        };
        text : string[.];
        name : string;
        molbase : extern;
        trajectory : array of struct{
            global_obs : real*4[.];
            coordinates : real*4[3,.];
        };
    };
};
```

molbase file:

```
struct {
  typbase : extern;
  molecules : array of struct{
    id : integer*4;
    observations : struct {
      global_obs : string[.];
    };
    text : string[.];
    name : string;
    groupings : struct {
      atoms : array of struct{
        id : integer*4;
        type : integer*4;
        name : string;
      };
      bonds : array of struct{
        id : integer*4;
        components : integer*4[2];
        type : integer*4;
        index : integer*4;
      };
      dihedrals : array of struct{
        id : integer*4;
        components : integer*4[4];
        type : integer*4;
      };
      residues : array of struct{
        id : integer*4;
        name : string;
        remark : string;
        type : integer*4;
        from_id : integer*4;
        to_id : integer*4;
      };
    };
  };
  trajectory : array of struct{
    coordinates : real*4[3,.];
  };
};
```

typebase file:

```
struct {
  atoms : array of struct{
    id : integer*4;
    type : integer*4;
    name : string;
    number : integer*4;
    periodic_row : integer*4;
    mass : real*4;
    charge : real*4;
    partial_bond_charge_increment : real*4[2];
  };
};
```

```

        radius      : real*4;
        N_i         : real*4;
        A_i         : real*4;
        g_i         : real*4;
        alpha       : real*4;
        datyp       : string;
        properties  : integer*4[8];
        equivalence : integer*4[4];
    };
};

```

Some yet undescribed features appear in this example. The first is the possibility to supply scalar observables corresponding to the single timesteps. Possible values are different energies like kinetic and potential energy. The names of the observables are read from **molecules[.observations.global_obs[]** in the trajectory file. Their values are stored in **molecules[.trajectory[.global_obs[]**

Besides **bonds** the file can contain arbitrary groups of atoms or groups of groups. These groups are organized in levels which appear here as the elements of **molecules[.groupings**.

Every level is an array whose elements describe the particular groups. Every group is a struct with the fields **id**, **type**, and fields specifying the contents of the group. **id** is numerical identifier unique over all groups specified for the molecule. Precisely the identifiers in one level should be consecutively numbered and bigger as all the identifiers used in the preceding levels.

There are two ways of specifying a group. The general way is to specify all elements explicitly. In that case the group contains a field **components[]** containing the ids of all elements of the group. If the elements of a group cover a consecutive sequence of ids, it is sufficient to specify only the first and the last id. In that case the group contains the fields **from_id** and **to_id**.

dihedrals are groups of 4 atoms connected by three bonds like a chain. Every such group defines two intersecting planes. The angle of intersection is called a *dihedral angle* or *torsion angle*. For performance reasons Amira will read *dihedrals* only if the molecule has less than 500 atoms. Torsion angles can be displayed with the *Molecule View*.

residues are groups representing the building blocks of a complex molecule, for example the amino acids of a protein. All groupings can be used to color the molecule in the *Molecule View* or in the *BondAngle View*.