21 Visualization and Real-Time Collaboration over Internet-2

J. Adler, A. Hashibon, and G. Wagner

Department of Physics, Technion, Haifa 32000, Israel

Abstract. We report on recent efforts in our research group to visualize data sets resulting from atomistic simulations, and to communicate these visualizations over high-bandwidth internet connections.

21.1 Why Visualize?

Visualization is essential for both development and presentation of atomistic simulations, as well as for program debugging, especially when complicated solid structures and boundary conditions come into play. Results of a simulation may consist of large sets of numbers – for instance, the positions of simulated atoms at a given stage. These may be visualized as a still picture, often as an animated picture, and even as a three-dimensional animated picture involving three-dimensional renderings. Once the interim and final states are visualized one can better understand the system and know which measurements to make.

Atomistic simulation/visualization is complementary to experimental observation because in many laboratory systems it is not possible to look nondestructively "within" the sample. For example, diamonds need high pressure to remain diamond and not turn to graphite. Thus it is not possible to cut samples open to view the atomic structure. We have a long series of collaborations [1,2] on diamond/graphite where the simulation visualizations confirm hypotheses deduced from experiment and also provide surprises (for example [3], where by watching the path of a hydrogen interstitial we found a new interstitial site!) Although in systems such as metal-ceramic interfaces one can visualize directly under an electron microscope this is sometimes possible only for quenched samples and may not enable study of dynamics or of internal details of structures near melting temperatures.

21.2 Why Visualize on Internet-2?

In our group all calculations are motivated from laboratory experiment. We currently collaborate with experimentalists in physics, chemistry and materials science departments. For our experimental collaborators the visualizations provide a common language – pictures transcend notational and conceptual

differences. However, we cannot expect our experimental collaborators to have powerful graphical workstations, and we also want to interact with researchers on different continents. Therefore we must be able to show results over the internet in a form that can be easily displayed on any computer. Thus we must use a visualization package that is easy to project over the internet, and develop means to project it.

With the advent of high-bandwidth internet connections ("internet-2"), it has become feasible to transmit sequences of high-resolution images across the internet, at a frequency of 1 s per image or better. If rendered in a remote web browser, these images can emulate a visualization session running locally and generating output "on the fly". This approach has the advantage that the receiver of the transmission is not required to run any particular software apart from a web browser.

21.3 Our Atomistic Simulations

Our recent projects relate to either carbon in various forms or to metal-ceramic interfaces. The carbon project [1–4] has involved D. Saada, I. Rosenblum and A. Sorkin from our group and S. Brandon from Chemical Engineering on the computational side and R. Kalish, A. Hoffmann and S. Prawer on the experimental side. The computational techniques here are classical molecular dynamics (MD) (Tersoff and Brenner potentials), tightbinding MD and ab-initio. Monte Carlo with the same potentials is also done as needed. Studies on aluminium/alumina and related systems have/are being made by Adham Hashibon and Geri Wagner from our group and Wayne Kaplan and students/postdocs on the experimental side.

Details of the physical results are published elsewhere; in the present paper we concentrate describing our solutions to the visualization requirements for these and similar atomistic simulations.

21.4 Requirements and Inventory

We require color, three-dimensionional rendering, (implying capabilities of good shading), the possibility of drawing several thousand atoms at once, animation facilities, and the ability to highlight specific bonds and configurations at will to convey information.

As presented at the 1998 workshop [5], we need an interactive graphics system which is cheap both in terms of software costs and hardware support required to run on each students desktop. Given the requirements listed above and cost issues, there is no ready-made solution that we have ever found.

Each member of our group has a powerful workstation (a.k.a. LINUX box with the requisite public domain software) on their desktop. We have been using MESA for many years on LINUX boxes with or without hardware acceleration. MESA is a public-domain clone of the industry-standard OpenGL

graphics library. MESA/OpenGL is built on-top of the X window system and allows for three-dimensional rendering of objects of arbitrary shape, and is clearly a well-suited software base for the needs at hand. Our old routines still basically work, but the interface is cumbersome and has reached a stage where none of the newer libraries or compilers were compatible. (We used the glaux libraries for our graphics interfaces and these are no longer supported.) The old routines were based on transmission of X-graphics; however firewalls and other necessary protections mean that direct projection of X-graphics is unreliable (works today, is blocked tomorrow). Lifting these precautions opens your system to hackers.

Thus we need to find ways to modernise our interface and communicate our visualizations while retaining MESA/OpenGL as the basic tool and LINUX as the operating system.

21.5 Implementing Visualization

Instead of applying modifications and patches we developed a new approach, based on newer software but according to the above philosophy. This approach involves MESA/OpenGL graphics, embedded in a graphical user interface that employs the Qt library, a freely-available library to build non-profit applications under X (`http://www.trolltech.com`). The Qt library is a toolkit for programming in X, similar to the popular Xt and Motif toolkits. It provides several short-cuts to build a graphical user interface, thus simplifying the programmer's task to some extent, at the price of slightly reduced transparency of the code. We chose to use Qt for the following reasons: (i) Qt provides a simple way to implement MESA/OpenGL graphics. (ii) Qt is becoming increasingly popular in the LINUX community and is likely to be around for some years to come. (iii) Qt has been implemented for several different operating systems, including LINUX (our system of choice) and MS-DOS Windows. It is able to make use of hardware graphics acceleration. (iv) Qt is freely available and is in fact part of many modern LINUX distributions.

The main concern in implementing the graphics is speed of operation. A beautiful interface with bells and whistles is useless if it takes for ever to render a given data set. Unfortunately, the choice of OpenGL as the underlying graphics engine (as opposed to rendering directly in X) limits the rendering speed. The main measure that may be taken to speed up rendering is to simplify the task at hand. In atomistic simulations, one intuitively envisages the simulated atoms as spheres. Spheres require a lot of time to render, depending on the rendering quality (they are approximated by several triangular faces). Rendering not spheres but cubes, or even dots, increases the operating speed considerably. For this reason, users must have the option to choose the render style (spheres, cubes, dots). The users can then study their data sets quickly using low-quality rendering, and switch to higher quality later on.

We are currently experimenting with rendering modes that offer high quality for near-distance atoms combined with low-quality rendering for remaining atoms.

Another important aspect is the ability to select specific domains for the visualization. It is always possible to render, say, a cross-section of a set of atoms by preparing data sets representing only the desired domain, but this is not convenient. Users should be able to set interactively boundaries in the x-, y-, and z-direction that define a domain to be rendered.

We call our visualization software AViz (Atomic Vizualization) and will make sources and instructions freely available on our research group website (`http://phycomp.technion.ac.il`), once we have completed the initial part of development. (Some work on shading and adding bonds still needs to be done to make the quality equivalent to our old routines.) Examples of source files and an overview of the current state of development can be found on our website [8].

21.6 Implementing Communication

Under X, the popular window system used by most computers running UNIX, the network connection may also be used to serve the display of remote machines directly. A visualization package may then run on a powerful local machine, while its ouput is directed to the client machine. In addition to problems with firewalls, X client-server applications work only if the client, i.e. the collaborator abroad, is running X as well (as opposed to MS-DOS Windows and other operating systems). In contrast, internet browsers provide security and protection, yet enable easy access to graphics regardless of the particular operating system that is employed.

For the communication of the visualization output, we therefore searched for cost-effective means of projecting an uninterrupted sequence of pictures onto the internet. When a browser is instructed to render the contents of a conventional web site, it communicates with the server machine, reads the content of the site, and then renders while closing the server connection. In contrast, we require the connection to be held open during the entire session, and the rendering to be updated continuously. This can be achieved using a technique known as "server-push", supported by all modern web browsers.

Information on how to implement a server-push application is available abundantly on the internet. The implementation can be achieved essentially at zero costs, involving only Perl scripts (there exist public-domain Perl scripting language interpreters). Finally, to transmit the server-pushed data, a web server software must be run on a computer that can access the visualization output that is generated during the session; most conveniently the one on which the visualization software is running. One of the most popular web servers, the Apache software, is freely-available and readily installed.

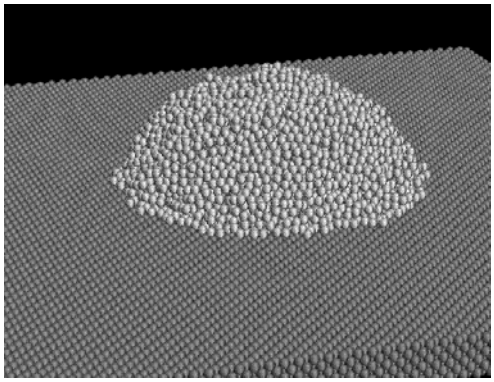

Fig. 21.1. Rendering of a MD simulation of the spreading of an aluminum drop on a metal-oxide surface

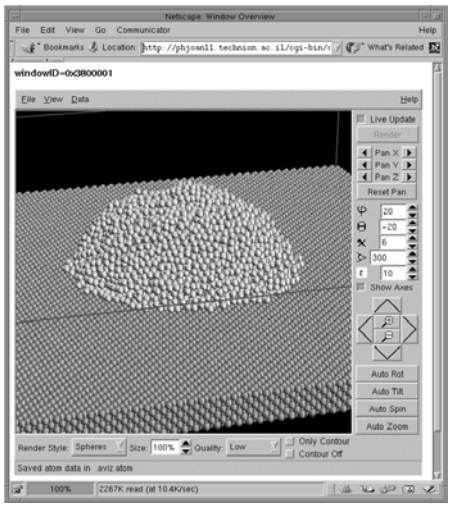

Fig. 21.2. Projection of the AViz interface including the rendering shown in Fig. 21.1 into the web browser

We have set up a web server that allows anyone with access to a browser to view any window on the screen of the server-push machine. In combination with the visualization software running on the machine, anybody can effectively study the visualization output over the internet, with no degradation of image quality. This is achieved by means of short Perl scripts, in combination with system commands. The main script forms essentially a loop in which a screen dump is performed and translated into an image file format, and then sent out onto the internet [6]. As the visualization software is operated, the web output is adjusted "on the fly" since the loop is iterated.

21.7 Examples

We illustrate with two examples of our graphics. One is the rendering of an MD simulation of the spreading of an aluminium drop on a metal-oxide surface (the data is taken from the PhD project of A. Hashibon, see also [7]) and the other is a picture of the full graphical interface. More examples can be viewed on the push server `phjoan11.technion.ac.il`, or in the web pages about our talk at the workshop [8].

Acknowledgements

We thank our experimental collaborators – R. Kalish, W. Kaplan, A. Hoffmann, and E. Polturak, and acknowledge that this work is based in part on earlier projects together with David Saada. Supported by the Israel Ministry of Science, US-Israel BSF and GIF.

References

1. D. Saada, J. Adler, and R. Kalish: Phys. Rev. B **59**, 6650 (1999)
2. I. Rosenblum, J. Adler, and S. Brandon: Comp. Mat. Sci. **12**, 9 (1998)
3. D. Saada, J. Adler, and R. Kalish: Phys. Rev. B **61**, 10711 (2000)
4. I. Rosenblum, J. Adler, S. Brandon, and A. Hoffman: Phys. Rev. B **62**, 2920 (2000)
5. J. Adler, A. Hashibon, A. Kanigel, I. Rosenblum, and D. Saada: *Visualization for molecular dynamics in solids*. In: *Computer Simulation Studies in Condensed Matter Physics, XI*, ed. by D.P Landau and B. Schüttler (Springer, Heidelberg Berlin New York 1999) pp. 186–189
6. Projecting a piece of data such as an image file onto the internet is straightforward by means of so-called cgi-scripts. These are programs that are activated by the web server software; their output is sent onto the internet just as the content of a static web site may be sent by the server. Listings of these short scripts are provided on our experimental web server.
7. A. Hashibon, J. Adler, M. Finnis, and W. D. Kaplan: "Atomistic study of structural correlations at a solid/liquid interface", submitted
8. `http://phycomp.technion.ac.il/aviz`