Molecular Dynamics Visualization with XML and VRML

B. Arun
arun@wiproge.med.ge.com

V. Chandru
chandru@csa.iisc.ernet.in

A. D. Ganguly
ganguly@cs.unc.edu

S. Manohar
manohar@csa.iisc.ernet.in
Department of Computer Science and Automation
Indian Institute of Science
Bangalore - 560012, India

Abstract

A new Extensible Markup Language (XML) application, Molecular Dynamics Language (MoDL) has been developed. MoDL provides a simple, but powerful tool for molecular dynamics visualization and has been developed by combining, for the first time, the strengths of XML and the Virtual Reality Modeling Language. The details of MoDL, its implementation and examples of its use are presented in this paper.

1. Introduction

Among all disciplines, chemistry needs to move away from the restrictions of figures on two dimensional paper to a dynamic 3D world the most. Chemical simulations output enormous amounts of data; making sense out of it is a herculean task. Being able to visualize a simulation, like following the trajectory of one molecule as it moves within other molecular structures can provide a better understanding of the various concepts at work and provide new insights.

Researchers in molecular dynamics at the Indian Institute of Science have long sought a way by which results of their computational simulations can be visualized. A specific problem that has been of interest is that of the behavior of a methane molecule when it interacts with a Zeolite cage. It has been known by the simulations that the methane molecule moves into the Zeolite cage and gets trapped inside. However, given the structure of methane and the bond structure of the cage it was puzzling as to how the methane molecule managed to get into the cage in the first place. Even though the exact coordinates of all the atoms of the cage as well as the coordinates of the methane molecule were available from the simulations, it has not been possible to visualize the exact moment when the methane molecule entered the Zeolite cage. The institute has high end molecular dynamics packages like Biosym running on SGI Onyx machines; these and other packages were unable to provide clear insights into the process.

At the Perceptual Computing lab we have brought together the expressive power of XML and the descriptive power of VRML to create a simple but powerful visualization tool for molecular dynamic visualization.

Extensible Markup Language (XML) is the emerging standard for structured data exchange over the Web. XML is a part of the World Wide Web Consortium's standardization efforts to make the Internet a better place to work in. XML is a strict subset of Standard Generalized Markup Language (SGML) which is the mother of all markup languages. The XML Specification lays down rules that allows users to define a set of tags which makes sense to a particular community. Since the release of the XML specification in February 1998, hectic activity has been going on both in the industry and the academia. As a result, numerous XML applications and tools are available and many more are being developed by the month. Mathematical Markup Language (MathML), Synchronized Multimedia Markup Language (SMIL), Web Interface Description Language (WIDL), to name just a few, are some of the XML applications that have made it big. XML is not being used as extensively as HTML due to lack of adequate development in related technologies like Extensible Style Language (XSL) and Document Object Model (DOM). Once these technologies are ready for use, XML will be all set to rule the Web.

Virtual Reality Modeling Language (VRML) is the ISO standard for representing 3D content on the Web. It is a text based file format for describing 3D objects and worlds. Along with 3D graphics, it also combines multimedia - audio and video. VRML already has extensive usage in medicine, engineering and scientific visualization, enter-

0-7695-0643-7/00 $10.00

tainment and education. VRML worlds can be viewed from inside browsers like Netscape and IE by using a VRML plugin. Various standalone VRML viewers also exist.

We have combined XML and VRML for visualizing chemical simulations over the web. The simulation data is marked up in a new XML application which is then converted to VRML for display on a Web browser using a plugin. We present the details of the language, the implementation and the results of this work in this paper.

2. Related Work

Among the first community of users of XML were the Chemists. The Chemical Markup Language (CML) that predates XML, is a powerful generic tool for management of molecular and technical information. Although it is possible to view 2D and 3D structures using a CML viewer, visualization is not a design goal of CML. Moreover, it does not provide support for chemical reactions or molecular dynamics in its present state of development. CML is still evolving and efforts are on to make its scope as wide as possible. Programs like MolScript and Prolet generate VRML from popular chemical data formats like MOL and PDB. Other tools like Rasmol can read in many file formats and display the structure in various forms like ball-stick and space-filling. It also provides the option of saving the structure as an image file like GIF. All the packages mentioned above deal only with static structures. X-Mol is a package that allow dynamics to be viewed. Using this software, users can view dynamic simulation given the coordinates of various atoms/molecules over the simulation period. X-Mol also allows the simulations to be saved as a movie.

XML-VRML combination has been used previously for some stock-exchange data visualization and web-site maps. In the latter case, the web page is described using a set of tags and this data is then converted into a VRML world that allows users to navigate the page in the 3D space.

3. Molecular Dynamics Language (MoDL)

During the design of this new markup language for visualizing chemical simulations; the foremost thing in our minds was to keep the language simple. We also have tried to make it intuitive for chemists, who are the prime target group of users.

The primary elements in this markup language are **atom, molecule** and **bond** which are self-explanatory. Each of these can be assigned types and attributes can be specified. Other elements like **bond-table, plot** and **curve** provide advanced capabilities and will be explained later.

3.1. Basic Features

A MoDL file begins with the tag **modl** and ends with the tag **modl** which means everything is included in the **modl** element. A modl document is then divided into a **head** and a **body**. The body contains information related to the various objects being visualized while the head holds information about the properties of the system. The head is used to specify attributes like the duration of the simulation, the size of the steps if the user wants to step through the simulation, whether bonds should be generated automatically or not, whether the visualization should loop through the simulation continuously or stop after one run. This tool can also be used to visualize static models. In this case, the code size of the VRML file can be reduced by assigning the value *off* to the *animation* attribute of the **head**.

The body contains the atoms, molecules and bonds to be displayed. For every atom, radius, color and position can be specified. Molecules can contain atoms and bonds. Every molecule is specified a position, an axis and an angle of rotation about this axis to uniquely specify its location in 3D. Bonds can be specified between atoms by first assigning unique identifiers to the atoms (using the *id* attribute) and then specifying these identifiers in the attributes of a **bond** element. There is a **DEFINE** construct which allows users to define particular types of atoms, molecules, bonds or radicals and then use them later to instantiate objects of that type. The use of these tags are illustrated with some examples below:

Oxygen and hydrogen atoms can be defined like this:

```
<DEFINE type='atom' name='O'
      radius='0.3' color='0 0 1' />
<DEFINE type='atom' name='H'
      radius='0.2' color='1 0 0' />
```

Here oxygen atoms are blue and have radius 0.3, hydrogen atoms are red and have radius 0.2. Using this we can now define a water molecule:

```
<DEFINE type='molecule' name='Water' >
  <atom type='O' id='o1'
          position='0 0 0' />
  <atom type='H' id='h1'
          position='-2 1 0.5' />
  <atom type='H' id='h2'
          position='2 1 0.5' />
  <bond from='h1' to='o1' />
  <bond from='h2' to='o1' />
</DEFINE>
```

Typically, bonds form between atoms when the distance between them is less than some particular value. We have used this concept so that authors don't have to explicitly

write every bond they want in the molecular structure. The maximum inter-atomic distance between pairs of atoms can be recorded in the head of the document in a **bond-table** element, and bonds will be put automatically between atoms where the distance is less than the specified value. In addition to this, authors can still explicitly put bonds between atoms where they want to.

3.2. Dynamics

Until now, everything has been static. To introduce dynamism to the visualization, we need to specify things like period of the clock, whether the clock should loop or stop after one period. The clock period and looping are specified in the **head**. **Meta** elements and **DEFINE** constructs also appear in the head. The **body** holds the elements for visualization. To place two water molecules in 3D space we write in the body the following:

```
<molecule type='Water' id='w1'
            position='0 0 0' />
<molecule type='Water' id='w2'
            position='4 0 2' />
```

To move the atoms/molecules we have two elements **TRANSLATE** and **ROTATE**. To translate one needs to specify a time-instant and the position of the atom/molecule at that time instant. To rotate one needs to specify an axis and the angle of rotation about it. The axis of rotation will be passing through the centre of mass of the object being rotated. Both of these elements are generic which means they can move either an atom or a molecule; the id of the object to be moved needs to be specified in the *object* attribute of TRANSLATE and ROTATE. ROTATE-ing an atom will have no effect since atoms are represented by spheres.

Now to move the water molecule named w2, we put w2 in a TRANSLATE element, like this:

```
<TRANSLATE object='w2' t='0.2'
            position='3 0 1' />
```

This means that when the clock is at 0.2 of the period, the molecule w2 will be at (3,0,1). For the intermediate time instants, the w2's position is calculated by using linear interpolation. Using TRANSLATE again, w2's position can be specified at several time instants. The TRANSLATE object is generic, i.e. it can move not only molecules, but atoms, radicals and bonds as well.

Another useful thing is to be able to view the simulation from any point in the 3D space. Users viewing the simulation can navigate around in the 3D scene using the navigation controls present in VRML browsers. In addition, users can specify any of the atoms or molecules to be viewpoints in the MoDL file, jump to its position and view the simulation. Users can also sit on one of the dynamic atoms and go for a ride, seeing the simulation as that particular atom would see it.

3.3. 2D and 3D Plots

Chemists typically want to see plots and graphs of various variables when the simulation is running. To facilitate this, MoDL has support for 2D and 3D plots, which can be static or dynamic. In dynamic plots, users will be able to see the graph changing as the simulation proceeds. Plots are specified using the **plot** element. They can be positioned anywhere in the 3D space of the visualization. A plot may have any number of curves, specified by the **curve** element inside **plot**. Three plot styles are supported - points, lines or histograms. They are specified by assigning the *type* attribute of **curve** to *points, lines* or *histograms*. Each curve in a plot can be specified a different type and color, making it easy to distinguish them (Fig 1). The data points are specified inside **curve** using the element **point**. Every **point** has an attribute *t* specifying the time when it should be displayed. To make a point static, *t* needs to be assigned a value 0.

The image below shows a "real-life" example. All the atoms are same but have been given different colors to distinguish between the fact that atoms in the middle are fluid while those on top and bottom form walls enclosing the fluid. The blue wavy line in the plot shows the variation of kinetic energy of the fluid atoms with time. The green and red wavy lines show potential energy and the white one shows the total energy of the system.

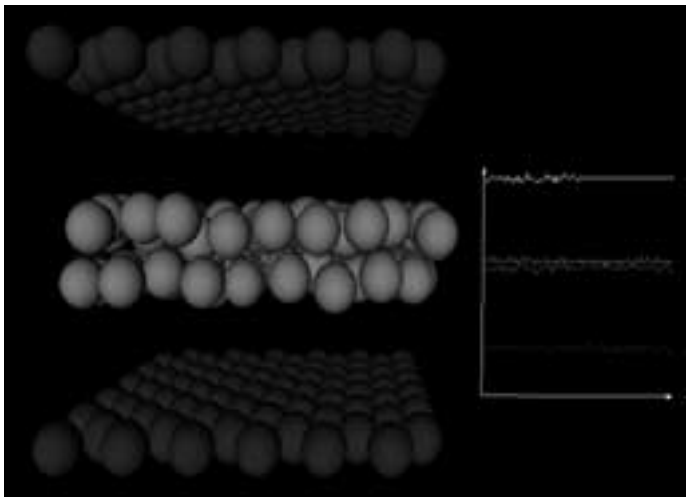

Figure 1. Plots of Potential, Kinetic and total energy of the System

3.4. Examples

Given below is an example of a methane molecule in MoDL and a screen shot (Fig 2) of it in VRML.

```
<modl>
  <head>
    <meta name='title'
          content='Example'/>
    <DEFINE type='atom' name='C'
           color='1 1 0' radius='0.3'/>
    <DEFINE type='atom' name='H'
           color='1 0 0' radius='0.2'/>
    <DEFINE type='molecule' name='Methane'>
      <atom type='C' id='c1'
            position='0 0 0'/>
      <atom type='H' id='h1'
            position='1 0.7 0.3'/>
      <atom type='H' id='h2'
            position='-1 0.7 0.3'/>
      <atom type='H' id='h3'
            position='-1 -0.7 0.3'/>
      <atom type='H' id='h4'
            position='1 -0.7 0.3'/>
       <bond from='c1' to='h1'/>
       <bond from='c1' to='h2'/>
       <bond from='c1' to='h3'/>
       <bond from='c1' to='h4'/>
    </DEFINE>
  </head>
  <body>
    <molecule type='Methane'
          position='0 0 0'
          viewpoint='true'/>
  </body>
</modl>
```

Figure 2. Methane

In the next example, referred to in the introduction, a methane molecule enters a Zeolite Cage made up of Sodium, Potassium, Sulphur and Oxygen, and gets trapped in it.

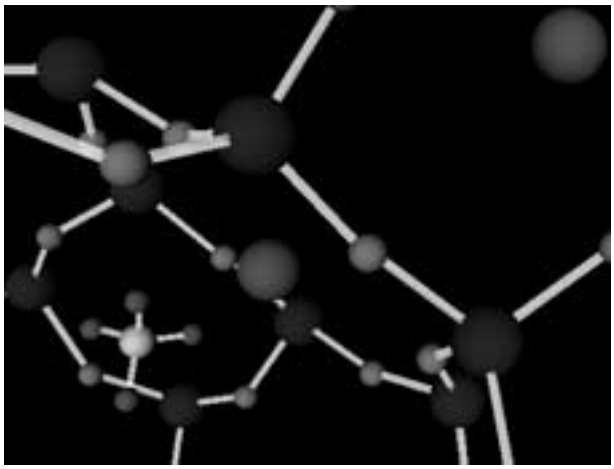

Figure 3. Methane trapped in Zeolite

This was a prime example of how important visualization is for chemists. Even though the methane molecule is bigger than the ring through which it enters the cage, it is able to do so by rotating itself to a suitable orientation - this was perfectly captured by our visualization. Fig 3 shows a screenshot of the Zeolite cage and methane at one time instant.

For the next example, we have a Fullerene tube of about 1750 Carbon atoms and a mixture of two fluids flowing through it. This was the first dataset with which we began this work. Fig 4 gives a view of the fullerene tube with fluid atoms inside it. Because of the number of atoms involved, standard molecular visualization packages were unable to provide the required visualization.

4. Visualization Tool

As we have mentioned before, we are using VRML for the visualization. The prime reason for this is that VRML is already the standard for 3D content on the web. This means numerous plugins exist to view VRML files; thus users would not have to use any new software to utilize MoDL. Another advantage of using VRML is that one can utilize the navigation controls that VRML plugins provide. These controls allow users to move about inside the molecular structure, zoom in on any part of the structure and view it from far. The user can also rotate his camera without changing his position. The controls also allow the

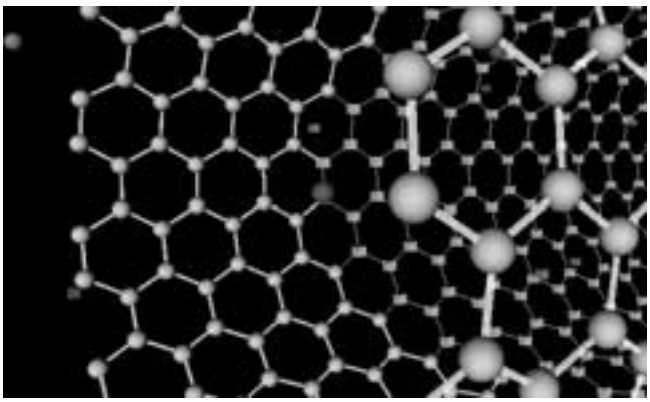

Figure 4. Nanotube

atoms/molecules to be rotated and viewed from any position and angle.

In most visualization packages atoms are represented by spheres and bonds are represented by cylinders. We have tried to optimize the rendering by using spheres to represent atoms only when they are near the user and by cubes when they are far away by using the LOD node of VRML. The distance after which the sphere is replaced by a cube depends on the radius of the atom, so that big atoms don't show up as cubes when close to the user. We have not used cylinders to represent bonds but triangular prisms. This way we are saving on the rendering since only 3 surfaces of the prism need to be rendered. We expect this will not jar the users since bonds don't have any physical existence and using cylinders to represent them was just a matter of convention. We have tried to optimize further by replacing the prisms by lines when the bonds are very far away from the user.

A dashboard of buttons is provided near the bottom of the screen, just above the VRML dashboard to allow the user to control the visualization (see Fig 2).

The following controls are provided to the user, in addition to any browser provided controls:

- Start Stop Step Flip
- Reset Viewer Original Zoom

'Start' and 'Stop' buttons will start and stop the simulation at any point. Here by simulation, we mean the visualization of the simulation; bear in mind that the actual simulation has already been run and we are working with data provided by the simulation program after it was completed. The 'Step' button makes the simulation proceed in steps; instead of continuous mode. This becomes very helpful when one wants to see a changing molecular structure after say, every 1 second during the simulation. The 'Reset' button resets the simulation clock to 0, ie. everything rewinds to the initial state. The 'Flip' button allows the user to flip the direction of time - ie. the simulation can run both forward and backward. Using this feature, users can rewind the simulation back to a certain point and replay.

Our visualization tool allows the user to change the viewpoint to the position of any atom/molecule and view everything from that atom/molecule's point of view. If this particular atom/molecule is dynamic, the user sits on it and views the static structure (if present) go past him. To do this, the user must first click the 'Viewer' button. Now all those atoms/molecules which can be viewpoints become clickable, ie. they can be selected by a mouse-click and the user will be transported to its position. Any atom/molecule can be made a viewpoint in the MoDL file when it is instantiated by assigning to the attribute *viewpoint* a value *true*. The 'Original' button is used to return the user camera to its initial position after having changed the viewpoint any number of times.

While working with a dataset which involved a large Zeolite cage and methane molecules getting trapped in it; we found that the large number of atoms and bonds of the Zeolite cage was cluttering the view, making it difficult to spot exactly where in the cage the methane molecules entered or left. This prompted us to include a Zoom/Occlude feature in the visualization tool. This will allow users to zoom to a particular dynamic atom/molecule; ie. only those objects which lie within 10 units of the selected atom/molecule will be displayed, the rest will not be shown. Users need to first click the 'Zoom' button and can then select any dynamic atom/molecule to zoom. This feature will make it easier for chemists to follow molecule trajectories inside complex structures.

5. The Conversion from MoDL to VRML

Here we describe very briefly, some particulars of the converter code and the VRML code which is produced by the converter. We have written the MoDL to VRML converter in Perl. We have used the Perl::XML module written by Larry Wall and Clark Cooper, which is a Perl interface over James Clark's expat library - an XML parser in C. The XML::Parser module is an **event-based** parser. This class of XML API's report parsing events (such as start and end of elements) directly to the application through callbacks. The application implements handlers to deal with the different events. This is in contrast to the class of XML API's known as **tree-based** in which the whole XML document is compiled into an internal tree structure and the application is allowed to traverse the tree. The main drawback of the latter scheme is that it often puts a great strain on system resources, especially if the document is large. Event-based parsers, on the other hand, provide a simple and flexible way of processing XML documents. Documents much larger than the available system memory can be parsed and required data structures can be constructed using the call-

back event handlers.

We have developed VRML prototypes for static and dynamic atoms, with viewpoints and without viewpoints. For every instantiation in the MoDL file, these prototypes are used to create new atoms, molecules and bonds. When a new atom or molecule type is defined in the **head** of the MoDL file, we emit prototypes for those types, which are used to declare atoms or molecules of those types in the **body**. The dynamics is implemented using the PositionInterpolator and OrientationInterpolator nodes of VRML[2]. For the simulation clock, we have used the TimeSensor node of VRML. This clock in VRML is continuous, we have wrapped code over it to make it discrete, so that the utility to step can be provided. The positioning of the user camera on viewpoint atoms and molecules is achieved using two ViewPoint nodes and javascripting. The Zoom/Occlude feature has also been implemented by javascripting. Whenever the position of a selected dynamic atom/molecule changes; its distance from every static atom/molecule is calculated, and the static atom/molecule is displayed or not accordingly.

6. Authoring tool for MoDL

Although, MoDL has been designed to be simple and flexible, creating large documents by hand can prove to be a cumbersome task. For chemical systems having few tens of atoms, it is possible to edit the MoDL representation using any text editor. But when the systems contain few hundreds to thousands of atoms and molecules along with dynamics, it is best to automate the process of generation of MoDL documents. We have created a tool, **MoDLEd**, the MoDL authoring cum editing tool. MoDLEd allows users to specify format of their data and generates a Perl script that converts data from that format to MoDL. The only constraints on the format is that each line has information for one atom/molecule and various values in a line (like x,y,z) are separated by space(s). Meta information about the types of atoms and bonds in a molecule can be specified in a separate file. Minimal editing facilities are also made available using the XML::DOM module. MoDLEd has been written in Perl with the GUI built using Tk.

7. Using MoDL and the Visualization Tool

The prime usage of MoDL and our visualization tool is to allow chemists to put their simulations on the web so that other people can study them across platforms by using **only** a web browser (eg: Netscape) and a VRML plugin (eg: CosmoPlayer). And putting up a visualization of the simulation instead of numerical data makes it more friendly in addition to the benefits of being able to visualize a simulation.

One way to do this would be to write or generate the MoDL file, validate it, convert it to VRML and put the VRML file on the web. Another way would be to create the VRML only when someone wants to view the simulation. To do this, a cgi-script is needed which will call the MoDL2VRML converter. On clicking a link, the modl file is read by this script - the vrml code is generated and sent to the client along with the appropriate mime type so that the VRML plugin is called up in the client browser. This is a clumsy method of transparently invoking a VRML browser from a MoDL file. However it is expected that two emerging standards will make this process much more cleaner. These are the X3D standard that is currently under development by the Web3D consortium and XSL (Extensible Style Language) under development by the World Wide Web Consortium.

8. Conclusions

The combination of XML and VRML can prove to be a very effective way of conveying chemical information over the Internet. Visualization of chemical simulations will get a boost once streaming is introduced in VRML. This will allow us to visualize a simulation when it is actually running in real-time. Other disciplines like Architecture and Planning can also benefit from the use of these technologies.

MoDL and the accompanying visualization can also be used as an instructional tool. While explaining a physical process, instructors can prepare a MoDL file from a simulation of that process and show the visualization to the students. This will allow the students to easily grasp ”what is going on”.

Our current work is focused on improving the functionality of MoDL as well as to address visualization problems in related areas. For instance, recently we have started work on visualizing shear in fluid suspensions as well as problems in crystallography.

9. Acknowledgements

The authors wish to thank Prof. Yashonath, Prof. G. Ayappa and their group for providing us the driving problem and the data sets, and for enthusiastically using our tools.

References

[1] Tim Bray, et. al. XML 1.0 Specification.

[2] W. Ihlenfeldt. Visualizing Chemical Data in the Internet - Data Driven and Interactive Graphics.*Computers and Graphics*,22(6):703-714,1999.

[3] International Standard ISO/IEC Virtual Reality Modelling Language, 1997.

[4] Peter Murray Rust. Chemical Markup Language.
[5] Jon Bosak, Tim Bray, XML and the Second-Generation Web. *Scientific American*, May 1999
[6] Daniel Lipkin. Integrating XML and VRML: A Technical Discussion.

0-7695-0643-7/00 $10.00 © 2000 IEEE