Submitted to a special issue of *Concurrency: Practice and Experience* following the *Java for Computational Science and Engineering Workshop* held at NPAC on the $16/17^{th}$ December 1996.

Submitted: $5^{th}$ February 1997.

# Java simulations for physics education

Simeon Warner[1], Simon Catterall, Edward Lipson

Department of Physics, Syracuse University, Syracuse, NY 13244-1130, USA

We discuss the use of World Wide Web-based Java simulations in teaching physics to K–12 and undergraduate students. Our work focuses on the physics of membranes and illustrating how such systems are studied. We propose that Java should be used not only to produce small versions of research simulations but also to provide models illustrating simpler concepts underlying the main subject matter. In particular, applets should be tailored to the context in which they appear and should be as intuitive to use as possible. The applets we are developing are described in the context of current client performance. We also highlight the development of collaborative systems as an area of particular interest.

## Introduction

Current simulations in many areas of physics and engineering require high-performance computers and long run times, which tend to make the subjects inaccessible to students. In collaboration with the Department of Physics and the College of Engineering at Cornell University; and the College of Engineering and Computer Science and the Northeast Parallel Architectures Center (NPAC) at Syracuse University, we are developing a set of interactive educational modules based on large simulations. Four modules are being developed: fluctuating membranes; fluid dynamics;

[1]Corresponding author; simeon@physics.syr.edu

crack propagation and structural failure; and avalanches. In this paper, we concentrate on our work in membrane physics and quantum gravity [1,2], from which we will produce the fluctuating membranes module.

The educational modules are to be integrated into undergraduate courses, and then into K–12 (ages 4 to 18) curricula. Material is being developed such that it can be used for individual study and in classroom situations. We intend to use the best available technologies. Currently, these include the World Wide Web (Web) and Java, but they may soon include Virtual Reality Modeling Language (VRML). Web technologies are being developed at an extremely rapid pace, so we are continually evaluating new technologies as they emerge.

Before Java became available, the entry-level general-science course 'Science for the 21st Century' (offered by the Department of Physics at Syracuse University [3]) used CGI (Common Gateway Interface [4]) scripts and HTML (Hypertext Markup Language [5]) forms to provide interactive Web access to neural-network models [6]. These were enthusiastically received by students and have proved a useful stepping stone to the development of Java applets in the current work. We are also drawing on NPAC's experience in developing Web material for K–12 students in the Living Schoolbook Project [7].

We discuss below our experiences writing simulations in Java, and preliminary studies in building educational modules. To date, evaluation has been by peers alone, and our modules are aimed at the undergraduate level.

## Membrane physics

The study of two-dimensional interfaces and membranes to help understand their properties is an important area of research for many fields of science, including biology (cell membranes), chemistry and chemical physics (reactions at interfaces, cosmetics, medicine), and theoretical physics (condensed matter, particle physics). It is remarkable that the surface behavior of red blood cells in the body can be described by theories very similar to the string theories of particle physics. We aim to produce modules that will bring out the generality of membrane models. They will provide a range of explorations from the biology of life processes through materials to cosmology. We hope to leverage interest in the abstract physics from the more accessible biological contexts.

Simulations of real membranes are designed to probe the underlying microscopic dynamics. In particular, the study of artificial membranes composed of amphiphilic phospholipids (see, for example, [8]) is a lively research field with applications in industry and medicine. For example, such membranes are able to close in on themselves forming spherical vesicles, and may be tailored to open up and release their load when the 'correct' physical and chemical conditions are found. In this way, they have potential as drug carriers [9].

In the area of particle physics, string theories can be thought of as theories of random surfaces. Thermal fluctuations in the biological context are then replaced by quantum fluctuations. To illustrate concepts in particle physics, we hope to draw on the common thread with real membranes. However, it is hard to see how we can explain any meaningful particle physics, other than to the high-end of K–12. At present, we are concentrating on real membranes and the biological context in particular.

We consider two classes of membrane: fluid membranes, where the constituent molecules can freely flow around each other for any shape of the membrane surface, and crystalline or polymerized membranes, where the molecules are held in place by strong covalent bonds. At any finite temperature, these membranes undergo thermal fluctuations and the physical properties of the surface depend on the interplay between these disordering effects and the ordering associated with a curvature-suppressing energy term. Complex phase structures exist between 'smooth' surfaces, 'crumpled' surfaces, and other more exotic arrangements.

## From applets to teaching tools

Our suite of simulations will illustrate a variety of phenomena, including the 'flickering' of red blood cells, the 'crumpling' transition in polymerized membranes, and string theory. We are using Java, supported by text, still images, video, and interactive simulation-on-demand. Java simulations of small systems provide the most interactive experience for the student. However, Java is currently comparatively slow, and client systems are not up to supercomputing standards. Thus we will also use simulation-on-demand for larger systems. Even here, Java is an excellent tool for building client-side user interfaces.

The term 'edutainment' has been coined to describe the idea of adding entertainment to an education application to enhance the learning experience. In essence, the old idea of making learning fun — taking medicine with sugar. However, we must recognize just how challenging it may be to entertain; students are unimpressed by last year's video games. It is a tall order to produce physics education material that will be considered 'cool' and thus interesting. While we cannot compete with the latest video games on presentation, we must not become complacent and assume content will be enough.

In spite of the need to be 'cool', we think simplicity is very important. Most students will use each applet for only a short time, and so time spent learning to use the applet is largely time wasted. We are not trying to teach people how to 'drive' an applet, we are trying to teach physics. So we want to leverage as much as possible from the students' experience with other computer interfaces (games, for example). Can we make the applets seem intuitive? If we can, then the learning experience will be less stressful and hopefully more productive. It is thus important not to fall into the 'gizmo trap.' Functionality should be added only where it doesn't make the basic functions more difficult to use. There can always be a second version with more complex features. Also important is screen real estate; adding redundant functionality often wastes this precious resource.

## Experience with simulation applets

We have found Java easy to use, and the code is very similar to our research simulations written in C and C++. This has actually worked in reverse now; our latest C++ simulation is based on the structure of the Java code.

*** Figure 1 here ***

Figure 1: Snapshot of a simple Java applet running a Monte Carlo simulation of an 8×8 crystalline 2D membrane.

The applet shown in Fig. 1 allows the viewpoint to be arbitrarily rotated as the membrane evolves under Monte Carlo updates. It uses the OOGL_OFF class written by Daeron Meyer of The Geometry Center [10]. This class has been extended with one new constructor and an update method to do data conversion. In this way, we were able to add display and rotation with very little effort. This applet provided us with useful proof of concept, though the speed of execution limits us to rather small grids (16×16 becomes quite slow), mainly because of graphics speed. However, this varies widely and is improving on all platforms as new implementations become available.

VRML is an obvious choice for 3D displays. We have made static VRML worlds from membrane examples with promising results. Although the VRML viewer we used (Silicon Graphics Inc. Cosmo 1.0 on Irix) could render very much faster and better than our Java code, a 33×33 mesh was still unacceptably slow on an Silicon Graphics Inc. Indy workstation. Typical research simulations are 64 or 128 nodes square.

VRML alone is not enough, as we need to have text for explanation and instructions. New products integrating VRML and Java may provide good solutions to this. The ideal scenario would be the ability to create a VRML component within a Java applet. Alternatively, we could pop up a separate VRML window dynamically generated from the Java simulation.

The fluid membrane simulation shown in Fig. 2 allows both node moves as in the crystalline case,

*** Figure 2 here ***

Figure 2: Snapshot of a Java applet running a Monte Carlo simulation of a 2D fluid membrane. One link is highlighted to illustrate the link flip that is taking place.

and also geometry changes (link flips). The current implementation optionally highlights link flips to help illustrate the simulation process. In this case, the membrane has spherical topology and the speed limiting factor is the image rendering.

The first impression of dynamics in Monte Carlo simulations is good, but it is rather misleading. Monte Carlo dynamics are not real and the individual configurations are not as meaningful as they appear. Instead, one must look at statistics from a representative ensemble of equilibrium configurations. Fig. 2 shows an inset graph of the internal energy of the surface, we are adding simple analysis routines to allow students to calculate mean values and autocorrelation times. Further, for one to extract meaningful data, the system must thermalize for any given set of parameters. This can be particularly problematic around phase transitions, an area of particular interest, where Monte Carlo simulations often take much longer to reach equilibrium. At present we have a button that sets the simulation running without updating the display (hence much faster) for a predetermined number of updates.

*** Figure 3 here ***

Figure 3: Snapshot of a supporting Java applet illustrating the properties of a simple spring. The user can move the end of the spring by dragging the block or one of the graph tracers to plot the force-extension and energy-extension graphs for a simple spring obeying Hooke's Law.

In addition to our headline simulations, we intend to develop supporting applets to illustrate principles behind the simulations. For example, some of the membrane simulations are based on ball-and-spring triangulations. We thus have an applet to help students understand springs (Fig. 3). This can then be tailored, along with supporting text, to various levels of understanding. For the younger students, just the idea of increasing force will suffice, whereas we hope to illustrate the importance of stored energy to more advanced students. We have purposefully made the interface very simple with buttons that clear the graphs, draw in the whole graphs, and offer help (including explanation of the other two buttons). The help information appears in a pop-up window.

## Speed

Java has made client-side computation a commonplace reality. In a classroom or many-user situation, there is clearly more power on desktops than on the server. This is especially important if we consider large groups where this scalability would be particularly important. If just the Java byte codes are downloaded, as opposed to movies or server-push animation from CGI simulation, then there will be a drastic reduction in server hits and data transfer. Thus client-side computation is essential with many slow links. Java class files are compact, typically only a few kilobytes long.

We tune the size and complexity of our Java models to the expected client performance. As technology progresses, we will be able to handle large and more interesting problems. In particular, there will be a significant performance improvement if just-in-time (JIT) compilers become widely used.

## Collaborative systems

There is obvious application of collaborative technologies to classroom situations. One simple scenario might have the teacher controlling all displays and passing control to a chosen student when required. For this, one requires applications that can act as either master or slave, and some method of token passing to establish the master at any given time. Statistical studies like Monte Carlo are ideal candidates for slightly more sophisticated collaborative systems involving combined data collection from a classroom of workstations. Imagine many groups running simulations in isolation, collecting data, and then performing statistical analysis. Each group could then submit data to a shared analysis package and perform analysis on data from the whole class. This would provide a wonderful illustration of the benefits of more data.

NPAC scientists are developing a collaborative system called 'Tango' [11] which is integrated with a Web interface. The NPAC team has already demonstrated the ease of integrating existing applets into their collaborative framework by porting two simple physics applets into Tango.

## Concluding remarks

It is clear that 'cool applets' alone will not solve the problem of developing well thought out educational material. However, interactive simulations will add significant value to educational material, and hopefully stimulate the students' interest in subjects such as physics which many consider uninteresting or arcane.

Java appears to be the best technology available now. It is not yet clear how Java will develop in the next couple of years, though we can be sure that it will change significantly. While the development of Java is primarily driven by the needs of commercial Internet users, it seems reasonably equipped for use in scientific contexts. The issues of speed, display technology (VRML/Java rendering), and stability are our primary concerns. We see collaborative tools as a particularly exciting development that will add another level of interaction to educational tools.

## Acknowledgements

The applets described in this paper were written, or contributed to, by Eric Gregory, Metin Sezgin and Roberto Salgado. This work has been supported by NSF Grant ASC-9523481.

## References


1. S. Catterall, G. Thorliefsson, J. Kogut and R. Renken, "Singular vertices and the triangulation space of the D-sphere", *Nuclear Physics* **B468** 263 (1996)

2. M. Bowick, S. Catterall, M. Falcioni, G. Thorliefsson and K. Anagnostopoulos, "The flat phase of crystalline membranes", *Journal de Physique I*, France 1321 (1996)

3. S. Catterall, M. Goldberg, E. Lipson, A. Middleton and G. Vidali, "Implementation of information technologies in the teaching of 'Science for the $21^{st}$ Century' ", *International Journal of Modern Physics C*, in press

4. CGI specifications are available from the WWW Consortium (W3C) site:
`http://www.w3.org/pub/WWW/CGI/`

5. The specifications for HTML forms have remained largely unchanged since HTML 1.0. The current specification is HTML 3.2. HTML specifications are available from the WWW Consortium (W3C) site: `http://www.w3.org/pub/WWW/MarkUp`

6. Neural-network models are part of the Mind and Machine module offered by the Department of Physics at Syracuse University: `http://www.phy.syr.edu/courses/modules/MM/index.html`

7. K. Mills, G. Fox, P. Coddington, B. Mihalas, M. Podgorny, B. Shelly, and S. Bossert, "The Living Textbook and the K–12 Classroom of the Future", *Proceedings of the 1995 International Symposium on Supercomputing*, ACM (1995);
online version: `http://www.npac.syr.edu/projects/ltb/SC95/index.html`.
Living Schoolbook (previously Living Textbook) site: `http://www.npac.syr.edu/projects/ltb/`

8. G. Gomper, and M. Schick "Self-assembling amphiphilic systems", in volume 16 of *Phase Transitions and Critical Phenomena*, Academic Press (1994)

9. See, for example: "Liposome Technology : Volume III : Targetted Drug Delivery and Biological

Interaction", edited by G Gregoriadis, CRC Press Inc., Florida (1983)

10. The OOGL_OFF class was extracted from Daeron Meyer's 3D viewer applet. This and other graphics packages are available from the Geometry Centre at the University of Minnesota: http://www.geom.umn.edu/~daeron/apps/

11. Tango Web based collaborative system: http://www.npac.syr.edu/tango/