COMPUTER GRAPHICS AND PARALLELISM, AN INTERDISCIPLINARY FABLE

Nan C. Schaller
Department of Computer Science
Rochester Institute of Technology
Rochester, New York 14623
(716) 475-2139
e-mail: ncs@cs.rit.edu

ABSTRACT

A tale of collaboration on graphics projects between undergraduate students leads to speculation concerning the ingredients for success in such projects. This article attempts to identify those elements of the academic environment that foster open and collaborative learning.

THE FABLE

This is a tale of three bright undergraduate computer science students and some projects that grew out of their separate but complementary interests. This combination of talents fueled a collaboration that has been both productive and spectacular. In the process, tools from different subdisciplines of computer science have been utilized for projects that are mainly graphical in nature.

Student number one, Eric Law: Even before beginning his undergraduate computer science concentration in graphics, Eric had experimented in the field, exploring such areas as ray tracing and simple texture mapping. Although I, as his instructor, wondered what he might learn from an introductory graphics course, Eric found that many of the fundamental concepts covered made his approach to graphics easier. For his Computer Graphics Laboratory project, he explored ray-tracing acceleration techniques and texture mapping. Eric subsequently completed a three-dimensional, interactive, neutral file format [HAI87] editor as an independent study project. He is currently working to integrate his previous graphics projects into a consistent system. For example, his three-dimensional editor can now produce input for his animation package which, in turn, generates data for his ray-tracer. Eric has also generated software for displaying images on many devices.

In 1989, while on co-op in the Boston area, Eric attended SIGGRAPH '89 and met many of his idols in the field. This had a profound effect. He says, "I found that they are just like me, and that I am capable of the work they do - given a little determination, that is. I came back and noticed how different the world was. My grades were consistently improving." Eric feels that, "Graphics is my niche. I feel sorry for those who just drift aimlessly through school. You need to 'get into' your major somehow. Graphics is visually oriented, well in tune with how I think."

Eric is often called upon to demonstrate his work for prospective students and their parents. He feels that this is one of the most gratifying things he has done during his five years at Rochester Institute of Technology (RIT). Knowing the administrators as well the professors in the department makes him feel significant and useful. He loves the recognition. He says, "As a result of my increased visibility, I am often sought by other people for help (graphics related or not). My willingness to explain details may be to blame, or perhaps it is my intense interest in the topics. Either way, it makes being here worthwhile, which has helped improve my academic performance."

Student number two, Michael Kirby: Mike also chose graphics as his computer science concentration. Even in the introductory computer graphics course, he showed initiative, implementing graphics algorithms that were beyond the scope of the course (fractal mountains and "sticky point" particle systems). Mike's participation in the second course, Computer Graphics Laboratory, resulted in some unique opportunities for him. He presented a paper on the Dynamic Modeling System, an interdisciplinary project that he was involved in, for the Seventh International Conference on Mathematical and Computer Modeling in Chicago (August 1989). [KIR89, TOR89] During that same summer, he became involved in an NSF sponsored Undergraduate Research Experience (URE) program at Cornell University. This project was again an interdisciplinary one. The other participants were a film and video professor and an art student. Their goal was to do three-dimensional animation using supercomputers. The URE program opened Mike's eyes to the possible speed advantages to be derived from utilizing parallelism in computer graphics.

Mike's interest in performance grew as he and Eric worked together to ray-trace increasingly more complex images. Their first attempt at increased performance was to utilize the auto-vectorizing C compiler on a Stardent GS1000 at Cornell. They were ray-tracing a 1048 X 1048 pixel image that contained over 87,000 objects, 5 light sources, and simple texture mapping. It took approximately 36 hours to gen-

erate. Because the ray-tracer had not been written with vectorization in mind, the result was virtually no speedup. Mike then took a co-op job where he had access to a Sun 4/490 workstation. Utilizing the workstation during off-hours and modifying the ray-tracing algorithm (Subsequent analysis had shown, for instance, that input alone had taken over two hours), Mike was able to ray-trace this same model in approximately 18 hours.

About this same time, Mike and Eric began working on a three-dimensional animation package that simulated the interactions between bouncing balls. They were also featured in an article, "Chaos: A New Order", in a rare color edition of *Reporter*, the RIT magazine. [FIS90]

Enter student number three, Peter Portante. Peter took a seminar in parallel computing from me. Aware of Eric and Mike's continual quest for CPU cycles, he recognized the potential for speedup if the ray-tracing computations were to be done in parallel. Peter decided to use this as the basis for his term project in the parallel computing seminar. The Dynamic Task Distribution System was designed to allow an application utilizing a farmer/worker model to distribute its work across multiple processors in a VAXcluster environment. (The RIT VAXcluster consists of six VAX 6000 and 8000 class machines, including one with two processors.) Synchronization was handled by utilizing the VMS distributed lock manager [VAX1, VAX2]. In the farmer/worker model, the farmer processor is responsible for doling out tasks to the worker processors. Each processor runs the same program, with the first to start execution assuming the role of farmer. All other processors become workers. To use this system, an application must provide an initialization routine, a sequencer, and a task definition. The initialization routine is called when each worker begins. The sequencer is used by the farmer when a worker processor requests a task. The task definition is used by the worker process when it receives an assignment from the farmer. The task definition can be any type of parallel application that is suitable to the farmer/worker approach. In this case, it was Eric's ray-tracer that Eric, himself, had modified so that its work could be partitioned: each worker requested a task from the farmer processor which, in turn, indicated which portion of the image that processor was to ray-trace next.

Upon testing the first version of this system, Peter noticed that the differences in speed of the individual processors caused a large difference in termination times. At Mike's suggestion, the algorithm was modified so that the portion of the task assigned to each worker was not uniform. Given N, the number of worker processors, only half of the work was allocated for the first N requests for a task. When the first processor came back for more, the remaining work was again halved and appropriately sized chunks were passed out as they were requested. This work division was continued until a minimum threshold was met. When all the workers were finished, the farmer concatenated and displayed the image.

Using his system, Peter was able to achieve significant speed-up (approximately 4 using 6 workers).

Although Peter had no graphics background, his interest in the field was so stimulated by his contacts with Mike and Eric, as well as by this project, that he has since taken the introductory graphics course. Eric says, "Peter has great skill in the systems area. He knows what can be done and how to do it. His analytical skills are very good. Once he moved in (Peter is Eric's roommate), he became interested in graphics. He saw how his systems experience could be utilized. I think my ray-tracer was a way for him to visualize his work."

During the 1990 winter quarter, the trio proposed a project for their Distributed Systems Laboratory course in which they would implement the Linda parallel programming paradigm on the department's network of Sun 3 workstations. The Linda paradigm uncouples processors in a multiprocessor environment in terms of both time and space. Its "tuple-space" can be viewed somewhat like a distributed database, providing a repository for information to be shared between processors. Once again their goal was to find lots of CPU cycles for graphics applications. Their proposal did not fit into the traditional framework of this course. Their instructor, however, was so impressed by the amount of research and planning that they had done before the course had even started that she readily agreed to their proposal. The class took a field trip to look at Cornell University's implementation of the Linda paradigm as part of their preparatory work. The trip proved to be extremely stimulating. It made them realize that in doing their project they would be exploring an area on the forefront of technology. The project was a success (in terms of a learning experience) and Peter is currently extending and fine-tuning it as an independent study.

THE MORAL

Every fable must have a moral, so what is the moral here? This is an example of the educational process at its best. I can't claim responsibility for it. It was an accident, ... or was it? I have thought about why this worked. I have asked Mike, Eric, and Peter why they thought it worked. None of us could put our fingers on one particular reason. It may, in fact, be pure serendipity. However, I want to share with you the features of the RIT curriculum that I believe were the catalysts for this collaborative learning "experiment". I want to do this in the hope of identifying mechanisms that might encourage this process to repeat itself.

(1) Visual Projects Help Students Learn About Other Subdisciplines

According to Eric, "Errors in graphics programs are visibly detected. In some ways it makes debugging easier." Graphics is a non-trivial application. To do it right, one must use tools found in many fields. In this story, three bright students made graphics central to their life. In doing so, they grasped at tools from other computer science disciplines, to gain the performance they sought. Specifically, they made extensive use of the tools of parallel computing, an exciting new field which offers many benefits to graphics in the long run.

Computer graphics, in turn, is itself a fruitful source of non-trivial problems for disciplines such as parallel computing. It provides natural examples that are immediate and exciting to the student. Many graphics algorithms are well suited to parallelization. In some, for example, Mandelbrot sets and ray-tracing, the work done for each pixel is essentially independent. Others, such as two-dimensional Fourier Transforms for image processing applications, require nontrivial patterns of communication between neighboring pixels. The recent availability of inexpensive parallel equipment has made the study of parallel computing at the undergraduate level a possibility. But we cannot teach graphics in a parallel computing course or parallel computing in a graphics course. We see here, however, that cross-fertilization can work to the benefit of students whose interests are in both areas.

(2) **An Approachable, Flexible Faculty Opens the Door to Collaborative Learning**

A key ingredient in the recipe is that faculty members be both approachable and flexible. According to Peter, "The project worked for two reasons. The first is that you were open to the idea of my working on systems that were not planned to be part of the course. If I had been restricted to working on transputers, I would not have done anything like this at all. The second reason is that ISC (the RIT's computer center) supported me in my work. They gave me extra disk space for this special project. They don't do that for everybody, but the task I was undertaking was big enough to warrant it."

The Distributed Systems Laboratory instructor also showed flexibility in allowing the students to develop a project that did not fit the standard format of that course.

(3) **Laboratory and Seminar Courses Can Promote Collaboration**

The open-ended format of both the seminar and laboratory courses that are offered at RIT gives the student permission to think in an open and imaginative way. In the Computer Graphics Laboratory course [SCH90], for example, students are encouraged to look at graphics from the outside. They are encouraged to figure out for themselves what information they will need in order to be able to complete their projects. They must then gather that information, through research, brainstorming with other members of the class, and/or through questioning experts. Furthermore, they are exposed to professionally produced graphics packages as well as those produced by students in previous quarters. Students report that this challenges them to go out and produce work of professional quality.

Eric reports, "Natural progression led me into the graphics lab course. During this course I saw other groups working on ambitious projects such as fractal exploration, a CAD system, and a Dynamical Modeling System. Gee, all of those projects sound more interesting than mine!"

All of this serves to give students the courage to say to the teacher, "Here's something I'm interested in. It's related in this way to what we're studying in class. How about it if I try it?"

(4) **Team Work and Collaborative Learning**

"Students learn better through non-competitive collaborative group work than in classrooms that are highly individualized and competitive." [BRU87] At RIT, the student is introduced to team work early in the curriculum. When group work is allowed for a class project, the size of a project can be much larger. The student gains respect for and knowledge from his/her classmates. Eric says, "We can work together well on projects because we've worked in teams before. Also, we all went through the same curriculum, learning the same concepts of design, trade offs, etc. Working in groups is 'key' because of the size of the projects."

(5) **An Open Laboratory Environment Promotes Collaborative Learning**

RIT's computer laboratories open the door for student interaction. They serve as a meeting place for students in the program. Eric reports, "I think the labs have something to do with it too. The white boards available for general use make a great teaching tool. I frequently find myself explaining ideas that I've read about to other students, in the hope that they will become interested in graphics and even implement these ideas (I rarely have the time). The graphics lab is about the right size to have inter-student camaraderie work. The labs promote an informal introduction to people and their current work. I have, for instance, given impromptu lectures about three-dimensional matrix transformations to students working on their Computer Graphics Laboratory projects, such as the one who was trying to visualize sonar data in three dimensions."

(6) **Co-op and Real Life Projects Provide Stimulation**

One of RIT's strong points is co-operative education. The computer science student is required to spend four quarters working in industry. This exposure to the work environment helps the student to tie the computer science curriculum to reality. Courses, such as the Computer Graphics Laboratory Course and the Parallel Computing Seminar, which allow a student to work on real-life interdisciplinary projects also contribute to this. Peter says, "Another reason why my application was successful is that it could be applied to the projects of various engineering professors who are in need of more CPU power. My project has produced a cost effective software product that will enable better use of the VAX systems at RIT. This is another reason why ISC supported it. Note that my parallel computing project has real world applications. It is not just a simulation of a hypothetical situation. Students need to get their hands dirty on real problems. That is what makes that stuff happen."

CONCLUSION

This fable illustrates the positive effects of collaborative learning and interdisciplinary projects. In particular, it shows that collaborative visual projects are an immediate and exciting avenue to learning about other subdisciplines of computer science. Thus, computer graphics can be more to the computer science curriculum than "just another application" area. It can be a tool for stimulating learning. Here I have suggested what I believe might be important ingredients in the creation of an environment that fosters open and collaborative learning. Open-ended courses, such as the RIT Computer Graphics Laboratory course, which have an interdisciplinary flavor, put students and faculty alike in the mind frame to accept "off the wall" projects. But, encouragement and support for collaborative work is most important for continued stimulation.

REFERENCES

[BEN90] Ben-Ari, M., *Principles of Concurrent and Distributed Programming*, Prentice-Hall, 1990.

[BRU87] Bruffee, Kenneth A., "The Art of Collaborative Learning", *Change*, March/April 1987.

[FIS90] Fiske, Barton, "Chaos: A New Order", *Reporter*, April 4, 1990.

[HAI87] Haines, Eric, "A Proposal for Standard Graphics Environments", *IEEE Computer Graphics and Applications*, November 1987.

[VAX1] *Introduction to VAXcluster Application Design.* (incomplete)

[KIR89] Kirby, M.P., Schaller, N.C., and Torok, J.S., "Visualization of Strange Attractors", *Proceedings Seventh International Conference on Mathematical and Computer Modelling, August 2-5, 1989, Chicago, Ill..*

[SCH90] Schaller, N.C., "Experiences with Scientific Visualization Applications in an Undergraduate Computer Graphics Laboratory Course", *Educator's Seminar - Education for Visualization*, SIGGRAPH '90, Dallas, August 1990.

[TOR89] Torok, J.S. and Schaller, N.C.,"Visualization in Scientific Computing - An Interdisciplinary Enhancement to Engineering Education", *Proceedings 1989 IEEE Frontiers in Education Conference.*

[VAX2] *VAX/VMS System Service Reference Manual,* Part I, Section 12. (incomplete)

****

COMFORT-- continued from page 60

Gerver, E. (1989), Computers and gender. In T. Forester (Ed.), Computers in the Human Context (pp. 481-501). Cambridge, MA: MIT.

Hill, T., Smith, N. & Mann, M. (1987). Role of efficacy expectations in predicting the decision to use advanced technologies: The case of computers. Journal of Applied Psychology, 72, 307-313.

Howard, G. & Smith,R. (1986). Computer anxiety in management: Myth or reality?, Communications of the ACM, 29(7), 611-615.

Kersteen, Z., Linn, M., Clancy, M. & Hardyck, C. (1988). Previous experience and the learning of computer programming: The computer helps those who help themselves. Journal of Educational Computing Research, 4(3), 321-333.

Kiesler, S. & Sproull, L. (1985). Pool halls, chips, and war games: Women in the culture of computing. Psychology of Women Quarterly, 9, 451-462.

Konvalina, J., Wileman, S. & Stephens, L. (1983). Math proficiency: A key to success for computer science students. Communications of the ACM, 26(5), 377-382.

Kramer, P. & Lehman, S. (1990). Mismeasuring women: A critique of research on computer ability and avoidance. Signs, 16(1).

Lockheed, M. (1985). Women, girls and computers: A first look at the evidence. Sex Roles, 13(3/4), 115-122.

Marcoulides, G. (1988). The relationship between computer anxiety and computer achievement. Journal of Educational Computing Research, 4(2), 151-158.

Miura, I. (1987). The relationship of computer self-efficacy expectations to computer interest and course enrollment in college. Sex Roles, 16(5/6), 303-311.

Ogozalek, V. (1989). A comparison of male and female computer science students' attitudes toward computers. SIGCSE Bulletin (Special Interest Group on Computer Science Education of the ACM), 21(2), 8-14.

Shneiderman, B. (1986). Empirical studies of programmers: The territory, paths, and destinations. In E. Soloway and S. Iyengar (Eds.) Empirical Studies of Programmers (pp. 1-12). Norwood, NJ: Ablex.

Whipkey, K. (1984). Identifying predictors of programming skill. SIGCSE Bulletin (Special Interest Group for Computer Science Education of the ACM), 16(4), 36-42.