

Using a Dataflow Language and the World Wide Web for Scientific Visualization

HELEN M. DOERR^{1,*} and BRUCE R. LAND²

¹Mathematics and Mathematics Education, 215 Carnegie Hall, Syracuse University, Syracuse, New York 13244-1150, and ²Cornell Theory Center, 606 Frank H. T. Rhodes Hall, Cornell University, Ithaca, New York 14853; e-mail: hmdoerr@sued.syr.edu and bruce@cornell.edu

ABSTRACT

The visual programming language Data Explorer has been a powerful tool for teaching and learning introductory computer graphics. The World Wide Web was introduced as a mechanism for providing students with lab exercises, sample programs, and related course material, and as a site for the publication of their final design projects. © 1996 John Wiley & Sons, Inc.

INTRODUCTION

The past decade of technological advances in high-performance computing and communications has brought dramatic improvements simultaneously in the infrastructure for accessing networks and in the performance of desktop workstations. Within the past 2 years, the advent of the supporting protocols for the World Wide Web (WWW) and browsers such as NCSA Mosaic and Lynx have resulted in an explosion of resources on the national information superhighway. At the same time, these tools now offer new opportunities for undergraduate science and mathematics education in a collaborative, networked environment. These tools are beginning to change not only what we teach, but how we teach, and they suggest new roles and new opportunities for both students and teachers within an expanded classroom [1]. In this article, we describe how this has happened in an interdisciplinary, undergraduate

course in computer graphics being taught at Cornell University.

For the past 2 years, an undergraduate course in computer graphics with an associated lab course has provided students with an introduction to the principles of interactive computer graphics and scientific visualization. More recently, we have created a Web site for the course to serve as an on-line resource for the students to review lab procedures and results. From the first day of the course, the students have access to all the information about the tasks that will be done throughout the course. More important, the final design projects (described below) from the previous offerings of the course are now available to the students and provide ideas and building blocks for their projects. Thus, the accomplishments of the students from each prior semester become part of the knowledge created through the course and an available resource for the next semester. In addition, the Web site provides linkages to the Cornell Theory Center's Data Explorer repository and on-line tutorial system (at <http://www.tc.cornell.edu:80/DX/>). These modules and tutorials become part of an expanded set of classroom resources for the students, providing

* To whom correspondence should be addressed.

alternative paths for exploring the material covered by the class instructor.

Scientific visualization is a way of organizing numerical results in a form which is accessible to humans. The human visual system is extraordinary in its ability to understand complex images. Visualization uses this innate ability to find patterns in data of all kinds, from satellite imagery to quantum field simulations. As visualization techniques become simpler and more accessible, applications abound in traditional engineering and science courses. This course is a first step in making visualization techniques available to undergraduates. The students who take this course are primarily upper-division science and engineering majors. It should be possible to construct a freshman-level course using DX which introduces computer graphics and visualization at an appropriate mathematical level and it would whet student appetites for further instruction. About half the students are from computer science, the department that gives the course. The rest are distributed among electrical and mechanical engineering, physics, mathematics, chemistry, and architecture. This diversity in student majors is reflected in the final design projects developed by the students. By way of example, Figure 1 illustrates the projectile motion of a basketball being thrown by a robot. Over the last 2 years, at least a fifth of the students taking the course either have gone on to work in research labs using scientific visualization or have followed up with independent research projects. Examples of these projects can be found on-line at <http://www.tc.cornell.edu/Visualization/>, and then following any of the three years worth of CS490 links in the Education section. These projects serve as useful documentation for anybody attempting to learn the functionality of DX.

GRAPHICS TECHNIQUES FOR SCIENTIFIC VISUALIZATION

Scientific visualization is a topic that requires mathematical, programming, and artistic skills, along with content knowledge in a specific domain of science or engineering. This introductory computer graphics course focuses on the mathematical skills, but also includes both programming to illuminate the mathematics and applications in science and engineering. The course, described more fully elsewhere [2,3], covers the following topics:

- Construction of surfaces by explicit polygon

lists, by parametric operations, and by hierarchic grouping of simple objects to form complex surfaces

- Modification of objects by three-dimensional geometric transforms to animate their motion
- Viewing of a group of objects including clipping to a window and perspective or parallel projection into a two-dimensional screen space
- Rendering of polygonal and parametric surfaces by hidden surface removal, by shading/lighting, by antialiasing, as well as by surface property modifications such as texture- or bump-mapping
- Use of global illumination models to render interactions between surfaces
- Modeling of scientific data for scientific visualization.

In addition to traditional lectures using the work of Foley and van Dam [4] and Watt [5], the students engage in a range of practical, hands-on experiences. Each student completes eight lab exercises and a final design project of his or her own choosing. Each lab exercise contains both sample programs to introduce the concepts and related descriptive material about the manipulations required of the student. All work is based on Web documents which explain the lab procedure, show examples, and supply source code. Currently the exercises cover basic computer graphic rendering and introductory scientific visualization. The first exercise shows the student how to specify a polyhedron by specifying the vertices and the polygonal faces of polyhedra. The students are expected to design a box and a simple propeller and animate the box opening and the propeller spinning. Specifying every vertex in a complex scene is impractical, so the second exercise illustrates how to use parametric operations to produce complex objects (e.g. torus or horn) by mathematical operations on a flat sheet. The operations may generate quadric surfaces, surfaces of revolution, or translation surfaces. The students are expected to construct a beverage bottle mathematically and to animate a spring/mass system. To increase scene complexity further, the third exercise demonstrates how to combine predefined, simple objects (polyhedra or parametric surfaces) into hierarchic objects. An example might be a robot with fingers made of cylinders, attached to an arm made of blocks, attached to a body which again is a cylinder (Fig. 1). The students must build one of several objects and animate it. This year the list included a

helicopter, a wagon, a robot, or a bird. Each of these objects had a required set of motions it had to perform.

By this point in the course the students can build very complex scenes, so the emphasis turns from building objects to the process of converting those objects into pictures on a computer screen. The fourth exercise shows the students how to move a computer-graphic "virtual camera" around in the scene, just as a real videotape recorder must be moved through a room. They are introduced to the concepts of camera motion and of using a camera to produce an image which itself becomes part of the final picture. The students must produce a TV camera and TV monitor. The TV monitor displays what the TV camera is looking at. Both must be visible in the final picture. The fifth exercise continues on the camera theme by requiring the students to perform the mathematical transformations to construct a perspective camera. The exercises then turn to computer graphic lighting and surface texturing techniques. In the sixth exercise, the students learn how to construct Phong illumination, allowing them to construct "spotlights." They must program a spotlight and fly it through a simple scene. The seventh exercise shows them how to map a picture onto a surface, how to apply a texture (e.g., "marble") to a surface, and how to make a smooth surface appear rough. These techniques increase the realism of a scene without increasing the complexity of the underlying geometric objects.

Now the students can build complex objects and view them in a way similar to videographic techniques. The eighth exercise uses these techniques on objects derived from data as an introduction to scientific visualization. Rather than teach the students a scientific discipline to motivate scientific visualization, we chose to use landscape generation as a "data set" to be visualized. First, the students were shown how to generate and visualize a fractal landscape. The gradient of the landscape altitude was calculated as a vector field used to generate "streamlines" which became streams on the landscape. Clouds were generated by a three-dimensional fractal density field and visualized in two different ways: by treating the density as numerical haze and by fitting polygons to a certain density level. The students had to add "rapids" to the rivers by coloring them white where the slope was high, add trees with positions based on altitude, and animate the cloud so that the density changed with time.

Armed with the techniques taught in the first eight exercises, the final exercise asked the students

to design an animation of their choice. There were only a few constraints: The animation had to use a selection of techniques taught in the course. The animation had to have a title frame which included the authors' names and a copyright notice. The animation had to use MPEG compression for final storage. Students worked in groups of two and each person was expected to put in about 25 hours on the project. The resulting animations show a huge diversity. Some were artistic, others were scientific. The complexity and sophistication of many of the projects can be amazing. The Web location <http://www.tc.cornell.edu/Visualization/contrib/cs418-sp94/cs418.html> has several examples of student work. Figures 2–4 illustrate some of these examples.

At the introductory level, the lack of programming tools often interferes with learning graphics operations. Typically, graphical operations either are provided as a large library of routines that students must learn, or derive from relatively simple programming projects that students must code from scratch. DX is a block diagram, point-and-click programming language designed for producing high-quality images based on three-dimensional data which may represent objects (walls or stars) or fields (density or electric field). The block diagram program is built of "modules" and "wires" connecting them. A module is a primitive program function that appears on the screen as a block with input and output tabs. Figures 5 and 6 show two DX program examples.

Wires are dragged with the mouse from outputs to inputs. DX modules include vector and scalar field rendering, and geometric operations (rotate, translate), in addition to x and y plotting. The block diagram interface of DX can be customized to make a "visual subroutine library" of often-used graphics routines. Students can construct very complex scenes including camera and lighting control in a short time. They can also "open up" various visual subroutines and modify or extend them. The graphical nature of the user interface and its easy extensibility make it possible for students to rapidly prototype a graphics operation, and to see the effects and modify the program. The Cornell Theory Center has added modules to make parametric surfaces, texture maps, bump maps, spotlights, and a variety of other graphically oriented functions.

EVALUATION AND FEEDBACK FROM STUDENTS

In the spring 1993, we made a videotape of the student final design projects. This videotape was

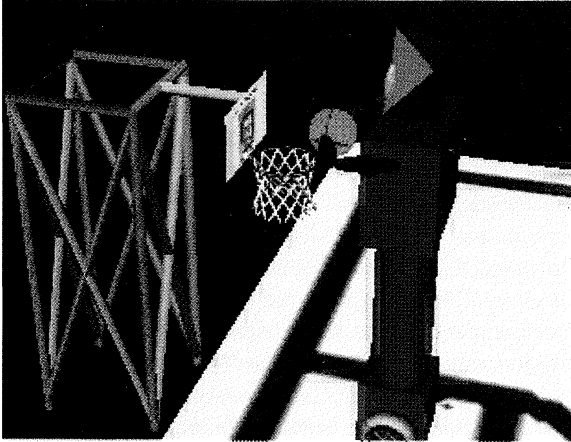


Figure 1 A robot shooting a basketball. For the animation, the trajectory of the basketball was iteratively solved using a symbolic algebra package. The trajectory was then used to control the movement of the robot so that it looked like the robot was producing the motion. The net is a parametric surface showing only the edges of the polygons making it up.

shown to the students in the next offering of the course and provided positive motivation for those students. In the spring 1994, posting of student animations to the internet became possible with the

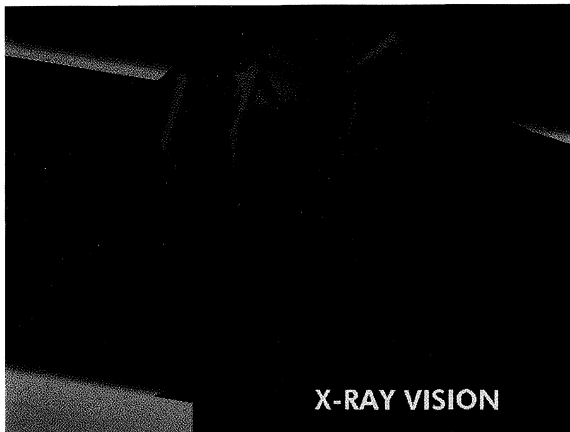


Figure 2 The title of the animation was “It’s not a bug it’s a spider.” The students constructed a simplified “spider” using geometric primitives in DX. The spider geometry (vertex list) was then exported to a program which solved Laplace’s equation on a square region, with boundary conditions of zero height, and with the additional constraints that at no place could the relaxation process move the solution below the highest point on the spider. The result is what appears to be a rubber sheet stretched over a spider. The positions of the sheet were imported back into DX and rendered along with the spider. Each frame of the animation required a separate solution.

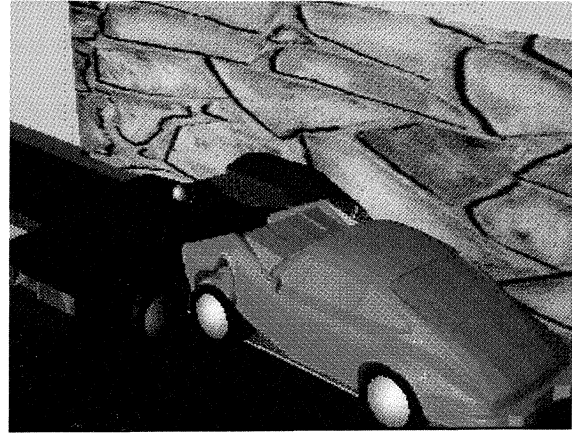


Figure 3 The title of the animation was “Fasten Your Seatbelts!!!” The students found geometric models of cars at an internet site, downloaded them, and converted them to DX object format. They figured out a parametric distortion of the geometry which approximated crumpling of the front of the cars. They then animated the cars and applied the distortion functions at the appropriate times (when the cars hit a wall or each other).

advent of the Web, Mosaic, and MPEG compression. Publication and distribution via the Web appear to have the potential for an even larger motivation effect. Moreover, students can now sign up for

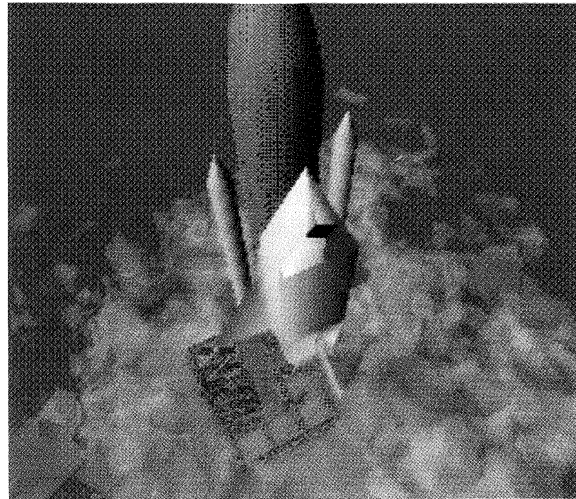


Figure 4 The title of the animation was “Space Shuttle.” The students modeled a space shuttle by explicitly designing a vertex list describing a polygonal approximation to a real shuttle. They modeled the fuel tanks and boosters by parametric operations. The smoke was a fractally deformed density field which was converted to an isosurface of constant density, then scaled flat along the ground.

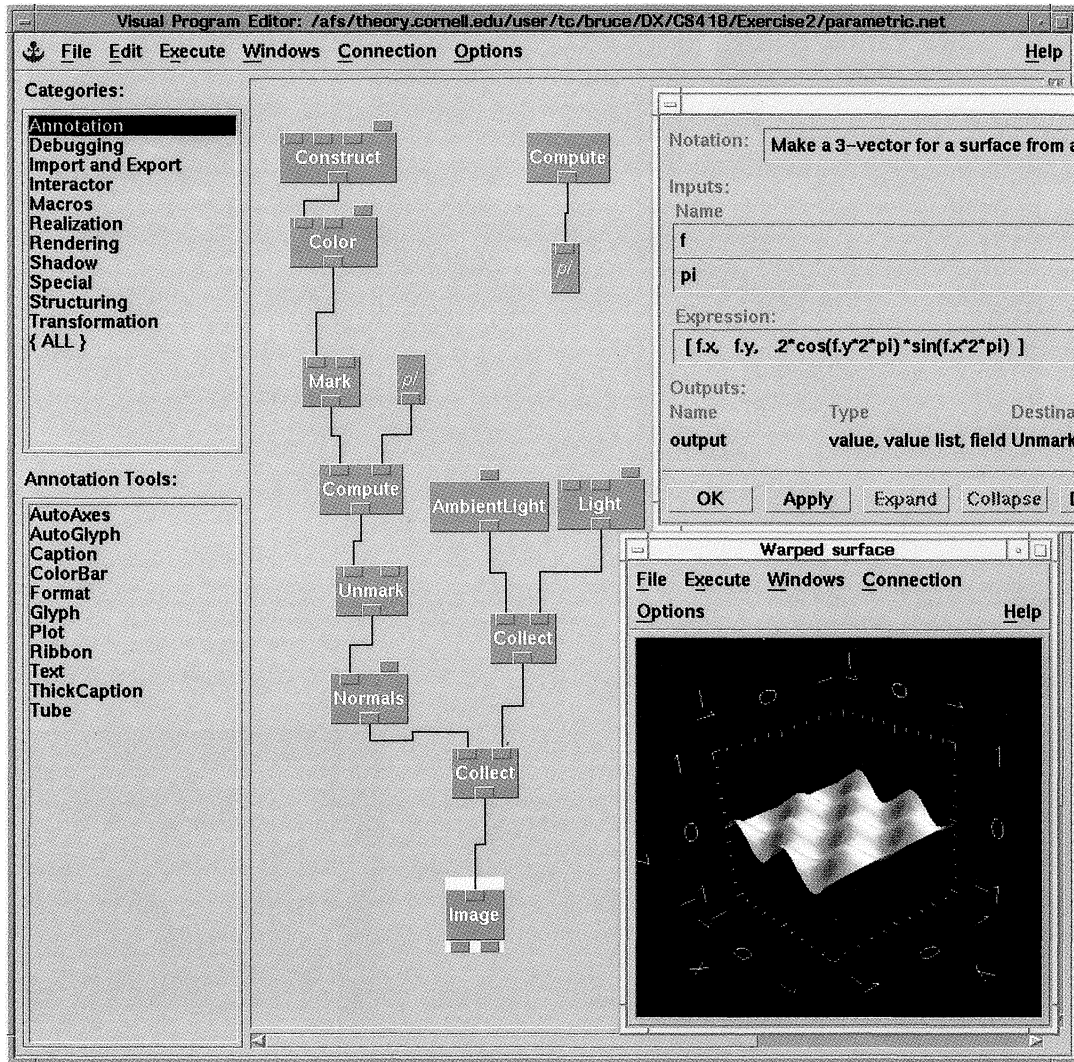


Figure 5 A DX program example. The program shown generates a flat 2D surface, deforms it to a 3D function of two variables, applies a lighting model, and renders it. The resulting image is shown in the lower right. The expression in the compute module which deforms the surface is shown in the upper right.

the course with a tangible image of the kind of work they can be expected to produce at the end of the semester. The students can view the previous year's final design projects throughout the semester. These animations serve as illustrations of the techniques which the students are currently studying. The final design projects serve to set a level of expectation for the course, define a style and allow students to plan for their final animation projects from the first day of class. As creators of animations, the students are motivated by the large potential audience to refine their work carefully. There are several issues regarding intellectual property rights that were addressed. We do not require that students post their

animation to the Net. In our view, the students clearly own their work and may not want to share it for any number of reasons. One side effect of this guideline is a level of self-selection of work that is published on the server, with the resulting tendency for only the best work to be submitted for inclusion.

Course evaluations were conducted for each of the 3 years this lab course has been taught. In general, the results indicated overall student satisfaction with the course. Comments were encouraged on the evaluations. Many students commented on the ease of use of the programming environment: "Excellent software—allowed us to concentrate on graphics rather than computer programming"; "The soft-

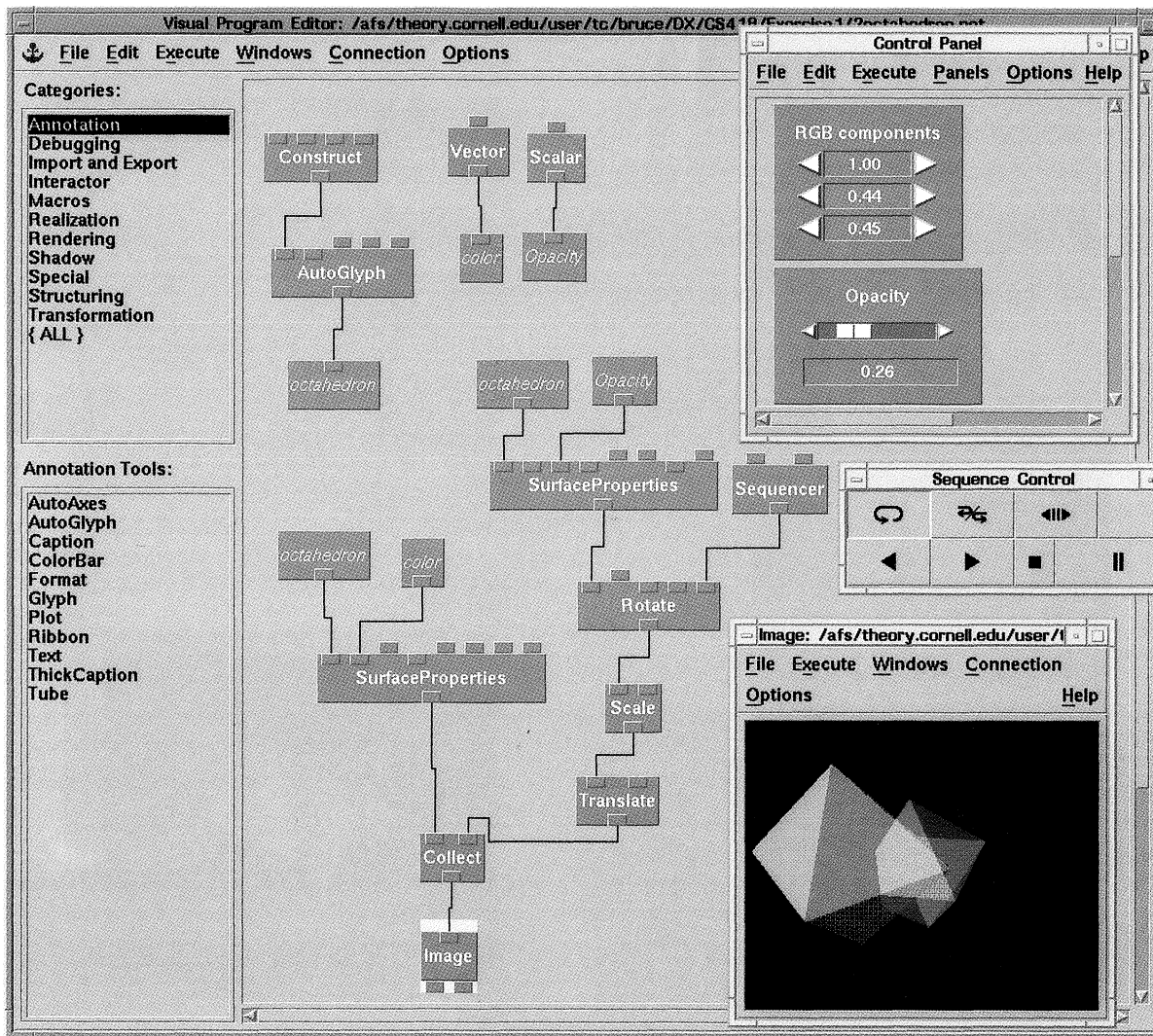


Figure 6 A DX program example. Shown here is the program which generates two octahedra and animates them. In the upper right corner is the control panel, defined by the student to vary the color of one shape and the opacity of the other. In the middle right is the sequencer which controls the animation. At the bottom right is the image corresponding to one frame of the animation.

ware was very simple and allowed easy application of the techniques from lecture”; “For a teaching environment, DX makes you do the right amount of stuff yourself to make you learn how the graphics details work.” Not surprisingly, a few students wanted to delve into the details of the inner workings of the graphics software: “Labs were too far from internal workings of Data Explorer”; “I liked DX but a few of the assignments should be in C so we could become familiar with graphics at a lower level.” The goal of the course was not to teach the inner workings of DX, but rather to provide the students with tools for scientific visualization and

creating computer graphics. The majority of responses indicated satisfaction with DX as a vehicle for learning the visualization and graphics techniques.

In the spring 1995, in addition to the course evaluation, we administered a pre- and postcourse questionnaire to examine the students’ attitudes toward and uses of the Web as a tool for enhancing teaching and learning. Because all of the lab materials were on the Web site, no paper was handed out for the labs. We found that about a third of the students printed less than 10% of the material, but a fourth printed 80% or more of the lab pages. This suggests

that although many students are eager to make nearly exclusive use of on-line information, there is still a significant number of students who desire to have the materials in print format. Of course, the use of the Web makes accommodating both of these groups rather easy. A full 60% of the students indicated on the postcourse questionnaire that the use of Web for course materials increased their productivity in labs. On the precourse questionnaire 40% of the students indicated that they felt that having materials on-line would help them learn at their own pace. This expectation was more than met, because on the postcourse questionnaire 55% of the students indicated that they were better able to learn at their own pace.

The availability of last year's animation projects influenced a large percentage of the students. About 30% of the students got ideas from last year's projects and another 35% indicated that it increased their motivation. In the judgment of the instructor, overall the projects were more elaborate. Many students commented on pride of workmanship and enthusiasm for having their final projects shared with not only their friends, but the world at large. Several students noted that putting the animation up on the Web site would become part of their portfolio as they created their job resumés. The two major suggestions that emerged from this assessment were, first, a strong desire on the part of the students to have the full course notes available on-line. This is now a current effort under way. Second, the students expressed a desire for a "chat" facility to support communications among the students and between the students and the course instructors and teaching assistants. This is now in place for the next course offering.

CONCLUSIONS

The visually oriented, block diagram language allowed the students to concentrate on learning the

applications and graphics content rather than focus on the details of a programming environment. Students were able to generate code to create images in a fraction of the time required of a C language environment, thus encouraging both experimentation and exploration. The acquired background in computer graphics was readily put to use in scientific visualization as students applied their skills in research projects across campus. The introduction of the Web site provided several advantages to the students, including access to all lecture and lab materials and additional related resources. The Web site also provided a forum for students to publish their final work to the larger research community. We anticipate that the work published last year will serve as a model and as a set of building blocks for new project ideas by this year's students, much as the video presentation had done in the past.

REFERENCES

- [1] R. D. Pea, "Beyond amplification: Using the computer to reorganize mental functioning," *Educational Psychologist*, Vol. 20, No. 4, 1985, pp. 167-182.
- [2] B. Land, "Teaching computer graphics and scientific visualization using the dataflow, block diagram language Data Explorer," in *University Education Uses of Visualization in Scientific Computing*, Vol. A-48, 1994, pp. 33-36. Amsterdam, Netherlands: International Federation for Information Processing.
- [3] The lab exercise software plus student lab instructions are available on-line from the anonymous ftp site at Cornell University (<ftp://tc.cornell.edu>) or <http://www.tc.cornell.edu:80/Visualization/Education/cs418/>.
- [4] J. D. Foley and A. Van Dam, *Computer graphics: Principles and practice*. 2nd Edition. Addison-Wesley, Reading, Massachusetts, 1990.
- [5] A. Watt, *3D computer graphics*, 2nd Edition. Addison-Wesley, Reading, Massachusetts, 1993.

BIOGRAPHIES



Helen M. Doerr is an assistant professor of mathematics education at Syracuse University. Her current research interests include mathematical modeling, functions, and problem solving. She continues to examine the use of networking technologies to support teaching and learning. Prior to her appointment at Syracuse, she was the associate director of scientific computational support for the Cornell Theory Center. She was responsible for the overall on-line information, education and training programs at the Theory Center, as well as the technical support and academic outreach efforts of the center in support of the national scientific community.



Bruce Land is the project leader of the visualization group at the Cornell Theory Center. He also teaches in the Computer Science Department at Cornell. Land received a PhD in neurobiology in 1976 from Cornell University, and a BS in physics from Harvey Mudd College in 1968. He was a lecturer in the Neurobiology Department at Cornell for 10 years and did research on coupling of activity at the vertebrate neuromuscular junction, both experimentally and by computer modeling. During this time, he taught in the Electrical Engineering Department at Cornell for 2 years. He came to the Cornell Theory Center in 1986 as a computational research associate, then started supporting graphics and animation in 1987. In 1992, he started teaching an introductory graphics course in the CS Department at Cornell.