JCTC Journal of Chemical Theory and Computation

Molecular Graphics of Convex Body Fluids

Adrian T. Gabriel,[†] Timm Meyer,[‡] and Guido Germano*[,†]

Department of Chemistry, WZMW, and Department of Mathematics and Computer Science, Philipps-University Marburg, 35032 Marburg, Germany

Received August 1, 2007

Abstract: Coarse-grained modeling of molecular fluids is often based on nonspherical convex rigid bodies like ellipsoids or spherocylinders representing rodlike or platelike molecules or groups of atoms, with site−site interaction potentials depending both on the distance among the particles and the relative orientation. In this category of potentials, the Gay-Berne family has been studied most extensively. However, conventional molecular graphics programs are not designed to visualize such objects. Usually the basic units are atoms displayed as spheres or as vertices in a graph. Atomic aggregates can be highlighted through an increasing amount of stylized representations, e.g., Richardson ribbon diagrams for the secondary structure of proteins, Connolly molecular surfaces, density maps, etc., but ellipsoids and spherocylinders are generally missing, especially as elementary simulation units. We fill this gap providing and discussing a customized OpenGL-based program for the interactive, rendered representation of large ensembles of convex bodies, useful especially in liquid crystal research. We pay particular attention to the performance issues for typical system sizes in this field. The code is distributed as open source.

1. Introduction and Motivation

Generating three-dimensional (3D) pictures of molecular simulation output is useful if not mandatory for understanding the results and for presenting them in publications, talks, and posters. There are hundreds of molecular graphics programs. Freeware examples are MolScript,[1] VMD,[2] Raster3D,[3] Chimera,[4] AtomEye,[5] RasMol,[6] gOpenMol,[7] Jmol,[8] PyMOL,[9] and Molekel.[10] Payware examples are Cerius2,[11] Discovery Studio,[12] SYBYL,[13] and MOLCAD.[14] Inevitably, the basic units of these programs are atoms displayed as spheres or as vertices in a wireframe or “neon tube” graph. However, the Gay-Berne family of anisotropic potentials employs soft ellipsoids as basic modeling units to represent whole rodlike[15] or platelike[16] (and thus usually mesogenic) molecules, in order to speed up Monte Carlo and molecular dynamics[17] calculations by giving up intramolecular detail. Other popular choices for the same purpose are soft spherocylinders;[18] soft biaxial ellipsoids,[19] hard ellipsoids and spherocylinders,[20] and several other site−site variants[21] are employed too. The use of these nonspherical convex rigid bodies has been linked traditionally to liquid crystal research[22−24] but has later been extended to the mesoscopic description of polymers[23] and, more in general, of rigid moieties in larger molecules.[25,26] The attention to “coarse-graining” in molecular simulation has been growing, as shown by a dedicated section in a recent issue of this journal,[27] though in most cases the full potential of a “united atoms” approach is not unleashed because for simplicity researchers too often limit themselves to model functional groups or sets of nearby atoms with one large sphere[28] rather than with other more matching shapes.

Most standard molecular graphics packages can highlight atomic aggregates through an increasing amount of stylized representations, e.g., ribbons or cartoons for the secondary structure of a protein,[29,30] molecular surfaces,[31−34] density maps, etc., but ellipsoids or spherocylinders are not usually implemented. Standard programs are written to process only sets of Cartesian coordinates $\{\mathbf{r}_i\}$ but not orientations $\{\hat{\mathbf{e}}_i\}$ (for the sake of simplicity, we assume axially symmetric bodies, whose orientation is fully determined by a versor,

* Corresponding author e-mail: guido@staff.uni-marburg.de; Web page: www.staff.uni-marburg.de/~germano.

† Department of Chemistry and WZMW.

‡ Department of Mathematics and Computer Science.

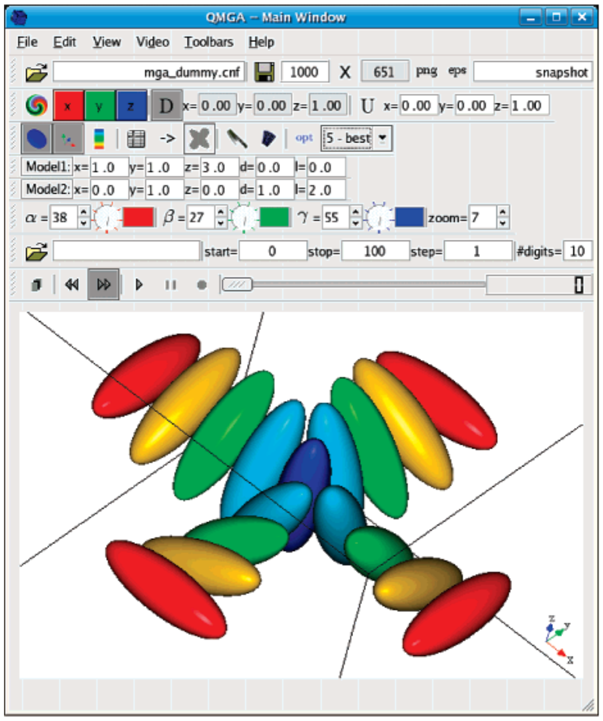
i.e., a unit vector; the generalization to the biaxial case with an orthogonal orientation matrix[35] or a quaternion[36] is straightforward). An exception is the simplified representation of DNA bases by flat biaxial ellipsoids,[37] recently implemented in some biomolecular graphics programs like Chimera.[38] However, this feature belongs to the above-mentioned category of schematic representations of specific groups of atoms, input as a set of Cartesian coordinates in PDB format. It is inflexible and inefficient when tweaked to display an arbitrary set of ellipsoids used to model a mesophase. Our attempt to employ Chimera in this sense was not satisfactory, though otherwise it is a fine and comprehensive program for its intended purposes. We converted center of mass coordinates and orientations of ellipsoids to a special data file with a much larger number of corresponding atomic coordinates. In addition to being cumbersome, this froze the program when the number of objects was within a typical range used in the study of collective properties of liquid crystalline phases, i.e., 10^4–10^5. For completeness, we mention the ORTEP[39] program that plots thermal ellipsoids for crystal structures, but clearly this is off the track for our aim, so we did not spent any time with it.

Until now, researchers in this niche resort to their own visualization code[40,41] or to workarounds with programs designed for other purposes,[42,43] possibly through conversion steps similar to the one described before. Some of these workarounds, apart from being complicated and time-consuming, preclude a visual feedback before the image is completed, i.e., the system cannot be zoomed, rotated, or sliced interactively in real time. We fill this gap providing a good dedicated molecular graphics program based on OpenGL[44] and available as open source.[45] Its name, QMGA, is an acronym for Qt-based Molecular Graphics Application, and the trailing A stands also for the first name of its principal author. A screenshot of QMGA's main window displaying a test system is shown in Figure 1. We preferred to develop a completely new program tailored for liquid crystal research rather than to extend an existing one burdened by a rich set of features useful in molecular biology, because this allowed us to focus on issues specific to liquid crystals, including the performance needed to display the large amount of objects that must typically be dealt with in this field. Of course we would be glad if our work will spur the future inclusion of QMGA's concepts and features in larger molecular graphics programs meant for general purpose.

2. Program Concepts and Features

In the following we discuss briefly the main concepts and features of our visualization program. The order in which they appear reflects to some extent their importance.

2.1. Fully Rendered View and Simplified View. A rendered picture is obviously the bare basis, since without it nothing is seen. Full rendering consists in drawing each molecule as a space-filling convex body (a sphere, an



mga_dummy.cnf 1000 X 651

x y z D x= 0.00 y= 0.00 z= 1.00 U x= 0.00 y= 0.00 z= 1.00

5 - best

Model1: x= 1.0 y= 1.0 z= 3.0 d= 0.0 l= 0.0

Model2: x= 0.0 y= 1.0 z= 0.0 d= 1.0 l= 2.0

α = 38 β = 27 γ = 55 zoom= 7

start= 0 stop= 100 step= 1 #digits= 10

Figure 1. The QMGA graphical user interface showing a test system.

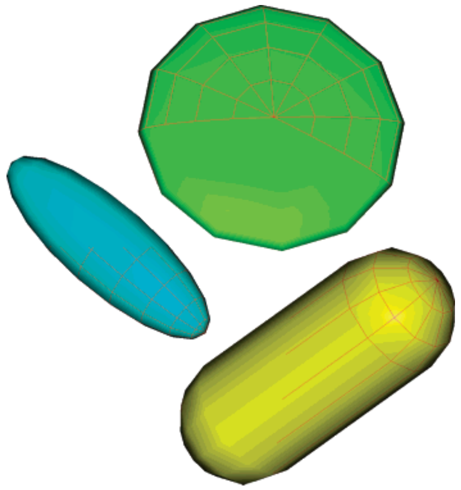


Figure 2. Oblate ($\kappa = 0.2$) and prolate ($\kappa = 3$) ellipsoids as well as a spherocylinder ($L = 2$) with wireframe overlay showing the polygonal surface structure employed by the rendering engine for a medium quality setting. The full range of render quality settings is shown in Figure 7.

ellipsoid, or a spherocylinder) approximated by a set of triangles, see Figure 2, in our case a generalized triangle strip. In stick rendering only the molecular axis $\kappa\hat{e}_i$ is drawn. Stick rendering is useful to see through the system for detecting supramolecular structures (or their absence), see Figure 3, and to reduce the computational effort when rotating or zooming a large system.

2.2. Color Coding. In conventional molecular graphics programs, the elementary objects are spheres representing atoms. The latter are usually colored according to their type along the Corey-Pauling-Koltun scheme: white for hydrogen, black or gray for carbon, blue for nitrogen, red for oxygen, etc.[6,47,48]

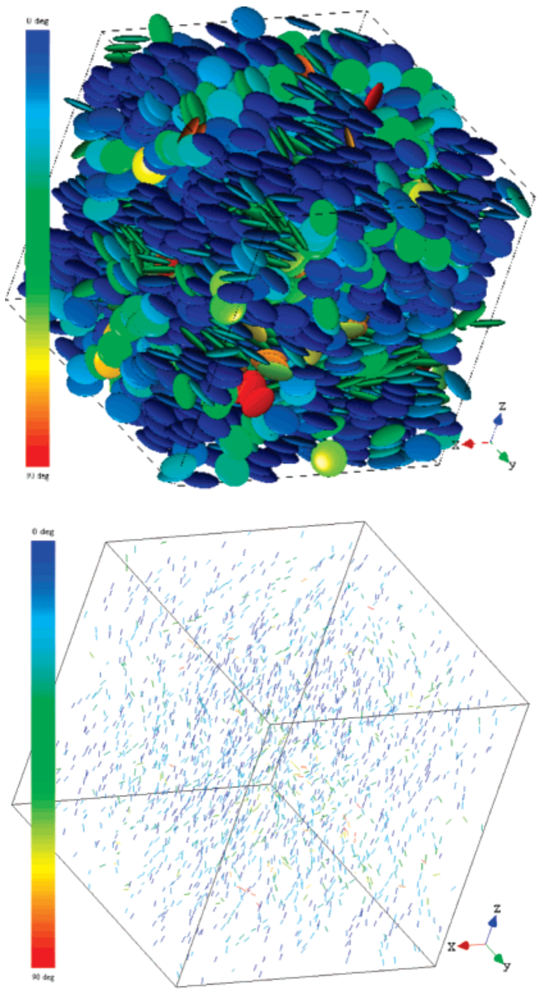


Figure 3. Nematic phase formed by soft oblate ellipsoids interacting with the GBDII potential ($\mu = 1$, $\nu = 2$, $\kappa = 0.2$, $\kappa' = 0.1$, $T^* = 12$, $P^* = 200$),[46] fully rendered and in stick view. The colormap is visible on the left.

Other coloring schemes are based on properties like hydrophobicity, charge, velocity modulus $v_i = |\mathbf{v}_i|$, and, therefore, temperature $T_i = 3m_iv_i^2/k_B$ (because of the equipartition theorem) or, for nonspherical bodies, orientation $\hat{\mathbf{e}}_i$. A color depending on the orientation is particularly useful for liquid crystals to give a first glance impression of the overall order of the phase (one color predominates in a more ordered phase) and has been used at least since the early 1990s.[40,41]

Each molecule i is colored depending on $c_i = |\hat{\mathbf{e}}_i \cdot \hat{\mathbf{n}}| \in [0,1]$, i.e. the absolute value of the scalar product between the individual molecular versor $\hat{\mathbf{e}}_i$ and an overall versor $\hat{\mathbf{n}}$ that is the same for the whole system; $\hat{\mathbf{n}}$ can be set to the director of the mesophase or to a user-defined value. The latter can be one of the three versors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$ of the Cartesian reference frame or a particular symmetry axis of the system, e.g., the cylinder axis for a cylindrical pore (see below). The director is the eigenvector corresponding to the eigenvalue with the largest absolute value of the order tensor Q:

$$Q = \frac{3}{2N}\sum_{i=1}^{N} \hat{e}_i \otimes \hat{e}_i - \frac{1}{2}E$$

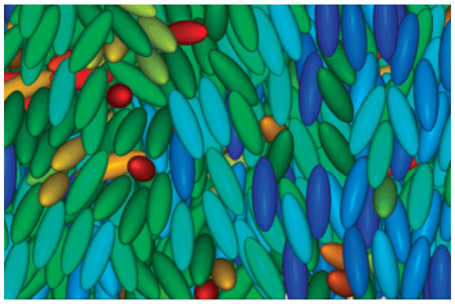


Figure 4. Closeup picture of a nematic phase formed by soft prolate ellipsoids interacting with the Gay-Berne potential ($\mu = 1$, $\nu = 3$, $\kappa = 3$, $\kappa' = 5$, $T^* = 3.45$, $\rho^* = 0.3$).[49]

In general, color coding consists in mapping a variable $x \in [a,b] \subset \mathbb{R}$ that describes a relevant property to a colormap M, i.e. a discrete set of colors, through a function $f_{\mathrm{col}}(x)$: $[a,b] \rightarrow M$. The actual form of f_{col} and M depends on the system. In the present case, $x = c_i$ is mapped to an RGB-encoded rainbowlike spectrum M_{RGB} by the function f_{RGB} (R = red, G = green, B = blue); each color is represented by a tuple of three integers R, G, B between 0 and 255. As shown on the left of Figure 3, M_{RGB} consists of 91 different colors, one for every degree of arccos $c_i \in [0, 90]$. Both the calculated director and a user defined reference versor are shown in the GUI. The user can modify his choice at runtime with an immediate effect on colorization. As an alternative in the case of mixtures, some or all molecules may be colored according to their type.

2.3. User Interface. The 3D representation can be rotated and zoomed with the mouse. The camera position information, i.e., the description from which point and distance in space the user looks upon the system, is shown using three angles and a zoom factor. All three values are continuously updated while zooming and rotating with the mouse. It is also possible to update the render area setting each orientation parameter through the keyboard. This way the user can reproduce exactly a desired viewpoint, e.g., to compare different systems. The hot keys x , y, z and c orient the system axes parallel to the screen axes in a preset manner.

2.4. Printing. A molecular graphics program is useful not only to understand one's own results but also to present them in public. To do so, image files are required and consequently the ability to take screenshots from the render area. With some graphic tools the screenshot picture's resolution depends on the size of the program window and therefore on the resolution of the monitor. As a result, it is not possible to save pictures with a resolution higher than that of the monitor, which leads to problems when these are printed on large scale, e.g., on posters. QMGA allows the user to specify the desired resolution of the picture independently of the output device, which is especially useful on large printouts, choosing at the very least between PostScript and PNG; see Figures 3−6 for examples. The aspect ratio of the picture is automatically taken care of, so that no deformations occur when setting a new resolution value or when resizing the window. Moreover, it is possible to export a screenshot as a POV-Ray[43] script, in order to achieve the final polished characteristics of a ray-traced image as well as many other features of the powerful POV-Ray program.

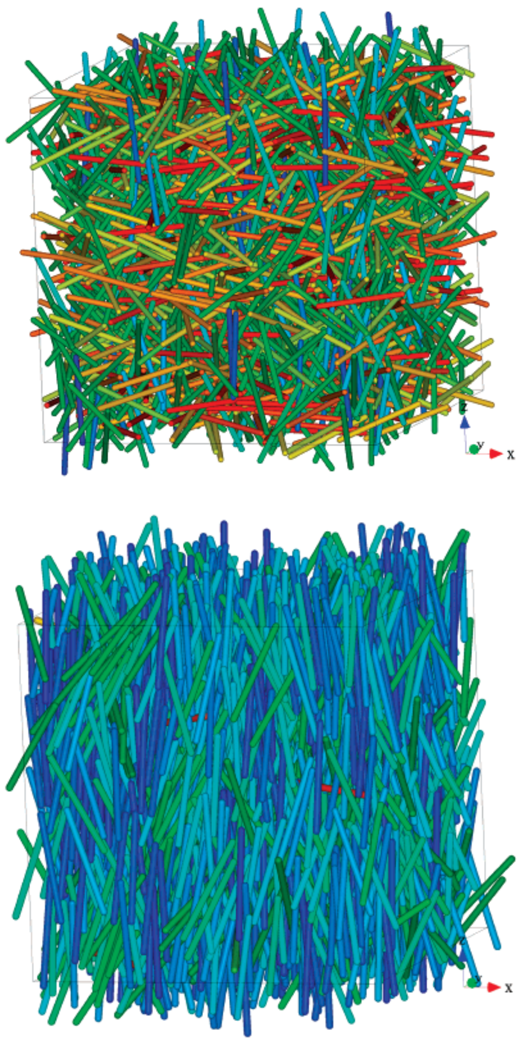


Figure 5. Spherocylinders in an isotropic (top) and nematic (bottom) phase.

2.5. Slicing. When a system is closely packed, what happens inside is not visible. However, there are cases where the inside is the interesting region: Figure 6 shows the simulation of a discotic liquid crystal inside a nanopore where the pore is sliced in half. Stick view is a possibility to look through a system, but if full render mode is wished, the choice must fall on slicing. QMGA's slice feature displays or hides objects depending on their center, meaning that no objects are truncated: they are either completely displayed or completely hidden. Slicing can take place along any combination of the coordinate axes.

2.6. Video. Since a molecular simulation usually evolves in time, the possibility to view and record animations is convenient. QMGA can load sequentially a number of files and display them one after the other, creating the impression of a motion picture. The interface allows not only for the standard actions expected from a video player (start, stop, and pause) but also forward and backward playback as well as frame capture.

With large systems the load and render times become long, resulting in a stagnant movie. In such a case it is possible (and advisable) to save all frames to disk as images and create a movie file from these. Though the recording of all displayed frames can be done by QMGA, currently there is no functionality to encode them automatically into a movie file, so this has to be done with an external program. A good freeware utility for this purpose is FFmpeg,[51] that produces, e.g., high quality AVI files.

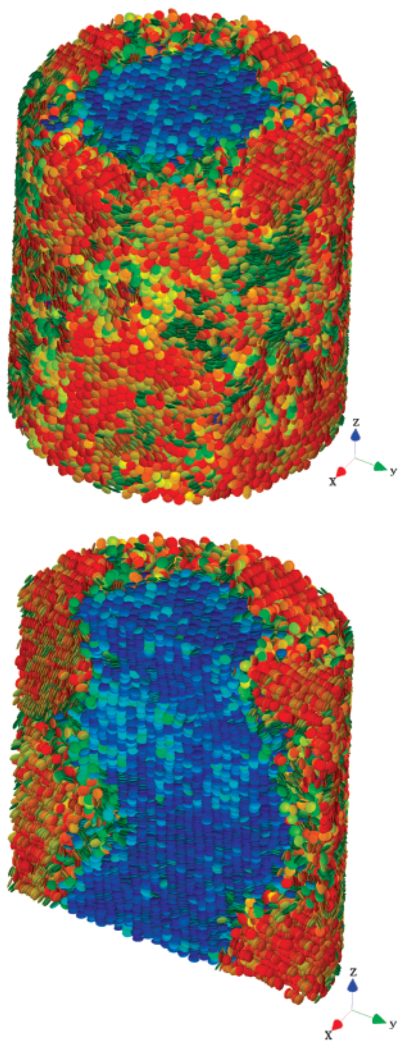


Figure 6. Snapshot from the molecular dynamics simulation of model discotic mesogens confined inside a nanopore.[50] To see that the pore organizes into a core–shell system with a columnar region in the center, the system was cut in half along the cylinder axis using QMGA's slice feature. An alternative is switching to a stick view.

2.7. Mixtures. Whereas many molecular simulations, especially coarse-grained ones of liquid crystals, deal with pure phases, there are also cases with more than one species. To accommodate for this, the internally used molecule class of QMGA has a private member of integer value that is used as a tag to divide the molecules into groups. It is then possible to assign different model parameters to each group. In an extreme case it is possible to give every single molecule its own representation by assigning a different tag to each.

For convenience, two toolbars are shown directly on the main program window to set the size parameters of the first two used objects. Since too many toolbars are confusing and many simulations deal with just one or two different molecular species, only these two were implemented. However, there is an additional window showing all used

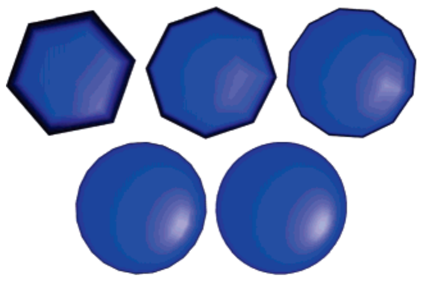


Figure 7. Top view of an oblate ellipsoid showing the effect of the render quality setting. There are five levels ranging from poor to very good. This setting enables the user to speed up the frame rate (if necessary) while working with the program and later on to produce high quality screenshots.

models with all their parameters, so that the parameters not accessible via the toolbars can be adjusted too.

2.8. Periodic Boundary Conditions. Most molecular simulations use periodic boundary conditions (PBC). From the graphical point of view this can be treated in two ways: show all molecules at their positions folded inside the unit box or employ positions without PBC applied on them. While folding is always possible and has been implemented, unfolding requires either the absolute coordinates or a sequence of folded trajectory frames.[52]

2.9. Lighting. It is mostly a mere matter of taste how one prefers the objects to look like, referring to lighting and surface. OpenGL provides the functionality to give the objects, e.g., a beamless or shiny metal or plasticlike finish. All necessary parameters are adjustable from a dialog window, and the resulting changes in colorization are shown immediately. More sophisticated effects can be achieved with the above-mentioned POV-Ray export feature.

2.10. Render Quality. The render speed is a function of parameters like the quality of the video driver, the quality of the video card, and the number of triangles to be drawn. When working with the program, smooth zoom and rotation is more important than a high-level representation. On the other hand, on a printout it is the other way around. The representation must look nice, and, since it is a still picture, render performance is not an issue any more.

To achieve a certain level of adjustability, five presets were implemented to influence the render quality of the shown objects. They range from fairly poor to an excellent, almost perfectly smooth representation. Figure 7 shows the difference on the example of a single oblate ellipsoid.

2.11. Remote File Access. Molecular simulations are often performed on remote supercomputers, while their results are visualized on the screen of a local desktop computer. To simplify file transfer, QMGA uses *ssh* to show the file system of the remote computer in a tree view that can be navigated with the mouse or keyboard. To show a file in the render area, it is copied to a temporary directory on the local machine and opened from there. This is done by a simple double-click, drag and drop or key-press. QMGA shows how much data are stored in the temporary folder and provides a button to purge all files. The remote login and file transfer, realized by *ssh* and *scp*, require a passwordless connection through an entry of the local computer's public key in the *.ssh/authorized_keys* file of the remote computer.

2.12. Saving of Options for Restart. More than 70 options are saved when the program closes and are loaded again when it is restarted. They include the last opened file, rotation and zoom settings, lighting settings, which toolbars are shown or hidden, the window position, the render mode (full or stick) and quality, etc.

3. Program Internals

3.1. Structure. QMGA is wholly written in the programming language C++ with a completely object oriented approach, as are most used libraries and toolkits. The window manager is realized with Trolltech's Qt,[53] that provides easy support for elaborate window items. For the 3D part we chose OpenGL[44] rather than the simpler VRML[54] or its successor X3D[55] because of the performance. Moreover, VRML/X3D never became as largely used and as well supported as OpenGL.

Since the output of most molecular simulation programs are text-based files with the variables describing individual molecules, we provide a small library of objects that handles a given system of molecules with respect to visualization. This library consists of three classes: (1) *Molecule* contains position, orientation, size, type, and color information for a single molecule. (2) *Colormap* reads RGB-based color values from a file and contains a function that sets a molecule's color according to a certain rule. (3) *CnfFile* (configuration file) reads all relevant data from a given simulation output file. The main components are a vector containing all *Molecule* objects and the *Colormap* object to be used for colorization.

The program basis of QMGA is given by Qt, that provides the graphical user interface. An OpenGL render area is embedded as a window frame. This area is filled with 3D objects using the information that was loaded into an instance of the *CnfFile* class. At run time it is possible to load and display different systems by overwriting the information stored in the *CnfFile* and sending the new commands to the render area.

3.2. Customization. Obviously some features will need customization to satisfy the pecularities of different users. First of all, different simulation programs will have different output file formats. However, the text parser of QMGA resides in just one single function, named *loadCnfFile()*, that is a member function of the class *CnfFile*. It is easy to modify this part of the code. All that has to be done is parsing the necessary information from the configuration file. In principle any format can be supported, if it provides at least position and orientation information for each molecule. When dealing with spherical objects and therefore there is no orientation information, the related variables can be set to arbitrary numbers.

The current format is the one used by the parallel domain decomposition molecular dynamics program *GBmega*[56] and is structured in the following way:

• header

int (number of molecules)

double (x side length of the unit box)
double (y side length of the unit box)
double (z side length of the unit box)
2 doubles (for moving boundary conditions)
• molecule information
12 doubles ($\mathbf{r}_i$,$\mathbf{v}_i$,$\hat{\mathbf{e}}_i$,$\mathbf{u}_i$), int (label), int (type tag, optional) where $\mathbf{r}_i$ is the position of molecule i , $\mathbf{v}_i$ is its velocity, $\hat{e}_i$ is its orientation, and $\mathbf{u}_i$ is its orientation velocity. Of these numbers, only the box sides, $\mathbf{r}_i$, and $\hat{e}_i$, are used for visualization purposes. The program can deal also with noncubic unit boxes; in this case, a 3 × 3 matrix must be input to specify it.

For example, the artificially created file displayed in Figure 1 looks like this:

```
17
9.0
9.0
9.0
0.0 0.0
-4.0  0.0  0.0   0 0 0   0.0  1.0  0.0   0 0 0  1
-3.0  0.0  0.0   0 0 0   0.0  2.0  1.0   0 0 0  2
-2.0  0.0  0.0   0 0 0   0.0  1.0  1.0   0 0 0  3
-1.0  0.0  0.0   0 0 0   0.0  1.0  2.0   0 0 0  4
 0.0  0.0  0.0   0 0 0   0.0  0.0  1.0   0 0 0  5
 1.0  0.0  0.0   0 0 0   0.0 -1.0  2.0   0 0 0  6
 2.0  0.0  0.0   0 0 0   0.0 -1.0  1.0   0 0 0  7
 3.0  0.0  0.0   0 0 0   0.0 -2.0  1.0   0 0 0  8
 4.0  0.0  0.0   0 0 0   0.0 -1.0  0.0   0 0 0  9
 0.0 -4.0  0.0   0 0 0   1.0  0.0  0.0   0 0 0 10
 0.0 -3.0  0.0   0 0 0   2.0  0.0  1.0   0 0 0 11
 0.0 -2.0  0.0   0 0 0   1.0  0.0  1.0   0 0 0 12
 0.0 -1.0  0.0   0 0 0   1.0  0.0  2.0   0 0 0 13
 0.0  1.0  0.0   0 0 0  -1.0  0.0  2.0   0 0 0 14
 0.0  2.0  0.0   0 0 0  -1.0  0.0  1.0   0 0 0 15
 0.0  3.0  0.0   0 0 0  -2.0  0.0  1.0   0 0 0 16
 0.0  4.0  0.0   0 0 0  -1.0  0.0  0.0   0 0 0 17
```

Color coding is another aspect likely to be customized. Again, this is simple to do, because the whole relevant instructions that determine which color a molecule shall be given is found in just three rather short functions. One of these is a member function of CnfFile called *colorizeMolecules()* with the main purpose of sending all read molecules successively to another function, that is a member of the *Colormap* class. The name of this function is *setColor()*, and here is the most likely place where a change has to be made. Notice that *setColor()* comes in two different overloaded versions to handle both colorization by axis and colorization by type. Figure 8 shows the code of the currently used version of *setColor()* that realizes the colorization by axis as described earlier.

Last, parts of the GUI are expected to be modified, e.g., to display specific data values. This can be achieved intuitively with the Qt designer, a graphical tool.

4. Performance

Several optimization approaches were used to reduce the workload on the render engine; the most important ones are described below. Benchmarks conclude this section.

4.1. Scene Graph versus Direct Rendering. Initially, the OpenGL render area was realized with little effort resorting to SiM's Coin3D toolkit.[57] Coin3D is an open source library consisting of a collection of objects like ready to use light models, standard forms (sphere, cone, etc.), and a mechanism to assemble everything into a scene graph. Coin3D behaves very much like SGI's OpenInventor,[58] that at the start of the project was still payware, and provides a set of classes that can be used directly to display OpenGL content in a Qt window frame, i.e., the connection between Coin3D and Qt was already built in. The scene shown in the render area was constructed completely relying on Coin3D classes, to a large extent with a single for-loop over all *Molecule* objects inside *CnfFile*. Another example for the help these toolkits provide is how the scene is saved to file. Qt contains a file dialog and Coin3D a number of functions to export the content of the render area to image file in various formats.

```
Molecule* mga::Colormap::setColor(
 Molecule* moleculeTmp, vector<double> &director ) const
{
  if( (moleculeTmp != 0) && (director.size() == 3) )
    {
      double orientationX=moleculeTmp->getOrientationX();
      double orientationY=moleculeTmp->getOrientationY();
      double orientationZ=moleculeTmp->getOrientationZ();
      int numOfMapLines=redVector.size();
      int mapLineNr = int( acos( fabs(
       orientationX*director.at(0) +
       orientationY*director.at(1) +
       orientationZ*director.at(2)   )
       )/M_PI*2*( numOfMapLines ) );
      if( mapLineNr == 90 ) { mapLineNr = 89; }
      moleculeTmp -> setRGB(
        getRed( mapLineNr ),
        getGreen( mapLineNr ),
        getBlue( mapLineNr ) );
    }
  else
    {
       cerr << "Error: no molecule given to setColor,
        or director corrupt." << endl;
     }
  return( moleculeTmp );
}
```

Figure 8. Function *setColor()* of the class *Colormap*. The variables *orientationX*/*Y*/*Z* are the components of the vector describing the orientation of the object; the vector object *director* was calculated previously and represents the director of the mesophase. The colormap itself contains a certain number of RGB coded colors arranged in lines. The line number of the color is calculated by taking the scalar product of the molecular orientation with the director and multiplying the result with the total number of colors in the map.

While such a simple approach based on Coin3D is shared by other molecular graphics programs[10] and works quite well for not too big systems of up to 10^4 molecules, it becomes very slow when large systems of about 10^5 molecules are rendered. Even if the program actually does still run stable with that much workload, the frame rate is too low for an effective use. This is consistent with the experience with Chimera described in the introduction: Chimera uses an even slower scene graph based on VRML. For this reason, we redesigned completely the render area rewriting it from scratch without Coin3D and fitting it to the special purpose of displaying many identical objects. The only limitation is the assortment of supported image output formats, that was reduced to PNG and PostScript because it was too much work to implement the complete list provided by Coin3D (JPEG, TIF, diverse raw pixel formats, etc.) without this library.

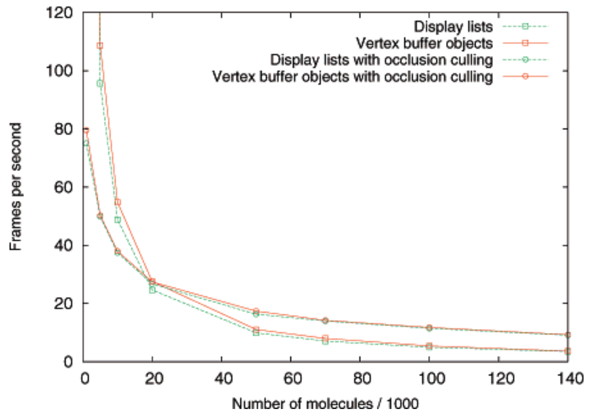


Figure 9. Graphic display of the benchmark results in Table 1. There is little difference between display lists and vertex buffer objects. Notice the crossover for the use of occlusion culling at about 20 000 molecules. Error bars are omitted because, except for 1000 molecules, they are smaller than the symbols used for the data points.

Table 1. Frames per Second for a System of *N* Discotic Ellipsoids in a Columnar Phase, Using Display Lists (DL) or Vertex Buffer Objects (VBO) without or with Occlusion Culling (OC)[a]

N/1000	DL	VBO	DL+OC	VBO+OC
1	485.2	585.7	75.0	79.6
5	95.6	108.7	49.9	50.3
10	48.8	54.8	37.4	38.0
20	24.7	27.6	26.8	27.4
50	9.9	11.1	16.3	17.4
70	7.1	8.0	14.0	14.2
100	4.9	5.4	11.5	11.8
140	3.5	3.6	9.1	9.3

[a] Errors are below 5%.

4.2. Display Lists versus Vertex Buffer Objects. To speed up object rendering we tried both OpenGL's display lists (DLs) and vertex buffer objects (VBOs). An OpenGL DL "precompiles" a model in the graphics memory, so that later it can be drawn by just calling a specific OpenGL function with the index of the DL. Prior to that we move to the specific position and do other transformations to render the model in the desired way. So the main rendering code remains the same, while it becomes easy to replace the molecular model by precompiling another one with a new DL.

An OpenGL VBO can be used similarly in many ways, but VBOs are more lightweight than DLs: They contain by default less information about the object, e.g., no transformations and materials. So the graphics driver can avoid overhead work needed to sort out whether this information is present and must be taken into account. The speed-up achievable by exchanging DLs with VBOs depends on the software and hardware environment. On our test system, VBOs provide just a slightly higher frame rate; see Table 1 and Figure 9. However, VBOs also yield a shorter and cleaner code, while DLs are at risk of being removed from future OpenGL releases, so we preferred VBOs.

4.3. Level of Detail. The next optimization makes use of the common render technique "level of detail" (LOD). Objects near to the camera are rendered with more detail than far away ones. Here we do not use just a linear approach but a self-adjusting one. The user chooses a quality level, and this defines the maximum and minimum rendering quality. If the frame rate drops below a certain level, then the quality of the particles automatically starts dropping from back to front, until either the frame rate becomes high enough again or all particles are drawn with minimum quality. On the other hand, if the frame rate is high enough, then the quality of the drawn particles is enhanced from front to back. It makes sense to use LOD though we employ an orthographic view, because it is still more probable that an object far away from the camera is covered, even if in part, than an object near to the camera.

4.4. Occlusion Query. In a dense system with many particles, it is most likely not necessary to render all of them. If those nearest to the camera are rendered first, then it may be possible to clip many others farther away. For this aim we make use of the OpenGL extension *GL_ARB_occlusion_query*, that asks the graphics card whether the next models must be drawn. Since it does not make much sense to query every single particle, we group them together dividing the bounding box of the system into $n \times n \times n$ smaller cells. Every particle becomes a member of one of these cells according to its center. After every particle has been assigned to one cell, the cells are resized to fit the complete models of the particles and not just their center points. If later we wish to know whether a group must be drawn, then the graphic card can provide an answer. If the answer is negative, then all the particles inside this cell can be discarded. This way, if the cells are drawn from front to back, then the rendering of many particles can be avoided.

4.5. Backface Culling. It is not necessary to render the back of an opaque object. OpenGL can take care of this by itself, if instructed with a simple library call, and so the rendering work is halved. However, this leads to a performance gain of just about 10%, because the library's internal calculations to find out what exactly is the back side of each item in its present orientation to the camera are almost as time-consuming as the avoided rendering.

4.6. Benchmarks. The benchmark results for QMGA presented here were performed on a computer with an AMD Athlon 64 3500+ processor running at 2.2 GHz with 2 GB RAM and an NVidia 8600GT graphic card adapter. The operating system was Fedora Core 7 Linux and the compiler g++ 4.1.2. To achieve meaningful and stable results, a benchmark option was introduced into QMGA. When activated, a series of random rotations is executed while measuring the current and average frame rates. The system used for the measurement consisted of about 140 000 discotic molecules in a columnar phase. To monitor the render speed as a function of the number of particles, the latter were sorted by their distance from the origin and included into larger and larger systems, whose shape was spherical because of the sorting. For each system the benchmark was run several times through 100 rotations with and without occlusion culling (OC). Active OC increases significantly the render performance in systems with more than approximately 20 000 molecules. Below this number the performance is reduced but is still so high that it is safe to leave OC always on.

Table 1 and Figure 9 display how many frames per second (fps) were rendered for test configurations ranging from 1000 to 140 000 molecules. Even with more than 100 000 molecules per configuration, QMGA behaves quite well, and even better when occlusion culling is activated. As few as 8 fps still feel almost completely fluent, and only below about 5 fps some jerkiness starts becoming noticeable.

5. Conclusions

We have filled a gap among molecular graphics programs providing and discussing an open source code for the visualization of large sets of convex bodies like ellipsoids and spherocylinders. This is useful especially not only for the coarse-grained modeling of liquid crystals but also of (bio)polymers and other chemical compounds, with anisotropic site−site potentials belonging to the Gay-Berne family. A rich set of features has been implemented employing easy to use toolkits (Qt) and state of the art libraries (OpenGL). Special attention has been dedicated to performance when displaying large systems of the order of 10^5 molecules. The final result was a useful and fast program fulfilling purposes that previously could be achieved only with difficulty or not at all.

Acknowledgment. We thank Prof. M. P. Allen, University of Warwick, for sharing with us his proprietary molecular graphics program for convex bodies MG and his Ph.D. student P. O'Brien for providing us the spherocylinder configurations shown in Figure 5. We are also grateful to Prof. M. Guthe, Philipps-University Marburg, for help with implementing OpenGL vertex buffer objects. Figures 3 and 6 display results of molecular dynamics simulatons run on the IBM Regatta p690+ parallel computer JUMP of the John von Neumann Institute for Computing in Jülich.

References

(1) Kraulis, P. J. MolScript − A program to produce both detailed and schematic plots of protein structures. *J. Appl. Crystallogr.* **1991**, *24*, 946. http://www.avatar.se/molscript (accessed Dec 20, 2007).

(2) Humphrey, W.; Dalke, A.; Schulten, K. VMD − Visual Molecular Dynamics. *J. Mol. Graphics* **1996**, *14*, 33. http://www.ks.uiuc.edu/research/vmd (accessed Dec 20, 2007).

(3) Merritt, E. A.; Bacon, D. J. Raster3D: Photorealistic molecular graphics. *Method. Enzymol.* **1997**, *277*, 505. http://skuld.bmsc.washington.edu/raster3d (accessed Dec 20, 2007).

(4) Pettersen, E. F.; Goddard, T. D.; Huang, C. C.; Couch, G. S.; Greenblatt, D. M.; Meng, E. C.; Ferrin, T. E. UCSF Chimera − A visualization system for exploratory research and analysis. *J. Comput. Chem.* **2004**, *25*, 1605. http://www.cgl.ucsf.edu/chimera (accessed Dec 20, 2007).

(5) 5. Li, J. AtomEye: an efficient atomistic configuration viewer. *Model. Simul. Mater. Sc.* **2003**, *11*, 173.

(6) RasMol homepage. http://www.umass.edu/microbio/rasmol (accessed Dec 20, 2007).

(7) gOpenMol homepage. http://www.csc.fi/gopenmol (accessed Dec 20, 2007).

(8) Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org (accessed Dec 20, 2007).

(9) Delano, W. L. The PyMOL Molecular Graphics System, DeLano Scientific, Palo Alto, CA, U.S.A. http://pymol-.sourceforge.net (accessed Dec 20, 2007).

(10) Molekel homepage. http://www.cscs.ch/molekel (accessed Dec 20, 2007).

(11) Cerius2 homepage. http://www.accelrys.com/products/cerius2 (accessed Dec 20, 2007).

(12) Discovery Studio homepage. http://www.accelrys.com/products/dstudio (accessed Dec 20, 2007).

(13) SYBYL homepage. http://www.tripos.com (accessed Dec 20, 2007).

(14) Brickmann, J.; Keil, M.; Exner, T.; Marhöfer, R. Molecular graphics − Trends and perspectives. *J. Mol. Mod.* **2000**, *6*, 328. MOLCAD: MOlecular Computer Aided Design. http://www.molcad.com (accessed Dec 20, 2007).

(15) Gay, J. G.; Berne, B. J. Modification of the overlap potential to mimic a linear site-site potential. *J. Chem. Phys.* **1981**, *74*, 3316.

(16) Bates, M. A.; Luckhurst, G. R. Computer simulation studies of anisotropic systems. XXVI. Monte Carlo investigations of a GayBerne discotic at constant pressure. *J. Chem. Phys.* **1996**, *104*, 6696.

(17) Allen, M. P.; Germano, G. Expressions for forces and torques in molecular simulations using rigid bodies. *Mol. Phys.* **2006**, *104*, 3225.

(18) Kihara, T. Convex molecules in gaseous and crystalline states. *Adv. Chem. Phys.* **1963**, *5*, 147.

(19) Berardi, R.; Fava, C.; Zannoni, C. A Gay-Berne potential for dissimilar biaxial particles. *Chem. Phys. Lett.* **1998**, *297*, 8.

(20) Allen, M. P.; Evans, G. T.; Frenkel, D.; Mulder, B. M. Hard convex body fluids. *Adv. Chem. Phys.* **1993**, *86*, 1.

(21) Martinez-Haya, B.; Cuetos, A.; Lago, S.; Rull, L. F. A novel orientation-dependent potential model for prolate mesogens. *J. Chem. Phys.* **2005**, *122*, 024908.

(22) Zannoni, C. Molecular design and computer simulations of novel mesophases. *J. Mater. Chem.* **2001**, *11*, 2637.

(23) Computational soft matter: From synthetic polymers to proteins; Attig, N., Binder, K., Grubmüller, H., Kremer, K., Eds.; Forschungszentrum Jülich: Jülich, 2004. http://www.fz-juelich.de/nic-series/volume23 (accessed Dec 20, 2007).

(24) Wilson, M. R. Progress in computer simulations of liquid crystals. *Int. Rev. Phys. Chem.* **2005**, *24*, 421.

(25) Prampolini, G. Parametrization and validation of coarse grained force-fields derived from ab initio calculations. *J. Chem. Theory Comput.* **2006**, *2*, 556.

(26) Amovilli, C.; Cacelli, I.; Cinacchi, G.; Gaetani, L. D.; Prampolini, G.; Tani, A. Structure and dynamics of mesogens using intermolecular potentials derived from ab initio calculations. *Theor. Chem. Acc.* **2007**, *117*, 885.

(27) Voth, G. A. Introduction: Coarse-graining in molecular modeling and simulation. *J. Chem. Theory Comput.* **2006**, *2*, 463.

(28) Venturoli, M.; Sperotto, M. M.; Kranenburg, M.; Smit, B. Mesoscopic models of biological membranes. *Phys. Rep.* **2006**, *437*, 1.

(29) Richardson, J. S. The anatomy and taxonomy of protein structure. *Adv. Protein Chem*. **1981**, *34*, 167.

(30) Richardson, J. S. Schematic drawings of protein structures. *Method. Enzymol.* **1985**, *115*, 359.

(31) Lee, B.; Richards, F. M. The interpretation of protein structures: Estimation of static accessibility. *J. Mol. Biol*. **1971**, *55*, 379.

(32) Connolly, M. L. Analytical molecular surface calculation. *J. Appl. Crystallogr*. **1983**, *15*, 548.

(33) Connolly, M. L. Solvent-accessible surfaces of proteins and nucleic acids. *Science* **1983**, *221*, 709.

(34) Connolly, M. L. The molecular surface package. *J. Mol. Graph*. **1993**, *11*, 139.

(35) Goldstein, H.; Poole, C. P.; Safko, J. L. *Classical mechanics*, 3rd ed.; Addison Wesley: San Francisco, 2002.

(36) Altmann, S. L. *Rotations, quaternions and double groups*; Oxford University Press: Oxford, 1986.

(37) Max, N. Hierarchical molecular modelling with ellipsoids. *J. Mol. Graph. Model*. **2004**, *23*, 233.

(38) Couch, G. S.; Hendrix, D. K.; Ferrin, T. E. Nucleic acid visualization with UCSF Chimera. *Nucleic Acids Res*. **2006**, *34*, e29.

(39) Burnett, M. N.; Johnson, C. K. *ORTEP-III: Oak Ridge Thermal Ellipsoid Plot program for crystal structure illustrations*; Oak Ridge National Laboratory Report ORNL-6895; 1996. http://www.ornl.gov/sci/ortep (accessed Dec 20, 2007).

(40) Chiccoli, C.; Pasini, P.; Semeria, F.; Zannoni, C. Three-dimensional visualization of molecular organization and phase transitions in liquid crystal lattice models. *Int. J. Mod. Phys. C* **1992**, *3*, 1209.

(41) Berardi, R.; Emerson, A. P. J.; Zannoni, C. Monte Carlo investigations of a Gay-Berne liquid crystal. *J. Chem. Soc., Faraday Trans.* **1993**, *89*, 4069.

(42) AVS/Express homepage. http://www.avs.com (accessed Dec 20, 2007).

(43) POV-Ray homepage. http://www.povray.org (accessed Dec 20, 2007).

(44) OpenGL homepage. http://www.opengl.org (accessed Dec 20, 2007).

(45) QMGA homepage. http://qmga.sourceforge.net (accessed Dec 20, 2007).

(46) Caprion, D.; Bellier-Castella, L.; Ryckaert, J.-P. Influence of shape and energy anisotropies on the phase diagram of discotic molecules. *Phys. Rev. E* **2003**, *67*, 041703.

(47) Corey, R. B.; Pauling, L. C. Molecular models of amino acids, peptides, and proteins. *Rev. Sci. Instrum*. **1953**, *24*, 621.

(48) Koltun, W. L. Precision space-filling atomic models. *Biopolymers* **1965**, *3*, 665.

(49) Germano, G.; Allen, M. P.; Masters, A. J. Simultaneous calculation of the helical pitch and the twist elastic constant in chiral liquid crystals from intermolecular torques. *J. Chem. Phys*. **2002**, *116*, 9422.

(50) Stillings, C.; Martin, E.; Steinhart, M.; Pettau, R.; Paraknowitsch, J.; Geuss, M.; Schmidt, J.; Germano, G.; Schmidt, H. W.; Gösele, U.; Wendorff, J. H. Nanoscaled discotic liquid crystal/polymer systems: Confinement effects on morphology and thermodynamics. *Mol. Cryst. Liq. Cryst*. **2008**, accepted for publication.

(51) FFmpeg homepage. http://ffmpeg.mplayerhq.hu (accessed Dec 20, 2007).

(52) Allen, M. P.; Tildesley, D. J. *Computer simulation of liquids*, paperback ed.; Oxford University Press: Oxford, 1989.

(53) Qt homepage. http://www.trolltech.com/products/qt (accessed Dec 20, 2007).

(54) VRML97 and Related Specifications. http://www.web3d.org/x3d/specifications/vrml (accessed Dec 20, 2007).

(55) X3D and Related Specifications. http://www.web3d.org/x3d (accessed Dec 20, 2007).

(56) Wilson, M. R.; Allen, M. P.; Warren, M. A.; Sauron, A.; Smith, W. Replicated data and domain decomposition molecular dynamics techniques for simulation of anisotropic potentials. *J. Comput. Chem*. **1997**, *18*, 478.

(57) Coin3D homepage. http://www.coin3d.org (accessed Dec 20, 2007).

(58) OpenInventor homepage. http://oss.sgi.com/projects/inventor (accessed Dec 20, 2007).

CT700192Z