Designing Computer Learning Environments for Engineering and Computer Science: The Scaffolded Knowledge Integration Framework[1]

Suggested Running Head: "Designing Computer Learning Environments for EE and CS"

Marcia C. Linn[2]

Submitted to: *Journal of Science Education and Technology*

October, 1994

---

[1] Portions of this paper were presented at the American Psychological Association Meeting, Ontario, Canada, August 22, 1993, under the title of "Cognition and instruction in higher education: applications of advanced technologies." The title of the symposium was "New Fellows in Educational Psychology—the implications of their work for university-level instruction."

The author thanks all the participants in the LISP–KIE work including Michael Clancy, who co-directed the project, Lydia Mann, John Bell, Elizabeth Davis, Christopher Hoadley, Michael Katz, Annika Rogers, and Christina Schwarz. The author also thanks all the participants in the spatial reasoning project including Alice Agogino, who co-directed the project, John Bell, Sherry Hsi, and James Osborn. Special thanks go to two anonymous reviewers. Thanks also go to Anna Chang, Madeleine Bocaya, and Dawn Davidson for assistance in production of the manuscript.

This material is based upon research supported by the National Science Foundation under grant MDR-8954753. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and not necessarily reflect the views of the National Science Foundation.

[2] 4611 Tolman Hall
Graduate School of Education
University of California, Berkeley
Berkeley, CA 94720
(510) 643-6379

ABSTRACT

Designing effective curricula for complex topics and incorporating technological tools is an evolving process. One important way to foster effective design is to synthesize successful practices. This paper describes a framework called *scaffolded knowledge integration* and illustrates how it guided the design of two successful course enhancements in the field of computer science and engineering. One course enhancement, the LISP Knowledge Integration Environment, improved learning and resulted in more gender-equitable outcomes. The second course enhancement, the spatial reasoning environment, addressed spatial reasoning in an introductory engineering course. This enhancement minimized the importance of prior knowledge of spatial reasoning and helped students develop a more comprehensive repertoire of spatial reasoning strategies. Taken together, these instructional research programs reinforce the value of the scaffolded knowledge integration framework and suggest directions for future curriculum reformers.

KEYWORDS

instructional technology

spatial reasoning

programming instruction

higher education

cognitive processing

science instruction

INTRODUCTION

Scaffolded knowledge integration, a framework for instructional design, guided the refinement of two university level courses as described in this paper. A computer science course in introductory LISP implemented the LISP Knowledge Integration Environment (LISP-KIE). An engineering course that introduces graphical communication implemented the spatial reasoning environment. In both cases, a process of trial, refinement, and synthesis, informed by the scaffolded knowledge integration framework, was followed. This framework, resulting from research on the Computer as Learning Partner curriculum, informed the design of the two courses described in this paper and holds promise for governing design of future courses.

Instructional designers commonly lament that research, especially cognitive research, is too general to be used for making decisions in the design of complex instruction. On the other hand, they also complain that specific examples of effective instruction are often too idiosyncratic to provide principles or generalizations for further work. This paper describes intermediate-level generalizations that are less abstract than those typically found in cognitive psychology but more useful than descriptions of specific design decisions or innovative courses. Essentially this paper illustrates that trial and refinement, which has been shown to benefit the design of complex curricular innovations, can also yield rules of thumb and conjectures to guide future designers.

A similar research paradigm for understanding human interactions with complex systems such as flight control systems or nuclear reactors has been advocated by Norman (1988), Winograd and Flores (1987), and Hutchins (in press). Hutchins describes the process as studying cognition "in the wild." Both Hutchins and Norman advocate gathering a corpus of natural histories, interpreting the corpus using cognitive concepts, and abstracting patterns that go beyond the particular to identify regularities in human cognition. In this paper a corpus of natural histories of curricular innovations are interpreted using concepts from cognition. A framework for curriculum innovation results from this analysis.

The framework guiding the design of the two innovative curricula described in this paper derives from the successful design of the Computer as Learning Partner (CLP) curriculum (Lewis & Linn, 1994; Linn, 1992b; Linn & Songer, 1991). The CLP design resulted from a process of trial and refinement with four main steps. First, expert and student ideas about the topic to be taught, were analyzed to define potential endpoints for the instruction and characterize the diversity of perspectives. Second, a preliminary design for the instruction, incorporating prior research on related topics was created. This preliminary instruction was implemented and tested in a typical class. Third, the results from the preliminary test were used to create a refined version of the instruction. This process of testing and refinement was repeated until the instruction met appropriate criteria. Fourth, the process of testing and refinement was synthesized and integrated with related research. Often this integration suggested new criteria and motivated the design of a new version of the CLP curriculum (e.g., Linn *et al.*, in press).

The scaffolded knowledge integration framework resulted from a synthesis of four versions of the Computer as Learning Partner (CLP) curriculum. This research was carried out in a middle school where the CLP curriculum is implemented in a semester-long course designed to improve students' understanding of thermodynamics (Linn, 1992b; Linn & Songer, 1991; Linn *et al.*, 1993; Songer & Linn, 1991).

The research team for CLP, LISP-KIE and spatial reasoning each involved experts on technology, pedagogy, and cognition. Innovative projects often feature partnerships among experts in a range of specialties. Such activity is growing on college and university campuses today, often spurred by the availability of technological tools (Beshears, 1990; Kuo, 1988; Lan, 1989a; Lan, 1989b; Linn, 1989; McGrath, 1989a; McGrath, 1989b).

In the next section, this paper describes the scaffolded knowledge integration framework that emerged from the CLP experience. Then the trial and refinement process for both the LISP–KIE project and the spatial reasoning environment are described. The last section of the paper synthesizes these experiences into an updated version of the scaffolded knowledge integration framework and discusses implications of this work.

SCAFFOLDED KNOWLEDGE INTEGRATION FRAMEWORK

Scaffolded knowledge integration is a method for encouraging students to develop integrated understanding of a complex domain. To achieve integrated understanding students link and connect ideas. Students with integrated understanding have a cohesive view of a domain, and can apply their ideas to personally relevant problems. Integrated understanding is reflected in students' willingness to look at anomalies and exceptions and try to determine how to fit them into their conceptual framework. In contrast, sometimes instruction discourages students from analyzing their own understanding and instead suggests that students just adopt the views presented by the instructor.

In science, where intuitive ideas held by students are often quite different from those of experts, building on student ideas is particularly important (Carey, 1985; diSessa, 1988; Eylon & Linn, in press; McCloskey *et al.*, 1980; West *et al.*, 1984). The scaffolded knowledge integration framework looks at student ideas as building blocks rather than impediments to understanding (Linn & Songer, 1991). As a result, students are encouraged to reconcile anomalies rather than just memorize isolated pieces of information.

The scaffolded knowledge integration framework reflects a view of the learner as holding a repertoire of models for complex phenomena and working to expand, distinguish, reconcile, refine, and link these models (see also Linn *et al.*, 1994). Models are defined broadly to include, heuristics, algorithms, rules-of-thumb, formal mathematical systems, abstract representations, and mechanisms. Conceptual change in this framework consists of realignments and additions to the repertoire of models. The goal of instruction is to motivate learners to consider new models and to integrate new models with their existing perspectives. Criteria for an ideal repertoire include parsimony, utility, and flexibility. In this paper, the focus is on designing instruction that motivates students to integrate new models with existing views and to distinguish among models.

The scaffolded knowledge integration approach characterizes the repertoire of models used by experts and students and then identifies a process that will allow students to add more sophisticated models to their repertoire. This process does not necessarily culminate in students becoming experts, but rather is designed to lay the groundwork so that students will have a firm foundation for further learning. In many cases, this foundation will prepare students to be informed citizens rather than experts.

This repertoire of models approach stands in contrast to instruction designed to instill correct models by diagnosing weaknesses and correcting them. For example, several recently developed tutoring programs seek to model students' understanding and provide instruction that modifies specific rules (e.g., Anderson, 1993). Others advocate "replacing" models as discussed by Smith, diSessa, and Roschelle (in press).

The scaffolded knowledge integration framework that supports expanding the repertoire and distinguishing among models has four main aspects: (a) identifying new goals for learning; (b) making thinking visible; (c) encouraging autonomous learning; and (d) providing social supports. These four aspects of the framework guided design of the CLP curriculum. They were then used to design the LISP-KIE and the spatial reasoning environment described in subsequent sections of this paper.

Identifying New Goals for Learning

Research on experts and novices suggests that expert models may not be the best goal for all introductory courses (Lewis & Linn, 1994). For example, it may be that expert models are so abstract and mathematical compared to students' models, that intermediate models are appropriate and these intermediate models may prove to be more fruitful for everyday problems than more abstract models. Indeed, often models used by experts lead to fragile understanding or number crunching when taught to students (e.g., Reif & Larkin, 1991).

For example, the CLP project found that students' ideas about heat and temperature were sufficiently mired in their everyday observations that the best model to help them make sense of thermodynamics was a heat flow model. It was simply too difficult for students to link their everyday observations to molecular kinetic theory. Rather, they first needed to gain some systematic understanding of their everyday experiences. Adding the heat flow model provided a firm foundation for molecular kinetic theory (Linn, 1992b) while also preparing students to reason about everyday problems (e.g., Lewis & Linn, 1994). Building on current ideas and developing a more sophisticated repertoire of models is a lifelong learning skill.

Three main ideas guide identification of new goals for scaffolded knowledge integration: (a) select models that build on students' intuitions, (b) seek ways to emphasize knowledge integration and cohesion, and (c) provide students with the resources to continue the sense-making process after they finish the current course.

Selecting models to build on intuitions means that curriculum developers need to respect the deliberations that characterize student sense-making activities and ensure that course activities make realistic demands on students. Since students spend far more time processing everyday experiences than they do solving abstract problems presented in science classes, over time, students are likely to return to their intuitive ideas, unless they have incorporated the models taught in science classes. The reasoning processes of students often accord respect to "evidence" that experts would ignore (Inhelder & Piaget, 1969; Kuhn *et al.*, 1988). Ensuring that students can understand and link new models makes courses more equitable and motivates instructors to help students learn. Many instructors abandon student-relevant practices because the textbook or framework requires them to "cover" too much or to communicate "esoteric" information. As a result, students often fail to utilize models taught in courses after the course is over because these models are disconnected from their experience and too abstract to apply to everyday problems.

Seeking ways to emphasize knowledge integration and cohesion ensures that students bring their ideas together rather than isolating them, adding new models learned from courses and distinguishing when these new models apply. Thus, the CLP project found that students come to class with descriptive models of thermodynamics based on personal experiences. For example, students remark, "Wool warms things up, so wrapping a cold drink in wool will heat it up." Here students base their view on an accurate observation that when they put on a wool sweater, they warm up. To help students bring ideas together, CLP encouraged students to add the heat flow model and to distinguish when each of their models apply.

Providing students with the resources to continue sense-making after they finish the course involves helping students recognize the role of new models of scientific events. Students who see themselves as expanding and refining a repertoire of models can effectively guide their own learning. Students who believe that scientific advance proceeds by fits and starts are likely to add models and distinguish among them. If the social nature of scientific knowledge construction forms a part of their experience, then students can incorporate models of social interaction into their view of the scientific enterprise and into their view of their own learning. In the CLP project, opportunities to learn about the nature of science have contributed to deep

understanding of scientific topics (Linn & Eylon, in press; Linn & Songer, 1993; Songer & Linn, 1991).

Thus, key features of new goals for scaffolded knowledge integration are to select models that build on student intuitions, encourage knowledge integration, and to help students view themselves as continuously refining scientific ideas. New goals that reflect these criteria emphasize choosing among alternatives for solving problems rather than learning only the correct outcome. They often rely on models of phenomena that are more pragmatic than those used by researchers in the discipline.

Making Thinking Visible

The second major feature of the scaffolded knowledge integration framework is to make thinking visible. Rather than describing only the outcome of a problem solution, only the correct path, or only the most powerful model, in order to scaffold students, the process of thinking needs to be illustrated for all learners. One aspect of visible thinking is to provide dynamic models that support diverse forms of expertise consistent with the recommendation of Collins, Brown, and Holum (1991). Offering multiple representations promises to meet the needs of a wider range of students.

Activities that make thinking visible include dynamic models of complex processes as well as explicit descriptions of thought processes. Often dynamic models illustrate thought processes that are difficult to communicate verbally. For example, Heat Bars in the CLP curriculum made the process of heat flow visible as illustrated in Figure 1. This software illustrates a powerful model and helps students link it to their own experience. Case studies and tutoring can also make thinking visible by providing guidance and explicit information about problem-solving processes, by illustrating how researchers select among models, and by showing how learners recover from following an unsuccessful path (Linn & Clancy, 1992b). A variety of research projects have developed instruction that use multiple representations to communicate problem solving processes (e.g., Champagne *et al.*, 1982; Larkin & Reif, 1979; Resnick, 1981; White, 1993; White & Frederiksen, 1990).

Encouraging Autonomous Learning

Making thinking visible and explicit runs the risk of convincing students that they should passively memorize rather than personally integrate scientific ideas. A major component of the scaffolded knowledge integration framework is transferring responsibility for understanding to the student. This means renegotiating the authority structure in most classrooms such that students learn to evaluate evidence rather than relying on the teacher to say what is right. Yet at the same time, if students are given responsibility without skills necessary to exercise that authority, they may flounder rather than integrate their ideas as was often observed in guided discovery learning programs (e.g., Duschl, 1990; Raizen, 1991). Thus, encouraging autonomous learning involves both transferring responsibility and ensuring that students can exercise the responsibility effectively.

When students take responsibility for learning, they both create and critique their ideas. They reflect on their progress and initiate activities to make their understanding more robust and comprehensive. Expert autonomous learners seek more and more cohesive and powerful explanations for phenomena. Yet, these autonomous learners also monitor their progress and make reasonable decisions about when to persist and when to abandon attempts at syntheses. Autonomous learners engage in what is often called metareasoning because they reflect on their own learning. Students gain skills as autonomous learners when given (a) opportunities to act as

investigators and critics developing criteria for evaluating evidence, (b) invitations to reflect on alternative interpretations of phenomena, and on how scientific ideas change and progress, and (c) opportunities to engage in sustained reasoning and to develop expertise in a circumscribed area.

In the CLP project, students have many opportunities to participate as investigators and critics and develop ways to evaluate scientific evidence. For example, after students conduct their first experiments, they jointly critique an experiment composed of elements drawn from several class investigations and create a list of class criteria. Students are regularly prompted to reflect while using the electronic laboratory notebook to conduct experiments. For example, students are asked to generate examples from personal experience in keeping object warm or cold and to describe how their personal examples and their classroom experiments fit together.

To give students a better sense of how alternative interpretations arise and of the nature of scientific evidence, CLP has pilot tested a variety of activities (e.g., Linn & Songer, 1993; Songer & Linn, 1991). For example, students contrast alternative theories concerning dinosaur extinction, evaluate relevant evidence, and gain insight into the character of competing theories. In another activity, students contrast two views concerning how light travels: light goes forever and light dies out.

To engage students in sustained reasoning, CLP has employed student projects (e.g., Linn & Clark, in press). These projects require students to plan and execute an experiment, critique experiments of others, synthesize results of several experiments, or design instruction on a scientific topic. To help students learn autonomously, guidance is provided by coaches who respond to student queries.

Encouraging autonomous learning while providing a feedback-rich environment has been the goal of several recent research projects. For example, diSessa (1992) has explored the benefits of a computational environment for supporting autonomy, Schoenfeld (1983) has sought to instill self-monitoring, and Chi & Bassok (1989) have identified characteristics of autonomous learners.

Providing Social Support

Learning occurs in a social context as philosophers and educators have asserted for some time (e.g., Dewey, 1901; Dewey, 1929). The social context can facilitate knowledge integration or stand in the way. Vygotsky (1978) has identified the "zone of proximal development" and stressed that with social support students can learn in this zone. Many researchers argue that communities of learners, by distributing knowledge, can solve more complex problems than can be solved alone and can provide the support students need to learn autonomously (Scardamalia & Bereiter, 1993). And, several groups argue that by participating in communities of practice, students can learn the skills of more expert members of the group (Brown & Campione, 1990; Lave & Wenger, 1991; Newman *et al.*, 1989; Pea, 1992).

To provide social support for knowledge integration, programs need to both orchestrate effective social interactions and to guard against interactions that reinforce stereotypes or interfere with knowledge integration (Linn & Burbules, 1993). In the CLP project, our preliminary studies identified impediments to group work. For example, (a) groups of size 3 or 4 excluded members while crowding around the computer screen, (b) males often interrupted and insulted females, (c) natural leaders gained more experience than others, and (d) rather than sharing ideas, students regularly divided the task into parts and each worked alone (Burbules & Linn, 1991; Madhok, 1992 April 4 & 5). To help students build on each other's ideas and benefit from peer input, CLP organized students into groups of two, designed software to support group work, and reengineered activities to foster joint problem solving (Linn, 1992b).

Thus, taking advantage of the social context of learning requires considerable trial and refinement. Students are typically inexperienced in collaborative problem solving and techniques for fostering such learning require further research (e.g., Cohen, 1994; Hawkins *et al.*, 1993; Webb, 1989; Webb & Lewis, 1988).

Overall, the scaffolded knowledge integration framework provides guidance for future curriculum designers. Both to refine the scaffolded knowledge integration framework and to improve instruction, two projects were undertaken. The LISP Knowledge Integration Environment and the spatial reasoning environment are described in the next sections.

THE LISP KNOWLEDGE INTEGRATION ENVIRONMENT (LISP–KIE)

The LISP Knowledge Integration Environment (LISP–KIE) was designed to help undergraduate students learning programming in Common LISP gain an integrated understanding of the domain. Teaching students to solve complex problems in introductory programming classes has proven particularly difficult (Linn, 1985; Linn & Clancy, 1992b). Typically students learn the syntax of the language and the techniques for solving straightforward problems, but are unable to apply their skills to the design and solution of complex programming problems. The goal of the LISP–KIE was to teach introductory programming students to solve complex programming problems by (a) employing case studies to communicate the skills students need, and (b) using LISP, a language that is well-suited to solving a broad array of programming problems.

The first step in designing this instruction was to seek new goals by analyzing how students and experts think about computer programming in general, and LISP in particular. The second step was to design a preliminary curriculum that balanced making thinking visible with encouraging autonomous learning. This preliminary curriculum was based on (a) research on programming, (b) research on cognitive processes, and (c) conjectures from texts designed to teach complex problem-solving. The third step was to implement these ideas and seek ways to provide social support. The resulting approach was studied and refined until a satisfactory design was achieved. The fourth step, synthesizing the results into a framework for future design begins in this paper.

How Do Experts Solve LISP Problems?

Following the CLP approach, the project studied expert models for LISP problem solving. Since LISP has not been extensively studied, the research on Pascal programming was reviewed. In Pascal, programmers chunked programming knowledge into predictable programming patterns and recycled designs from prior programming solutions (Linn & Clancy, 1992a). The approach was different for LISP. Although LISP experts recycled information, there was remarkable variability from one expert to the next concerning what constituted a building block for future programming performance (Linn *et al.*, 1992). This may reflect the fact that most LISP experts are self-taught while Pascal programmers primarily learn from texts that emphasize a common set of patterns. It also reflects the richness of the LISP language. LISP is primarily a functional language but can be used to write functional, object-oriented, or procedural programs. Pascal is a procedural language.

Procedural solutions involve a straightforward application of divide-and-conquer strategies and are widely used. In contrast, the functional way of thinking required for complete understanding of LISP is less intuitive. In functional solutions, one determines a step and uses the result of that step to determine the next step. This application of divide-and-conquer requires keeping track of intermediate results and even experts generally lack intuitive models for these sorts of solutions.

Looking more closely at the skills and heuristics used by expert LISP programmers, Davis et al. (Davis *et al.*, 1993) identified a number of challenges for designing effective LISP instruction. Because LISP is a relatively new instructional language, there is little agreement among expert LISP programmers and among LISP instructors concerning how LISP knowledge should be organized and communicated to students. Two sorts of LISP experts emerged. One group of experts uses models based on applicative operators. Applicative operators are high-level functions that perform complex actions. For example, *reduce* is an applicative operator that applies a function (such as addition or multiplication) to a list. Individuals who rely primarily on applicative operators develop deep understanding of available applicative operators, and also create their own high-level operators that they use in solving most problems. Another group of LISP programmers solve most problems using a recursive model, rather than by using applicative operators. Recursive solutions involve designing functions that then call themselves. Individuals who use recursive solutions have abstract recursive patterns that they reuse. These recursive patterns are more abstract than the patterns that were reused by Pascal novices and experts (Clancy & Linn, 1992; Soloway, 1985).

Thus, two main differences between LISP and Pascal experts emerged. First, although both groups chunk information into programming patterns, Pascal experts generally agree on what the patterns for Pascal ought to be. In contrast, LISP experts generally either focus on recursion, or focus on applicative operators. In addition, those who focus on applicative operators differ among themselves concerning which patterns they reuse, whereas in Pascal there is more consensus. Thus, there are more individual differences among LISP experts than there are among Pascal experts. This reflects, in part, the richness of LISP compared to Pascal.

Second, study of LISP experts also revealed that even LISP experts face difficulties in acquiring the functional thinking required to trace the evaluation of a LISP program (Linn *et al.*, 1992) In contrast, experts in procedural languages are generally adept at tracing most programs (Jeffries *et al.*, 1981). Experts reported difficulty in tracing complex functions, and have problems identifying cases that would be effective for revealing the action of these programs. Both students and experts develop intuitive models for the procedural approach to programming characteristic of Pascal because they are familiar with isolating the steps in a sequence and refining each step. Functional thinking is less familiar.

In consequence, LISP instruction needs to either emphasize some subset of LISP, or support students as they select among alternative models from the start. In addition, students need help in order to develop functional models of programming, no matter which emphasis they learn.

Thus, by studying LISP a number of issues were identified for the design of LISP programming instruction. First, the instructor needs to support the potential diversity of recyclable patterns that students might develop. Rather than imposing a set of patterns on students, it seemed clear that instructors and experts prefer support in developing their own personal set of patterns. These patterns might be targeted to the kinds of problems that the individual expects to solve. They might also be targeted to the experiences and thinking strategies characteristic of the individual. Second, the instructor needs to make the functional thinking necessary for LISP visible. This is a definite challenge for all LISP programmers, and one where techniques for increasing understanding would be valued by experts and novices alike. Third, instruction needs to offer useful models for functional thinking compared to the procedural thinking characteristic of Pascal. Distinguishing specifically on how divide-and-conquer strategies work in functional as opposed to procedural domains would help most learners. Those programmers who have a good model of functional evaluation are likely to be better at identifying chunks of programs that can be solved with straightforward LISP functions. Thus, the new goals for LISP involve supporting students as they develop a personalized repertoire of models, helping students distinguish functional and procedural models, and instilling the value of recycling patterns.

How Do Students Solve LISP Problems?

Analysis of students' acquisition of LISP knowledge revealed several obstacles to learning this programming language. First, many students have difficulty learning about parentheses and quotes in LISP (Davis *et al.*, 1993). A primary explanation for this difficulty is that parentheses and quotes in LISP are used in subtly different ways depending on whether one is defining a function or calling a function that has previously been defined. Since many students attempt to map the superficial characteristics of one example onto another example, students frequently get confused about parentheses and quotes. This is consistent with other research on programming (Adelson & Soloway, 1985; Marco, 1988).

In addition, studies of novices revealed variability in the ability to identify chunks of LISP that could be recycled, and in the ability to abstract LISP knowledge. Students often identified patterns that were either too detailed or too general for use in subsequent problems (Hoadley, 1993; Katz, 1991). Although there was disagreement among experts concerning what constituted appropriate chunks of LISP for recycling, there was no disagreement concerning the need to abstract chunks of LISP and recycle them. In this area, novices have difficulty and instruction could make a difference.

Finally, as they learn LISP, some students rely almost exclusively on their skill in planning problem solutions, postponing learning the details of LISP. Others rely almost exclusively on learning the details or syntax of LISP, and postpone thinking about LISP planning (Schwarz, 1993). When planning and detail knowledge are uncoordinated, knowledge remains disconnected. Thus, effective instruction should help students coordinate details with design skills.

How Do Texts Present LISP?

Comparing textbooks used for teaching Common LISP with textbooks that were effective in teaching Pascal, revealed two important hints for instructional design. First, the case studies designed for Pascal (Clancy & Linn, 1992) offer details on problem solving missing in other texts and especially in LISP texts. The LISP–KIE incorporates the case study approach to make LISP problem-solving decisions visible. Second, Abelson and Sussman's (1985) textbook on Scheme, a dialect of LISP, and other texts described a helpful substitution model for communicating to students the nature of functional thinking. Review of textbooks also revealed considerable differences in the treatment of programming patterns. Some books started by introducing recursion, often neglecting patterns altogether. Others started by introducing applicative operators and varied considerably in the choice of applicative operators and in the emphasis on patterns. In addition, some texts emphasized relatively esoteric applications of applicative operators, while others described only straightforward applications of applicative operators. Thus, for LISP, texts leave most of the knowledge integration to students. The substitution model was one exception and the LISP-KIE capitalized on this model.

Trial and Refinement of LISP–KIE

LISP–KIE was improved as a result of trial and refinement as summarized in a number of publications (Bell *et al.*, in press; Davis *et al.*, in press-a; Davis *et al.*, in press-b; Linn, 1992a; Mann *et al.*, in press). The process focused on the four aspects of the scaffolded knowledge integration environment.

Identifying new goals for LISP students. The research described above suggested that LISP courses emphasize (a) distinguishing function calls from definitions to understand parentheses

and quotes, and (b) developing reusable patterns. Instruction emphasizing effective use of parentheses and quotes was very effective in alleviating students' difficulties, and was refined in subsequent versions of the course (Davis *et al.*, 1993). In the first version of LISP-KIE esoteric uses of applicative operators stood in the way of students understanding LISP and developing reusable patterns (Katz, 1991), so these were eliminated. Assisting students in developing a repertoire of reusable patterns turned out to present remarkable difficulties (Linn, 1992d). This stemmed in part from the idiosyncratic character of patterns developed by experts. Encouraging students to develop a particular set of patterns might inadvertently stifle the ability to design a personally appropriate set of patterns in the future. Instead, supporting students as they struggle to understand LISP evaluation and create their own set of patterns proved effective (Linn, 1992d; Mann *et al.*, in press).

Making functional thinking visible. In order to communicate the LISP evaluation model to students, LISP–KIE featured the LISP Evaluation Modeler (LEM), an on-line tool that makes the substitution approach visible. This tool traces the evaluation of any LISP function step-by-step (see Figure 2). Instead of trying to do this by hand, programmers can have the computer do it. LEM reduces the processing demands of tracing by hand where intermediate results need to be kept in mind. Both experts and novices find this tool to be extremely valuable. Recall that both experts and students have difficulty with functional thinking. Experts and students alike enjoy using the LISP Evaluation Modeler to determine how complex functions will be evaluated. Both groups find that LEM helps them to debug programs, and to refine their designs (Mann *et al.*, in press).

An aspect of programming instruction that perplexes both those teaching procedural languages and those teaching functional languages concerns making skill in complex problem solving visible. The skills need to be explained and tailored to personal problem solving practices The Pascal case studies (Linn & Clancy, 1992a) are effective in making design skills visible. To impart these same skills for LISP, the LISP–KIE featured LISP case studies. The LISP case studies were available both in text format, and in an interactive format (Bell *et al.*, in press). The interactive format was designed to provide additional support for students as they learned the problem-solving skills emphasized in the case studies. One weakness of the paper case studies available for Pascal was that students often were lulled into the "illusion of comprehension," and failed to solve problems using the problem-solving process described in the case study. Interactive Case Studies overcome this difficulty by asking students to engage in reflection and prediction, and providing feedback on these activities in the context of complex problem solutions (Bell *et al.*, in press). For example, students using the interactive LISP case studies (a) predicted how a particular aspect of a problem might be solved, (b) explored how the problem was solved by the expert, and (c) reconciled their prediction with the expert's solution. Thus, the LISP case studies added interactive feedback to an instructional technique that had already proven effective in Pascal.

Encouraging autonomous programming. Programming students often need specific encouragement to critique problem solutions and make connections among ideas. Many students assume their programs will always be correct. Such a view prevents students from learning to analyze and criticize their own programs. Instruction on parenthesis and quotes emphasized critiquing alternative solutions (Davis *et al.*, in press-b). Often students who were able to solve problems could not critique solutions generated by others. To facilitate skill as a critic, the LISP–KIE also featured CodeProbe as a part of case studies. CodeProbe supports students' debugging activities, and encourages effective trouble-shooting practices by helping students generate a full set of test cases for their programs and by requiring students to predict outcomes for specific cases (Bell *et al.*, in press) (see Figure 3).

Providing social support in computer labs. The LISP-KIE featured groups of two students solving problems in scheduled computer laboratories. Students were guided by teaching

assistants and the instructor. To improve social interactions, several alternative configurations of the computer laboratory were investigated. Observation indicated that when students worked in pairs but could easily consult another pair they were most likely to benefit from peer explanations. Problems that encouraged students to compare designs rather than divide and conquer took more advantage of the social context of learning. Students relied far more on the teaching assistant than on peers for explanations and preferred to work alone on complex problems but to consult peers for help with the computer software. Capitalizing on social aspects of learning programming requires further work.

Evaluation of LISP–KIE

Overall, how did the revisions improve LISP–KIE? Two main results emerged. First, students' understanding of LISP became increasingly more sophisticated as the course was refined (Bell *et al.*, in press). The LISP Evaluation Modeler, the CodeProbe troubleshooting tool, and the Interactive Case Studies jointly contributed to improve student understanding. This was particularly noticeable in the projects that students chose for their final course assignment. In early versions of the course the most difficult projects selected were relatively straightforward problems, such as writing a simple ELIZA or "doctor" program. This program repeated input in a slightly modified format. In the last version of the course students selected a relatively complex database program as their final project.

A second major benefit of the LISP–KIE was the support it provided for students with limited prior computer experience, and especially women. In early versions of the course women compared to men earned lower grades and were more likely to drop the course. Women also reported less prior computer experience. Since even in science and education, females typically earn higher grades than males, this pattern indicated that the course might benefit from refinement. In the final version of the course women and men earned equal grades, and women persisted as frequently as men (see Figure 4 and Table I). This change in persistence and success reflects greater support for all students and especially those with limited computer experience. For example, difficulties in learning parentheses and quotes in the first versions of the course derailed students early in the term. By making parentheses and quotes less formidable, students could focus on other issues. Simplifying the uses of applicative operators also enabled students to succeed, independent of prior computer experience.

SPATIAL REASONING ENVIRONMENT

Compared to the LISP–KIE, the spatial reasoning environment had more modest goals involving an aspect of an engineering course in graphical communications, namely, spatial reasoning (Hsi & Linn, 1994). Many believe that spatial reasoning is an important skill in engineering. For example, when designing new artifacts, engineers imagine parts and objects that do not yet exist, and visualize how these parts will interact with other imagined parts. In evaluating a design for an artifact, engineers visualize how the design will look when it is implemented. Often, two-dimensional drawings need to be imagined in three dimensions.

Spatial reasoning is frequently cited as a barrier to the participation of women in engineering, and as a difficulty faced by many engineering students (Agogino & Linn, 1992 May-June; Shepard & Metzler, 1971). Although spatial reasoning is important in engineering, it is rarely taught in the pre-college curriculum. Many engineering students enter college programs with limited spatial reasoning experience (Newcombe, 1981). Furthermore, research suggests that spatial reasoning improves with practice, so students given experience in spatial reasoning improve their spatial reasoning skills (Lohman, 1988).

A primary motivation for the spatial reasoning environment was the conjecture that lack of spatial reasoning experience might discourage some students from persisting in engineering who would otherwise be successful. In particular, if spatial reasoning skill reflects spatial reasoning experience, then students lacking the experience might be discouraged from persisting (Newcombe, 1981). An intervention to help students develop skill in solving spatial problems might reverse this trend.

To develop spatial reasoning instruction for engineering students, the same process as described for the LISP–KIE was followed. First, experts' and students' spatial reasoning processes were analyzed. Then preliminary instruction based on prior research and analyses of expert and student understanding, was designed and tested. Finally, the design and testing process was iterated until a satisfactory intervention emerged. As in the LISP–KIE, several technological tools were designed to provide spatial reasoning experience. The final synthesis of these experiences also suggests improvements to the scaffolded knowledge integration framework.

How Do Experts Solve Spatial Reasoning Problems?

The skills of expert spatial reasoners have not been extensively studied. Thus, a group of engineering instructors and professional designers were asked to solve a series of spatial reasoning problems while describing their approach (Bell & Linn, 1993). Problems were selected from common inventories designed to measure spatial ability and from engineering tasks likely to be encountered in the workplace.

Consistent with the LISP–KIE results, there were substantial individual differences among expert engineers with regard to their spatial reasoning ability. Most experts relied on a repertoire of strategies for performing spatial tasks. These included a "holistic" strategy where an object was rotated in its entirety, a "pattern" strategy where familiar parts of the object were rotated and then connected together, and an "analytic" strategy where details of an object, such as the lengths of lines or size of angles between lines, were used to rotate the object. Experts sometimes used a set of heuristics to decide which of these strategies was appropriate, and sometimes switched between strategies after they started on a problem solution. In addition, experts differed with regard to their propensity for using one strategy or another. Some experts always preferred a holistic strategy and only used analytic or pattern strategies when the holistic strategy failed. Others generally used analytic strategies, sometimes using techniques from descriptive geometry, and only rarely used holistic approaches. Some relied on a rich repertoire of patterns and quickly identified these patterns in complex objects, resorting to holistic and analytic strategies only rarely.

In addition, when asked, experts varied considerably in their beliefs about the role of spatial reasoning in engineering. Many said that although spatial reasoning seemed to contribute to many activities, it was rarely used in isolation. That is, when analyzing plans or creating a design, they drew on spatial reasoning skills in conjunction with other skills. Some reported that spatial reasoning was less important than students believed.

Another parallel between expert engineers and expert LISP programmers is that both groups had developed their skills primarily without instruction. Many engineers reported that they had never encountered any instruction in spatial reasoning nor remembered receiving any training or guidance. The experts found discussing spatial reasoning unusual. Most had difficulty verbalizing how they solved spatial reasoning problems. Some indicated that they viewed themselves as poor spatial reasoners. For example, a female engineering professor commented that she had never been able to use a holistic reasoning strategy. This perceived weakness influenced her choice of engineering specialization and still worries her today. She was surprised to learn that many expert engineers, including mechanical engineers, reported the same

difficulties. She had avoided mechanical engineering because she found holistic reasoning difficult. None of the engineers and designers interviewed reported ever discussing spatial reasoning with others.

How Do Students Solve Spatial Reasoning Problems?

To determine students' attitudes and approaches to spatial reasoning, we interviewed students in an introductory engineering course. In general, students had little insight into spatial reasoning, although most believed that it was something that would contribute to success in engineering. Students, like experts, relied on analytical, pattern and holistic approaches to spatial reasoning. Like the engineering professor who worried about her spatial reasoning skill, many students worried that their skills would hinder their success in the graphical communication course and possibly in their careers. Unlike experts, students could not verbalize any heuristics concerning when to use one spatial strategy and when to use another.

Research on spatial reasoning suggests that some students rely on holistic strategies while others rely on analytical strategies as a matter of preference (Kail *et al.*, 1979; Shepard & Metzler, 1971). Furthermore, this research suggests that holistic strategies are more efficient than analytic strategies in that students using holistic strategies reach answers sooner. This difference in strategy selection is a factor in the gender differences frequently reported for spatial reasoning. Males are more likely to rely on a holistic strategy, and females are more likely to rely on an analytic strategy in solving spatial reasoning items when they are presented on spatial reasoning assessments (Linn & Petersen, 1985). Holistic reasoners perform items faster and get higher scores.

Thus, both experts and students vary in the strategies that they use for spatial reasoning. Neither group has received much instruction in spatial reasoning. Both groups believe that spatial reasoning is part of engineering although the experts see it as less central than the students enrolled in graphical communication.

Spatial Reasoning Instruction

The most common instruction that has been offered in spatial reasoning is providing students with hints and experience solving spatial problems. A number of studies report the efficacy of this form of instruction (Connor & Serbin, 1980; Connor & Serbin, 1985; Lohman, 1988).

Design and Refinement of the Spatial Reasoning Environment

Based on studies of students and experts, and analysis of prior instruction, the project defined new goals for spatial reasoning, emphasizing instruction that helps students develop a repertoire of holistic, pattern-based, and analytic strategies. Helping students develop heuristics to guide selection of an appropriate strategy was desired. Finally, providing students more experience with spatial reasoning, inspired by other successful programs, was emphasized.

Trial and refinement in the spatial reasoning environment focused on (a) making spatial reasoning strategies visible, (b) providing a dynamic model linking the alternative strategies, (c) troubleshooting spatial problem solutions, and (d) solving spatial reasoning problems.

Identifying new goals for spatial problem-solving Research on expert spatial reasoning supported the repertoire of strategies approach to spatial reasoning and delineated three predominant strategies. How can these be taught? What experiences help students develop a

repertoire of spatial reasoning strategies? As shown in Figures 5, 6, and 7, a common strategy in solving spatial reasoning items in mechanical engineering is to rotate an object to a standard orientation, and then to look for familiar patterns. For example, a chair, when turned to a standard orientation, appears to be composed of an angle bracket with legs. By rotating a chair to this orientation, experts can easily recognize familiar patterns. This strategy or proceduralization could also help students.

Another common strategy is to visualize holistic rotations of objects. Experience with solid objects and examining holistic rotations might help students gain familiarity with the kinds of transformations that arise when objects are holistically manipulated, and therefore develop skill in performing holistic manipulations themselves.

Another strategy is analytic: generating orthographic views from an isometric 3D view. This involves measuring angles or lengths of objects and matching faces or views. Students are usually taught analytic strategies for rotating and comparing aspects of an object.

Making spatial thinking visible. A computer tool called *Display Object* (Osborn & Agogino, 1992) was created to support all three approaches to spatial reasoning. It permits the rotation of three-dimensional objects, the positioning of objects in familiar orientations, and the analysis of objects.

Display Object supports understanding of pattern-based, analytic, and holistic spatial reasoning strategies. Students or instructors can add objects and patterns to the library. The software allows students to essentially rotate objects, to rotate patterns from objects, and to snap objects to common rotations. The Display Object tool plays a similar role to that of the LISP Evaluation Modeler. Both are designed to support complex reasoning strategies by reducing processing demands and both help students link their problem-solving approaches.

Encouraging autonomous learning of spatial reasoning. Both a laboratory and a spatial-tutoring season were designed to support independent spatial reasoning during problem solving. In these sessions, students solved a series of spatial reasoning problems independently and then used Display Object to solve the problems. They compared the solutions and reflected on their own reasoning processes. Similar to the instruction featuring the LISP evaluation modeler in the LISP–KIE, students made predictions concerning an object's appearance in different rotations, tested their prediction against the outcome using Display Object, and analyzed the results

Paralleling the approach to teaching autonomous learning in LISP–KIE, the spatial reasoning environment also provided experience in troubleshooting and debugging. Since there are some limits to spatial representations when three-dimensional objects are depicted in two dimensions, Display Object was used to create two-dimensional depictions that describe an ambiguous image of the three-dimensional object. To help students develop trouble-shooting skills and debug spatial situations, students carried out a series of activities with ambiguous objects to detect when two-dimensional information was sufficient to determine the characteristics of a three-dimensional object.

The spatial-tutoring session provided additional practice for students who encountered difficulties. Using a pre-test as a screening device, students at risk for failure were identified. These students were invited to the tutoring session where they could use Display Object, as well as discuss and contrast spatial reasoning strategies.

Providing social support for engineering students. Discussion during the spatial-tutoring session provided social support for a repertoire of strategies. It was included in order to help students recognize the value of using alternative strategies for solving spatial reasoning problems. Students learned to distinguish among these strategies in the session. The discussion also

empowered students by legitimizing use of a repertoire of strategies. Often in these discussions, students reported that they thought that they were "cheating" or somehow using an inferior strategy when they resorted to analytic or pattern techniques rather than relying on holistic techniques. These discussions echoed experiences from the expert interviews. By providing these tutoring sessions and allowing students to discuss with each other their strategies for spatial reasoning, the project helped students learn to value their own approaches rather than assuming that another approach is better.

Capitalizing on the social nature of learning is an important goal of the scaffolded knowledge integration framework. In engineering, social aspects of learning are particularly important because of the expectations that students have about each other. Most engineering students expect that female students are less likely to have spatial reasoning skill. Many female students report being frustrated by expectations that their fellow students have about them (Hsi, 1992 Oct 23-25). These expectations come from what might be referred to as "The Tyranny of the Mean" (Linn, 1994), in that individuals when reasoning about their peers, generalize from population differences. Since more men than women are in the field of engineering, students often assume that men are more likely than women to succeed in engineering. They may extend this general view to a stereotyped expectation about individual women in engineering programs. Instead, among students entering engineering, the success ratio for men and women is similar (Seymour & Hewitt, 1994).

These stereotyped perceptions can result in deleterious effects of social interactions. This is particularly well-illustrated in one of the group learning situations that arose in the spatial reasoning environment (Agogino & Linn, 1992 May-June). In one group, diary records of individual members revealed that male students believed that the single female member of their group was unable to solve the problem. The female group member, in turn, developed the perception that her contributions, which were accurate, were simply not valued by other members of the group. Thus, the social process of the group derailed the intellectual process, resulting in dissatisfaction on all sides. The male students were dissatisfied because one member of the group did not contribute, and the female student was dissatisfied because her contributions, although useful, were ignored.

Emphasizing the social nature of learning can help students identify strategies used by others. Students may appropriate such strategies for their own repertoire. Thus, the tutoring session for spatial reasoning that emphasized the sharing of reasoning strategies among students took advantage of the social nature of learning by allowing students to learn strategies used by others. Yet these strategies must be used judiciously, since they can reinforce stereotypes.

Evaluation of the spatial reasoning environment

The Spatial reasoning environment was successful as indicated by several measures. Students reported enjoying the opportunity to engage in spatial reasoning. In addition, most students reported that they became more proficient and more aware of their own spatial reasoning processes as a result of the laboratory activities. Those students who participated in the tutoring session also reported that they enjoyed the session and felt that it contributed to their success in the course. Far more females than males attended the tutoring session, and the females were particularly appreciative of the opportunity to address spatial reasoning strategies. Furthermore, relative to males, females learned more spatial reasoning skills from the tutoring (see Table II and Figure 8) (Hsi & Linn, 1994).

Examination of course progress and spatial reasoning skill revealed some interesting patterns. The focus of this course was graphical communication, and a large part of the course was the conduct of the design projects where students determined the solution to a problem and rendered

their solution using a computer assisted design package. Thus, one might anticipate that spatial reasoning skill would contribute to success in the course.

In fact, results revealed no relationship between spatial reasoning skill and success on the midterm, final, or course grade (Hsi & Linn, 1994). One explanation for this lack of relationship concerns the strong emphasis on analytical techniques characteristic of the course. Students could succeed in the course without using a holistic strategy at any point. However, there were opportunities to use a holistic strategy if the students desired to do so.

Finally, comparing the Spatial reasoning environment version of the graphical communication course to the previous version of the course demonstrated several benefits for the spatial reasoning environment. First, the number of students who dropped the course declined substantially. Very few students dropped the Spatial reasoning environment course, compared to prior versions of the course taught by the same instructor. Second, grades and test scores for males and females were similar in the spatial reasoning course while males were more successful than females in previous versions of the course (Figure 8). In general, females earn better grades than males in engineering so the project changes brought this course in line with the others (Kimball, 1989; Linn, 1992c; Linn & Kessel, in press).

In summary, following the trial and refinement process reveals parallels between LISP-KIE and the spatial reasoning environment. In both cases, experts used a repertoire of strategies rather than preferring a single approach to problem solving. In both cases, visualization tools helped students learn the strategies used by experts. Indeed, in both LISP and spatial reasoning, experts who lacked insight into the problem-solving process were found, and in the case of engineering, one female engineering professor had made career decisions based on her less-than-complete understanding of the spatial reasoning processes used by other expert engineers. In addition, in both cases, the intervention that was most effective involved clarifying aspects of problem solving that might be undercommunicated in normal courses. Case studies, for example, communicate problem-solving processes that might be neglected in a typical course focused on outcomes, and the tutoring sessions communicated individual differences in problem-solving in a way that might be neglected in a typical engineering course.

THE SCAFFOLDED KNOWLEDGE INTEGRATION FRAMEWORK REVISITED

Studies of the spatial reasoning environment and LISP-KIE reinforce many of the ideas in the scaffolded knowledge integration framework and refine others. Many parallels and commonalities in these two innovations to enhance learning and instruction in computer programming and engineering illustrate the advantages of this framework.

Providing New Goals for Learning, Revisited

Both the spatial reasoning environment and LISP-KIE sought new goals for learning based on analysis of expert and novice behavior. Studies of these environments reinforced the idea that experts use a repertoire of strategies and demonstrated the advantage of building on ideas students found familiar. They also argued against emphasizing strategies that students could readily integrate with their intuitive ideas. For example, when esoteric applications of applicative operators were emphasized, many students become perplexed, and failed to integrate their ideas. Instead, they tried to memorize some exemplary problems, in hopes that these examples would map onto future problems. Changing the LISP–KIE course so that only straightforward applications of applicative operators were emphasized, actually resulted in students gaining more sophisticated understanding of LISP than was the case when complex characteristics of these applicative operators were emphasized (Katz, 1991). This finding echoes

the CLP finding that heat flow is more powerful than molecular kinetic theory for students who, at first, cannot distinguish heat and temperature.

Both the LISP Evaluation Modeler in LISP–KIE, and Display Object in the spatial reasoning environment encourage knowledge integration and cohesion because they support the full repertoire of strategies characteristic of experts. Both also allow students to select the models they prefer, rather than imposing models on students.

Choice of the LISP Evaluation Modeler rather than a machine-level model of LISP evaluation supports both recursive and applicative operator-based patterns, and provides a firm foundation for subsequent courses. The LISP Evaluation Modeler provides a straightforward model of LISP evaluation. It is easy to understand, and it applies reasonably well to both recursive and applicative operator problems. All the problems that students encounter during the introductory course, and many problems that students solve throughout their programming careers can be studied using the LISP Evaluation Modeler. Both experts and students find the modeler effective (Mann *et al.*, in press).

Similarly, Display Object connects the pattern, analytic, and holistic strategies. Thus, Display Object has a unique focus in helping students develop a repertoire of strategies and learn when to use these diverse approaches to spatial reasoning. This project extends the scaffolded knowledge integration framework by illustrating why multiple strategies are all effective and helping students distinguish appropriate circumstances for each.

Do All Experts Have the Same Repertoire of Models?

These studies illustrated that experts have diverse models for solving complex problems. There is controversy in the cognitive community concerning whether experts have uniform understanding of their domains, or whether there are individual differences in expert understanding. Thus, Chi, Feltovich and Glaser (1981) suggested that experts had fairly uniform ways of organizing physics knowledge. Reif and Larkin (1991) demonstrated that, in solving mechanics problems, experts tended to use free body diagrams, and novices tended to iterate on available formulas. In both of these studies, however, experts were asked to solve problems that were appropriate for novices. In addition, most of the experts studied were instructors. Instructors might choose similar strategies for communicating to novices about the problems that are commonly assigned to novices, yet use different methods when solving complex problems themselves. Thus, studies of experts solving complex problems reveal considerably more diversity (Clement, 1991; Nersessian, 1991).

The Linn et al. (1992) study of LISP experts asked for solutions to problems that were genuinely challenging to the experts. Both instructors and experts in the workplace were studied. Thus, the diversity among experts identified in these studies may reflect that (a) experts were solving problems that were challenging to them, (b) experts were involved in diverse activities, and (c) experts were using autonomously-generated patterns for problem solving. Just as we would not expect a group of experts in English literature to write similar essays on, for example, “Jane Austen’s feminism,” so would we expect diversity in solving complex programming and engineering problems.

Making Thinking Visible, Revisited

Often, students need assistance in integrating their ideas, and a dynamic model can be particularly effective in fostering integration. Both the LISP Evaluation Modeler and the Display Object tool make the use and selection of problem solving strategies more visible for students.

The LISP Evaluation Modeler is so successful that it is valued by both experts and students. The substitution model is a straightforward model; however, it is difficult to apply. Tracing a program using this model often overloads the processing capacity of both experts and novices. By making thinking visible with the LISP Evaluation Modeler, the LISP–KIE fosters knowledge integration.

Similarly, Display Object helps students keep track of information that they themselves might have difficulty with, by rotating an object as well as displaying the object from all possible orientations. Whereas students often cannot perform this kind of holistic rotation themselves, the opportunity to observe holistic rotation is effective in helping students distinguish their analytic strategies from their holistic strategies, and in fostering a holistic strategy for straightforward objects. Students in the tutoring session often reported that they had been reluctant to try a holistic strategy, and that Display Object empowered them to reason holistically and then check their expectations out against the results from Display Object. Thus, Display Object makes holistic thinking visible and supports students in developing a repertoire of spatial reasoning strategies.

Case studies make thinking visible by going beyond the outcome of the problem-solving process to describe the steps necessary to complete the process. Often students fail to gain feedback or information about the process of problem-solving. Interactive case studies as implemented in LISP–KIE are more effective than paper-based case studies because the guidance that students might desire as they go through the process of problem-solving strategies is more immediate and available.

Tutoring, as emphasized in the spatial reasoning environment, has been hailed by many as the most effective way to provide the guidance and explicit problem solving processes necessary for effective learning (Bloom, 1984). In both of these projects, providing guidance and explicit problem solving assistance was particularly beneficial for students who entered the course with less experience than their peers. Guidance, for example, in solving problems involving parentheses and quotes was particularly useful for students as they began the LISP course. Guidance for students concerning the diversity of strategies appropriate for solving spatial reasoning problems was equally beneficial for students as they began the spatial reasoning course.

Encouraging Autonomous Learning, Revisited

These investigations highlight the importance of encouraging self-monitoring and autonomous learning to make students responsible for their own knowledge integration. The investigations for both LISP–KIE and spatial reasoning revealed that experts in these areas varied substantially from one to another yet all exhibit skill in generating and critiquing ideas and in monitoring progress. This variability in expertise means that in order to make sense of alternative strategies students need to develop self-monitoring skills. Students have an opportunity to select strategies for problem-solving in these complex domains that are most suited to their own evolving knowledge structures. Furthermore students need to develop some heuristics for selecting among strategies depending on the nature of the problem. Both of these skills are aspects of self-monitoring. In order to encourage self-monitoring, both LISP–KIE and spatial reasoning engage students in reflecting on their own problem solving. They both also help students recognize the diversity in problem-solving strategies that are available.

Both CodeProbe and the ambiguous objects presented with Display Object encourage students to take the role of investigator and critic in the process of learning complex material. Taking both these roles illustrates the process of self-monitoring and helps students see the power of strategies that they might otherwise ignore.

Both spatial reasoning and LISP–KIE also help students understand the epistemological characteristics of their fields. Thus, rather than assuming that there is a single right answer or a single best path for solving a problem, both of these approaches emphasize the diversity of methods and paths to solution, the importance of contrasting alternatives, and the value of considering different methods for solving a problem. Ultimately, students who recognize several reasonable explanations for phenomena are likely to expand their repertoire of models of scientific explanation from a view that there is a single correct answer to a view that explanations have strengths and limitations.

Both LISP–KIE and the spatial reasoning environment encourage students to compare their own approaches to those of experts in case studies and in tutoring situations. These are activities designed to encourage students to reflect on their own learning process and develop increased skill in monitoring their problem-solving activities.

One of the most crucial activities for experts solving complex and difficult problems is monitoring progress and allocating time effectively. Some LISP students develop uncoordinated understandings, emphasizing planning or emphasizing discipline-specific information. These students clearly need assistance in self-monitoring. Although this problem has been studied in LISP-KIE, devising solutions to it remains challenging.

Providing Social Support, Revisited

Both LISP-KIE and the Spatial Reasoning Environment sought to capitalize on the social nature of learning by supporting group learning in laboratory and tutoring sessions. In both projects these experiences has strengths and drawbacks. Working in groups often helped students expand their repertoire of strategies or refine understanding of a familiar strategy. Often students expressed strategies in new ways so that other students could understand them for the first time. At the same time, group members also discouraged their peers by criticizing on the basis of normative views. For example, in both projects, females sometimes experienced disrespectful and unwarranted treatment based on the normative view that computer science and engineering are male domains.

Capitalizing on the social nature of learning requires careful analysis of the learning environment. Normative views, based on the population of males and females and of underrepresented minorities can mitigate against benefits gained from appropriating problem solving practices refined by others. This aspect of the framework requires further investigation.

IMPLICATIONS

The scaffolded knowledge integration framework builds on the work of the Computer as Learning Partner project (Linn, in press). Over 10 years, the CLP project refined an approach to teaching heat, temperature, light, and sound in middle school. The projects described here use these ideas to refine university courses in programming and spatial reasoning. This framework is also consistent with a number of recent research investigations that focus on learning in complex, cumulative domains (Collins *et al.*, 1991; Spiro & Jehng, 1990; Spiro *et al.*, 1987; Torney-Purta, 1993 Aug 22). In general, the scaffolded knowledge integration approach applies to both precollege and university courses in complex domains.

Studying learning and instruction in these three projects strongly reinforces the view of the learner as developing and refining a repertoire of models described at the onset. Studies of experts in all three domains revealed that they used a repertoire of strategies, selecting approaches matched to individual problems. Helping students expand their repertoire and

distinguish among strategies seems remarkably promising. Experts all use strategies that vary in sophistication and even in accuracy. Tradeoffs between efficiency and accuracy are common, and the meta-cognitive skills necessary to achieve these tradeoffs are important for expert problem-solving. This principle guided the development of new goals for learning in every project.

To integrate this repertoire, the framework emphasizes making problem solving strategies explicit while at the same time placing responsibility on the student to connect and refine this knowledge. Students may resist taking responsibility for their own learning, especially if instructional conditions fail to emphasize autonomous learning. Some students resist even when encouraged and remark that memorizing has always worked in the past. Changing students' dispositions to integrate and refine their ideas is key to developing and distinguishing a repertoire of models, but may prove difficult.

Taking advantage of the social nature of learning can support students who may resist autonomous learning. The social context of a community of scholars can help change the disposition to integrate by supporting newcomers as they experiment with linking and connecting ideas. Creating productive communities, especially computer-supported communities, offers a challenge for future researchers.

The scaffolded knowledge integration framework suggests promising principles, conjectures, rules-of-thumb, and heuristics for design of complex instruction. By testing and refining these ideas, design teams have the opportunity to build a repertoire of instructional strategies and to delineate the conditions under which they apply.

REFERENCES

Abelson, H. & Sussman, G. (1985). *Structure and interpretation of computer programs*. Cambridge: MIT Press.

Adelson, B. & Soloway, E. (1985). The role of domain experience in software design. *IEEE Transactions on Software Engineering, SE*(11), 1351-1360.

Agogino, A. M. & Linn, M. C. (1992 May-June). Retaining female engineering students: Will early design experiences help? [Viewpoint Editorial]. In Wilson, M. (Ed.) *NSF Directions, 5*(2), p. 8-9.

Anderson, J. R. (1993). Problem solving and learning. *American Psychologist, 48*(1), 35-44.

Bell, J. E. & Linn, M. C. (1993). *Scaffolding novices in spatial reasoning and visualization for engineering.* Working paper. Berkeley, CA: University of California at Berkeley, School of Education.

Bell, J. E., Linn, M. C., & Clancy, M. J. (in press). Knowledge integration in introductory programming: CodeProbe and interactive case studies. In Soloway, E. (Ed.), *Interactive learning environments*.

Beshears, F. (1990). LabView forum introduces software for real-time data. *Instructional Technology Program Newsletter* (A publication of the Instructional Technology Program, University of California, Berkeley), *3*(1), 3.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*, 4-16.

Brown, A. L. & Campione, J. C. (1990). Communities of learning and thinking, or A context by any other name. *Contributions to Human Development, 21*, 108-126.

Burbules, N. C. & Linn, M. C. (1991). Science education and the philosophy of science: Congruence or contradiction? *International Journal of Science Education, 13*(3), 227-241.

Carey, S. (1985). *Conceptual change in childhood*. Cambridge, MA: MIT Press.

Champagne, A. B., Klopfer, L. E., & Gunstone, R. F. (1982). Cognitive research and the design of science instruction. *Educational Psychologist, 17*(1), 31-53.

Chi, M. T. H. & Bassok, M. (1989). Learning from examples via self-explanations. In Resnick, L. B. (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 251-282). Hillsdale, NJ: Lawrence Erlbaum Associates.

Chi, M. T. H., Feltovich, P., & Smith, E. L. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5*(2), 121-152.

Clancy, M. J. & Linn, M. C. (1992). *Designing Pascal solutions: A case study approach* (1st ed.), (Principles of Computer Science) (Aho, A. V. & Ullman, J. D., Series Eds.). New York, NY: W. H. Freeman and Company.

Clement, J. (1991). Non-formal reasoning in science: The use of analogies, extreme cases, and physical intuition. In Voss, J. F., Perkins, D. N., & Siegel, J. (Eds.), *Informal reasoning and education*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Cohen, E. G. (1994). Restructuring the classroom: Conditions for productive small groups. *Review of Educational Research, 64*(1), 1-35.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, 15*(3), 6-11, 38-39.

Connor, J. M. & Serbin, L. A. (1980). *Mathematics, visual-spatial ability, and sex roles.* Binghamton, NY: State University of New York.

Connor, J. M. & Serbin, L. A. (1985). Visual-spatial skill: Is it important for mathematics? Can it be taught? In Chipman, S. F., Brush, L. R., & Wilson, D. M. (Eds.), *Women and mathematics: Balancing the equation* (pp. 151-174). Hillsdale, NJ: Lawrence Erlbaum Associates.

Davis, E. A., Linn, M. C., & Clancy, M. J. (in press-a). Learning to use parentheses and quotes in LISP. *Computer Science Education*.

Davis, E. A., Linn, M. C., & Clancy, M. J. (in press-b). Students' off-line and on-line experiences. *Journal of Educational Computing Research*.

Davis, E. A., Linn, M. C., Mann, L. M., & Clancy, M. J. (1993). Mind your Ps and Qs: Using parentheses and quotes in LISP. In Cook, C. R., Scholtz, J. C., & Spohrer, J. C. (Eds.), *Empirical Studies of Programmers: Fifth Workshop* [paper presented at the Fifth Workshop on Empirical Studies of Programmers, Palo Alto, CA] (pp. 62-85). Norwood, NJ: Ablex.

Dewey, J. (1901). *Psychology and social practice*, (Contributions to education). Chicago, IL: University of Chicago Press.

Dewey, J. (1929). *The sources of a science of education*. New York: Horace Liveright.

diSessa, A. (1988). Knowledge in pieces. In Forman, G. & Pufall, P. (Eds.), *Constructivism in the computer age* (pp. 49-70). Hillsdale, NJ: Lawrence Erlbaum Associates.

diSessa, A. (1992). Images of learning. In De Corte, E., Linn, M. C., Mandl, H., & Verschaffel, L. (Eds.), *Computer-based learning environments and problem solving.* Berlin: Springer-Verlag.

Duschl, R. A. (1990). *Restructuring science education: The importance of theories and their development*. New York: Teacher's College Press.

Eylon, B. & Linn, M. C. (in press). Models and integration activities in science education. In Bar–On, E., Scherz, Z., & Eylon, B. (Eds.), *Designing intelligent learning environments*. Norwood, NJ: Ablex Publishing Corporation.

Hawkins, J., Frederiksen, J., Collins, A., Bennett, D., & Collins, E. (1993). Assessment and technology. *Communications of the ACM, 36*(5), 74-76.

Hoadley, C. M. (1993). *Functional abstraction and code reuse in LISP*. [Technical Report]. Berkeley, CA: University of California, HyperMedia Case Studies.

Hsi, S. (1992 Oct 23-25). Beliefs about learning and persistence in engineering: A study of freshman women. Paper presented at the *AAUW National Educational Equity Conference,* Mills College.

Hsi, S. H. & Linn, M. C. (1994). *Spatial reasoning and success in engineering.* Working paper. Berkeley, CA: University of California at Berkeley, School of Education.

Hutchins, E. (in press). *Cognition in the wild*. Cambridge, MA: MIT Press.

Inhelder, B. & Piaget, J. (1969). *The early growth of logic in the child.* New York: Norton.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E. (1981). The processes involved in designing software. In Anderson, J. R. (Ed.), *Cognitive skills and their acquisition* (pp. 255-283). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kail, R., Carter, P., & Pellegrino, J. (1979). The locus of sex differences in spatial ability. *Perception and Psychophysics, 26*, 182-186.

Katz, M. A. (1991). *The role of applicative operators in beginning LISP programming*. Unpublished master's thesis, University of California, Berkeley, CA.

Kimball, M. M. (1989). A new perspective on women's math achievement. *Psychological Bulletin, 105*(2), 198-214.

Kuhn, D., Amsel, E., O'Loughlin, M., & with the assistance of Schauble, L. (1988). *The development of scientific thinking skills,* (Developmental Psychology Series). Orlando, FL: Academic Press.

Kuo, F. (1988). Learning science by modelling. *Instructional Technology Program Newsletter (A publication of the Instructional Technology Program, University of California, Berkeley), 1*(1), 2.

Lan, H. (1989a). Computer-aided engineering. *Instructional Technology Program Newsletter (A publication of the Instructional Technology Program, University of California, Berkeley), 2*(4), 3.

Lan, H. (1989b). Engineering design courses to be revitalized. *Instructional Technology Program Newsletter (A publication of the Instructional Technology Program, University of California, Berkeley), 2*(3), 4.

Larkin, J. H. & Reif, F. (1979). Understanding and teaching problem solving in physics. *European Journal of Science Education, 1*, 191-203.

Lave, J. & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge, MA: Cambridge University Press.

Lewis, E. L. & Linn, M. C. (1994). Heat energy and temperature concepts of adolescents, adults, and experts: Implications for curricular improvements. *Journal of Research in Science Teaching, 31*(6), 657-677.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher, 14*(5), 14-16, 25-29.

Linn, M. C. (1989). Models for computer-based student laboratories. [Reprinted from "Dear Colleagues" in the Instructional Technology Program Newsletter (A publication of the Instructional Technology Program, University of California, Berkeley), Vol. 2, No. 2]. *Technology and Learning*, p. 11.

Linn, M. C. (1992a). The art of multimedia and the state of education [Review of Nix, R. & Spiro, R. (Eds.) (1990). Cognition, education, and multimedia: Exploring ideas in high technology. Hillsdale, NJ: Lawrence Erlbaum Associates]. *Educational Researcher, 21*(1), 30-32.

Linn, M. C. (1992b). The computer as learning partner: Can computer tools teach science? In Sheingold, K., Roberts, L. G., & Malcolm, S. M. (Eds.), *This year in school science 1991: Technology for teaching and learning*. Washington, DC: American Association for the Advancement of Science.

Linn, M. C. (1992c). Gender differences in educational achievement. In Pfleiderer, J. (Ed.), *Sex equity in educational opportunity, achievement, and testing [Proceedings of 1991 Educational Testing Service Invitational Conference]* (pp. 11-50). Princeton, NJ: Educational Testing Service.

Linn, M. C. (1992d). How can hypermedia tools help teach programming? *Learning and Instruction, 2*, 119-139.

Linn, M. C. (1994). The tyranny of the mean: Gender and expectations. *Notices of the American Mathematical Society, 41*(7), 766-769.

Linn, M. C. (in press). A research base for science education: Historical perspectives. In *Proceedings of the Geselleschaft Fur Didaktik Der Chemie Und Physik (GDCP)*. Kiel, Germany: GDCP.

Linn, M. C. & Burbules, N. C. (1993). Construction of knowledge and group learning. In Tobin, K. (Ed.), *The practice of constructivism in science education* (pp. 91-119). Washington, DC: American Association for the Advancement of Science (AAAS).

Linn, M. C. & Clancy, M. J. (1992a). Can experts' explanations help students develop program design skills? *International Journal of Man-Machine Studies, 36*(4), 511-551.

Linn, M. C. & Clancy, M. J. (1992b). The case for case studies of programming problems. *Communications of the ACM, 35*(3), 121-132.

Linn, M. C. & Clark, H. C. (in press). How can assessment practices foster problem solving? In Lavoie, D. R. (Ed.), *Towards a cognitive-science perspective for scientific problem solving* NARST.

Linn, M. C., diSessa, A., Pea, R. D., & Songer, N. B. (1994). Can research on science learning and instruction inform standards for science education? *Journal of Science Education and Technology, 3*(1), 7-15.

Linn, M. C. & Eylon, B. (in press). Curriculum and the psychology of learning and instruction. In Husén, T. & Postlethwaite, T. N. (Eds.), *The International Encyclopedia of Education* (2nd ed.). New York: Pergamon Press.

Linn, M. C., Katz, M., Clancy, M. J., & Recker, M. (1992). How do LISP programmers draw on previous experience to solve novel problems? In De Corte, E., Linn, M. C., Mandl, H., & Verschaffel, L. (Eds.), *Computer-based learning environments and problem solving.* Berlin: Springer-Verlag.

Linn, M. C. & Kessel, C. (in press). Success in mathematics: Which students persist? In Schoenfeld, A., Dubinsky, E., & Kaput, J. (Eds.), *Research in Collegiate Mathematics Education*.

Linn, M. C. & Petersen, A. C. (1985). Emergence and characterization of sex differences in spatial ability: A meta-analysis. *Child Development, 56*, 1479-1498.

Linn, M. C. & Songer, N. B. (1991). Cognitive and conceptual change in adolescence. *American Journal of Education, 99*(4), 379-417.

Linn, M. C. & Songer, N. B. (1993). How do students make sense of science? *Merrill-Palmer Quarterly, 39*(1), 47-73.

Linn, M. C., Songer, N. B., & Eylon, B. (in press). Shifts and convergences in science learning and instruction: Alternative views. In Calfee, R. & Berliner, D. (Eds.), *Handbook of educational psychology*. Riverside, NJ: Macmillan.

Linn, M. C., Songer, N. B., Lewis, E. L., & Stern, J. (1993). Using technology to teach thermodynamics: Achieving integrated understanding. In Ferguson, D. L. (Ed.), *Advanced educational technologies for mathematics and science* (pp. 5-60) (Vol. 107). Berlin: Springer-Verlag.

Lohman, D. F. (1988). Spatial abilities as traits, processes, and knowledge. In Sternberg, R. J. (Ed.), *Advances in the psychology of human intelligence* (pp. 181-248) (Vol. 4). Hillsdale, NJ: Lawrence Erlbaum Associates.

Madhok, J. J. (1992 April 4 & 5). Group size and gender composition influences on discussion. In *1992 Berkeley Women and Language Conference: Locating Power*. Berkeley, CA: University of California.

Mann, L. M., Linn, M. C., & Clancy, M. J. (in press). Can tracing tools contribute to programming proficiency? The LISP Evaluation Modeler. *Interactive Learning Environments*(Special Issue on Computer Programming Environments).

Marco, R. (1988). *Knowledge organization in novice student programmers: Relation to skill and instruction in programming classes*. Unpublished dissertation, University of California, Department of Educational Psychology, Berkeley, CA.

McCloskey, M., Caramazza, A., & Green, B. (1980). Curvilinear motion in the absence of external forces: Naive beliefs about the motion of objects. *Science, 210*, 1139-1141.

McGrath, O. (1989a). Architects use graphic design software to win competitions. *Instructional Technology Program Newsletter (A publication of the Instructional Technology Program, University of California, Berkeley), 2*(5), 1.

McGrath, O. (1989b). DEC Gift helps students learn computer–aided optimal design. *Instructional Technology Program Newsletter (A publication of the Instructional Technology Program, University of California, Berkeley), 5*(5), 2.

Nersessian, N. J. (1991). Conceptual change in science and in science education. In Matthews, M. R. (Ed.), *History, philosophy, and science teaching* (pp. 133-148). New York: Teacher's College Press.

Newcombe, N. (1981). Spatial representation and behavior: retrospect and prospect. In Liben, L., Patterson, A., & Newcombe, N. (Eds.), *Spatial representation and behavior across the life span: Theory and application* (pp. 373-388). New York: Academic Press.

Newman, D., Griffin, P., & Cole, M. (1989). *The construction zone: Working for cognitive change in school*. London: Cambridge University Press.

Norman, D. A. (1988). *The psychology of everyday things*. New York, NY: Basic Books.

Osborn, J. & Agogino, A. M. (1992). An interface for interactive spatial reasoning and visualization. In *Proceedings of the ACM CHI'92 human factors in computing systems conference* (pp. 75-82).

Pea, R. D. (1992). Augmenting the discourse of learning with computer-based learning environments. In De Corte, E., Linn, M. C., Mandl, H., & Verschaffel, L. (Eds.), *Computer-based learning environments and problem solving*. Berlin: Springer-Verlag.

Raizen, S. A. (1991). The reform of science education in the U.S.A. déjá vu or de novo? *Studies in Science Education, 19*, 1-41.

Reif, F. & Larkin, J. H. (1991). Cognition in scientific and everyday domains: Comparison and learning implications. *Journal of Research in Science Teaching (Special Issue: Students' models and epistemologies), 28*(9), 733-760.

Resnick, L. (1981). Instructional psychology. *Annual Review of Psychology, 32*, 659-704.

Scardamalia, M. & Bereiter, C. (1993). Technologies for knowledge-building discourse. *Communications of the ACM, 36*(5), 37-41.

Schoenfeld, A. H. (1983). Beyond the purely cognitive: Belief systems, social cognitions, and metacognitions as driving forces in intellectual performance. *Cognitive Science, 7*(4), 329-363.

Schwarz, C. (1993). *Differences in planning and monitoring skills of college LISP learners.* [Technical Report]. Berkeley, CA: University of California, HyperMedia Case Studies.

Seymour, E. & Hewitt, N. (1994). *Talking about leaving: Factors contributing to high attrition rates among science, mathematics, and engineering undergraduate majors.* Report on an ethnographic inquiry at seven institutions. New York: Alfred P. Sloan Foundation.

Shepard, R. N. & Metzler, J. (1971). Mental rotations of three-dimensional objects. *Science, 171*, 701-703.

Smith, J. P., diSessa, A. A., & Roschelle, J. (in press). Misconceptions reconceived: A constructivist analysis of knowledge in transition. *Journal of the Learning Sciences*.

Soloway, E. (1985). From problems to programs via plans: The content and structure of knowledge for introductory LISP programming. *Journal of Educational Computing Research, 1*(2), 157-172.

Songer, N. B. & Linn, M. C. (1991). How do students' views of science influence knowledge integration? *Journal of Research in Science Teaching, 28*(9), 761-784.

Spiro, R. J. & Jehng, J. C. (1990). Cognitive flexibility and hypertext: Theory and technology for the nonlinear and multidimensional traversal of complex subject matter. In Nix, D. & Spiro, R. (Eds.), *Cognition, education, and multimedia* (pp. 163-206). Hillsdale, NJ: Lawrence Erlbaum Associates.

Spiro, R. J., Vispoel, W., Schmitz, J., Samarapungavan, A., & Boerger, A. (1987). Knowledge acquisition for application: Cognitive flexibility and transfer in complex content domains. In Britton, B. C. & Glynn, S. (Eds.), *Executive Control Processes in Reading* (pp. 177-199). Hillsdale, NJ: Lawrence Erlbaum Associates.

Torney-Purta, J. (1993 Aug 22). Students' organization of knowledge in social sciences and in law. Paper presented at the *New Fellows in Educational Psychology: Implications of their Work for University-level Instruction,* Toronto, Canada: American Psychological Association.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes* (Cole, M., et al., Series Eds.). Cambridge, MA: Harvard University Press.

Webb, N. M. (1989). Peer interaction and learning in small groups. *International Journal of Educational Research, 13*(1), 21-39.

Webb, N. M. & Lewis, S. (1988). The social context of learning in computer programming. In Mayer, R. E. (Ed.), *Research on teaching and learning computer programming* (pp. 179-206). Hillsdale, NJ: Lawrence Erlbaum Associates.

West, L. H. T., Pines, A. L., & Sutton, C. R. (1984). In-depth investigations of learners' understandings of scientific concepts and theories. In Pines, A. L. & West, L. H. T. (Eds.), *Cognitive structure and conceptual change*. New York, NY: Academic Press.

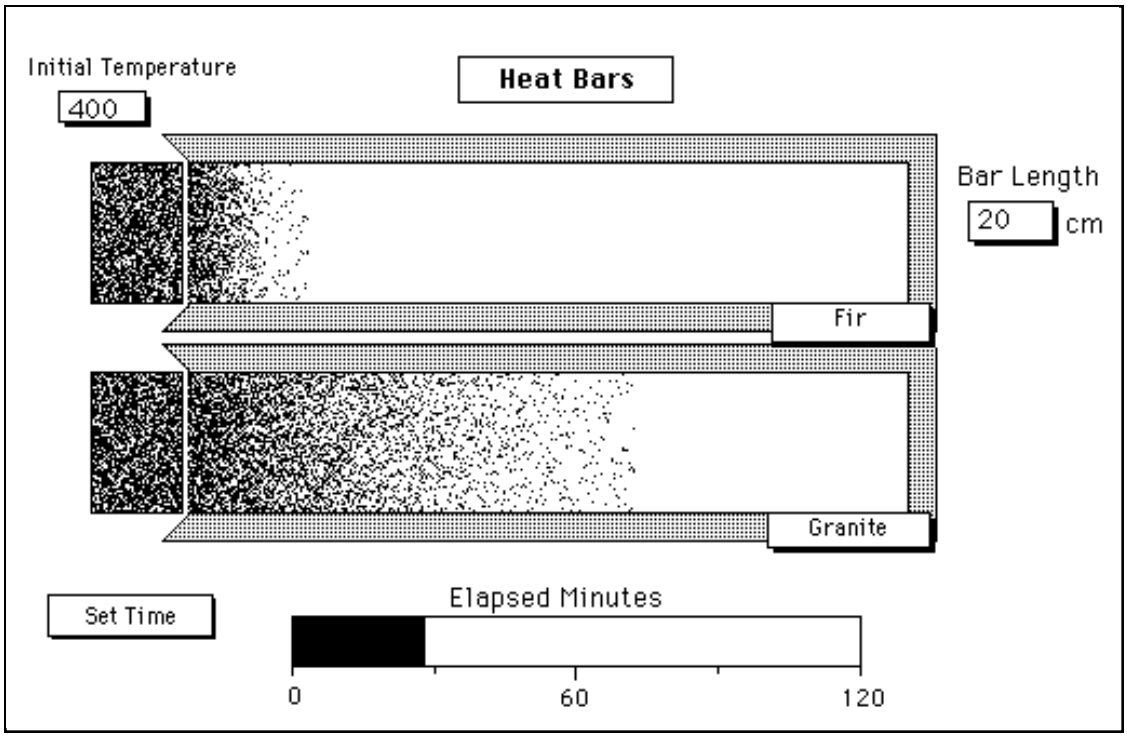
White, B. Y. (1993). ThinkerTools: Causal models, conceptual change, and science education. *Cognition and Instruction, 10*(1), 1-100.

White, B. Y. & Frederiksen, J. R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence, 24*(1), 99-157.

Winograd, T. & Flores, F. (1987). *Understanding computers and cognition: A new foundation for design*. Reading, MA: Addison-Wesley Publishing Co.

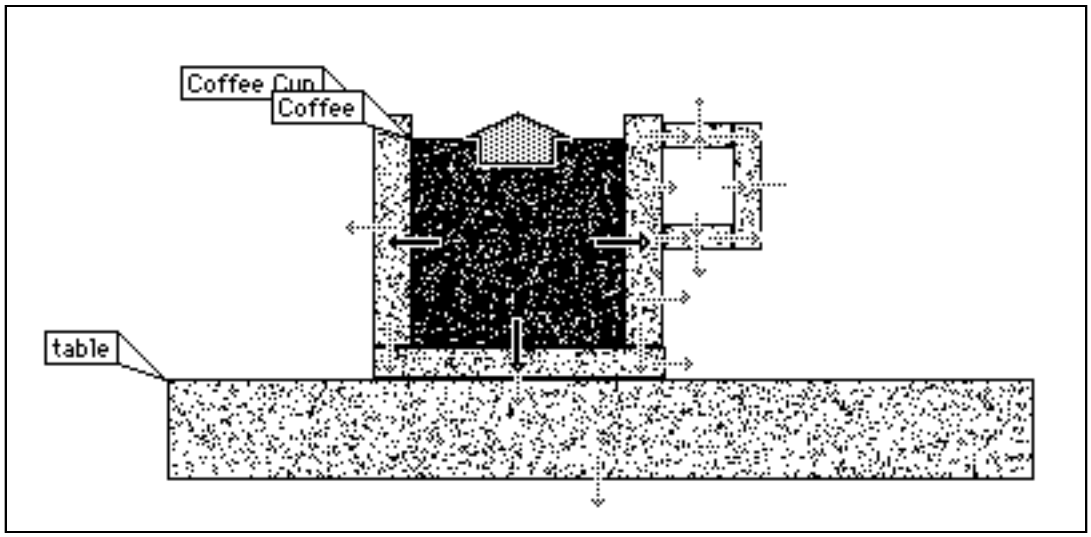
Figure 1: Heat Bars and Thermal Model Kit

A. Heat Bars



The HEAT BARS program shows students how bars of different materials would heat up over time when put in contact with a heat source. The shading of the bars changes as time elapses indicating the heat flowing through the bar.

B. Thermal Model Kit



Thermal Model Kit allows students to design and simulate heat flow in an everyday situation. Here a cup of coffee sits on a table. The thickness of the arrows indicate the speed of the heat flow between different objects. As in Heat Bars, the shading indicates the temperature of the object.

Figure 2: Applying LEM to the applicative operator "find if."

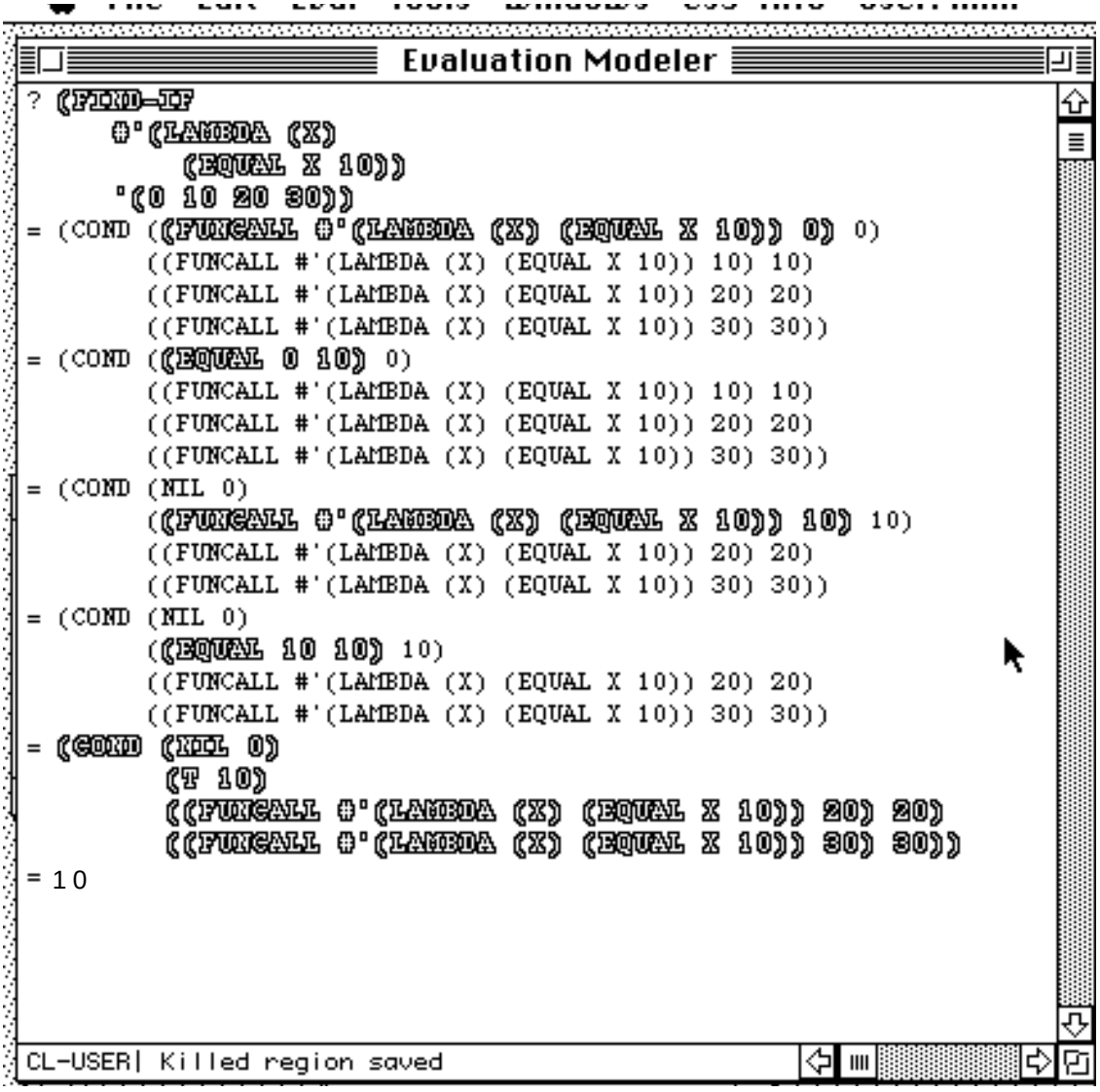


Figure 3: Interactive Case Studies: Testing with CodeProbe

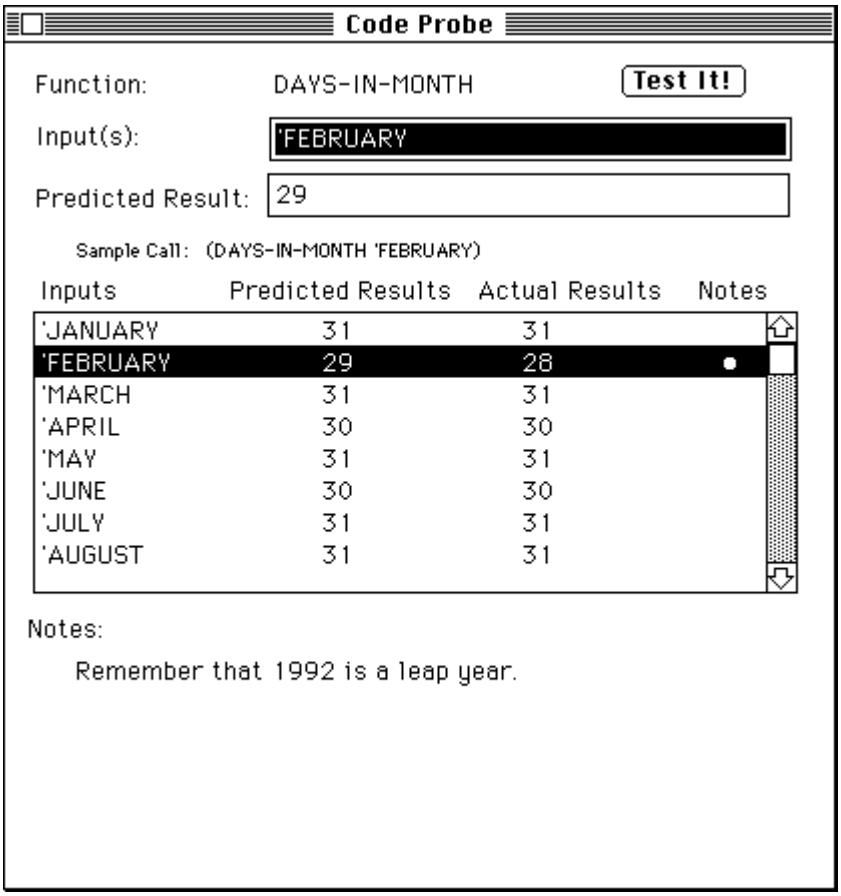


Figure 4: Grades in Computer Science Course by Gender: Effect of Improved Instruction

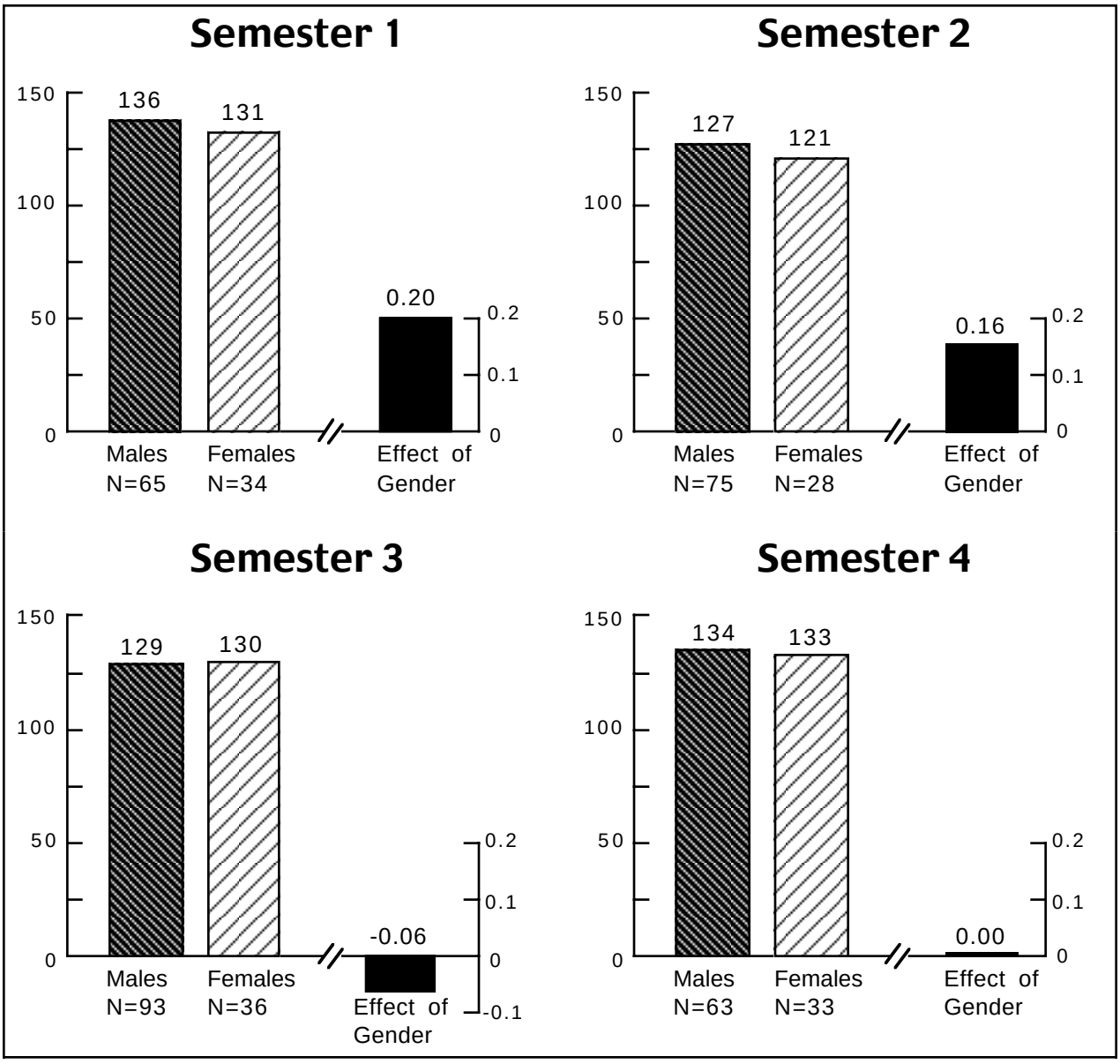


Figure 5

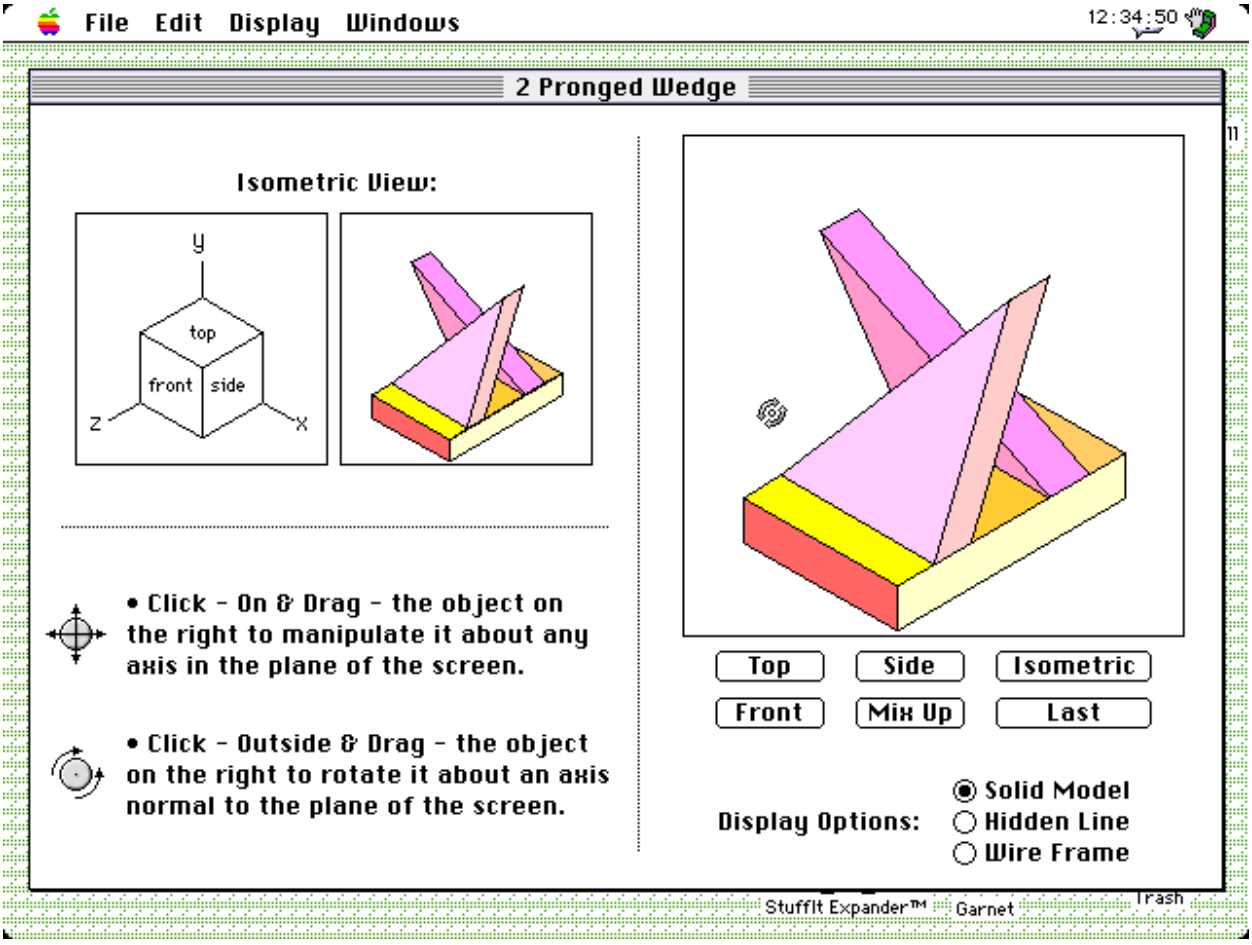

File Edit Display Windows
12:34:50
2 Pronged Wedge
Isometric View:
y
top
front
side
z
x
• Click - On & Drag - the object on the right to manipulate it about any axis in the plane of the screen.
• Click - Outside & Drag - the object on the right to rotate it about an axis normal to the plane of the screen.
Top
Side
Isometric
Front
Mix Up
Last
Display Options:
Solid Model
Hidden Line
Wire Frame
StuffIt Expander™
Garnet
Trash

Figure 6

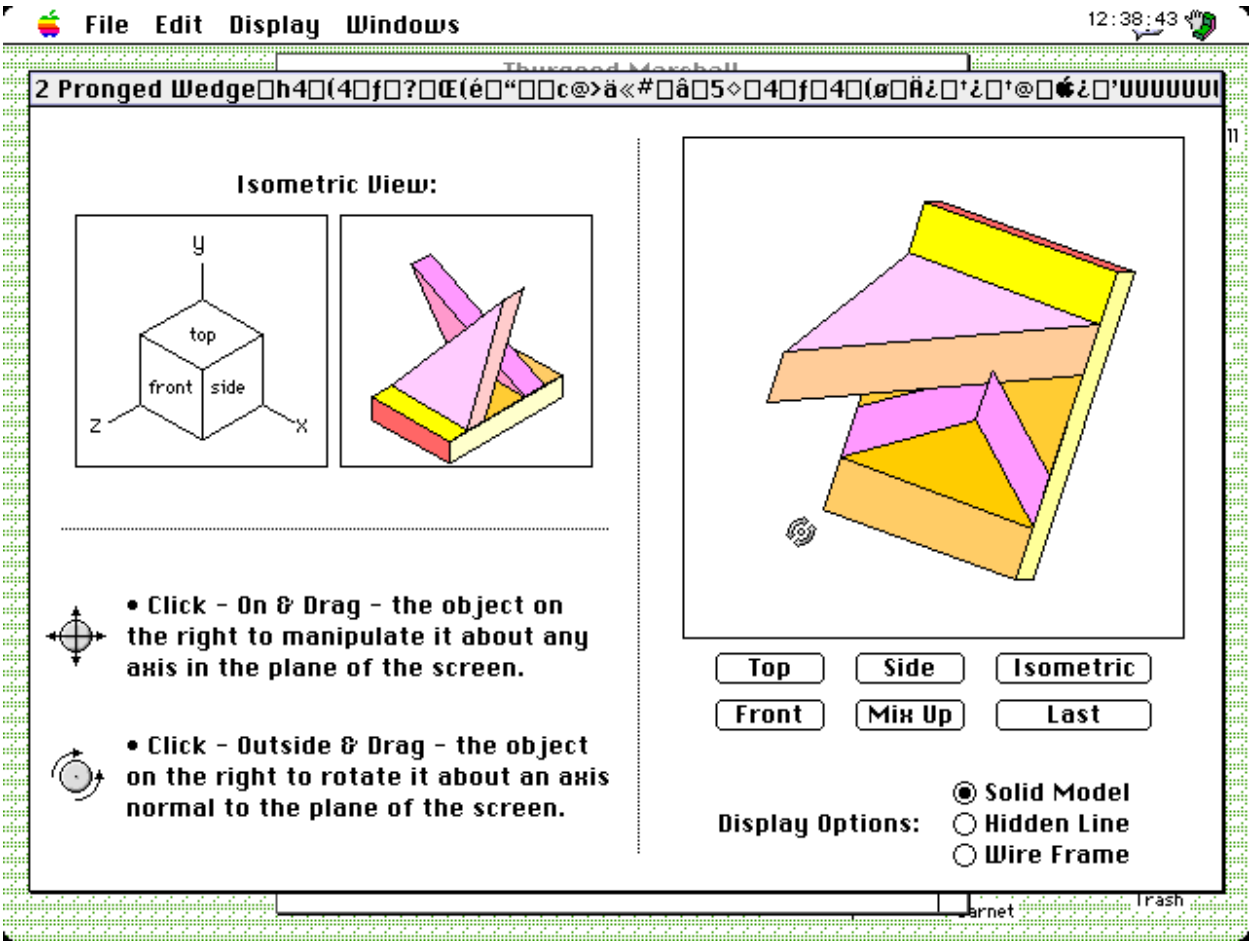

File Edit Display Windows
12:38:43
2 Pronged Wedge
Isometric View:
y
top
front
side
z
x
• Click - On & Drag - the object on the right to manipulate it about any axis in the plane of the screen.
• Click - Outside & Drag - the object on the right to rotate it about an axis normal to the plane of the screen.
Top
Side
Isometric
Front
Mix Up
Last
Display Options:
Solid Model
Hidden Line
Wire Frame
Trash

Figure 7

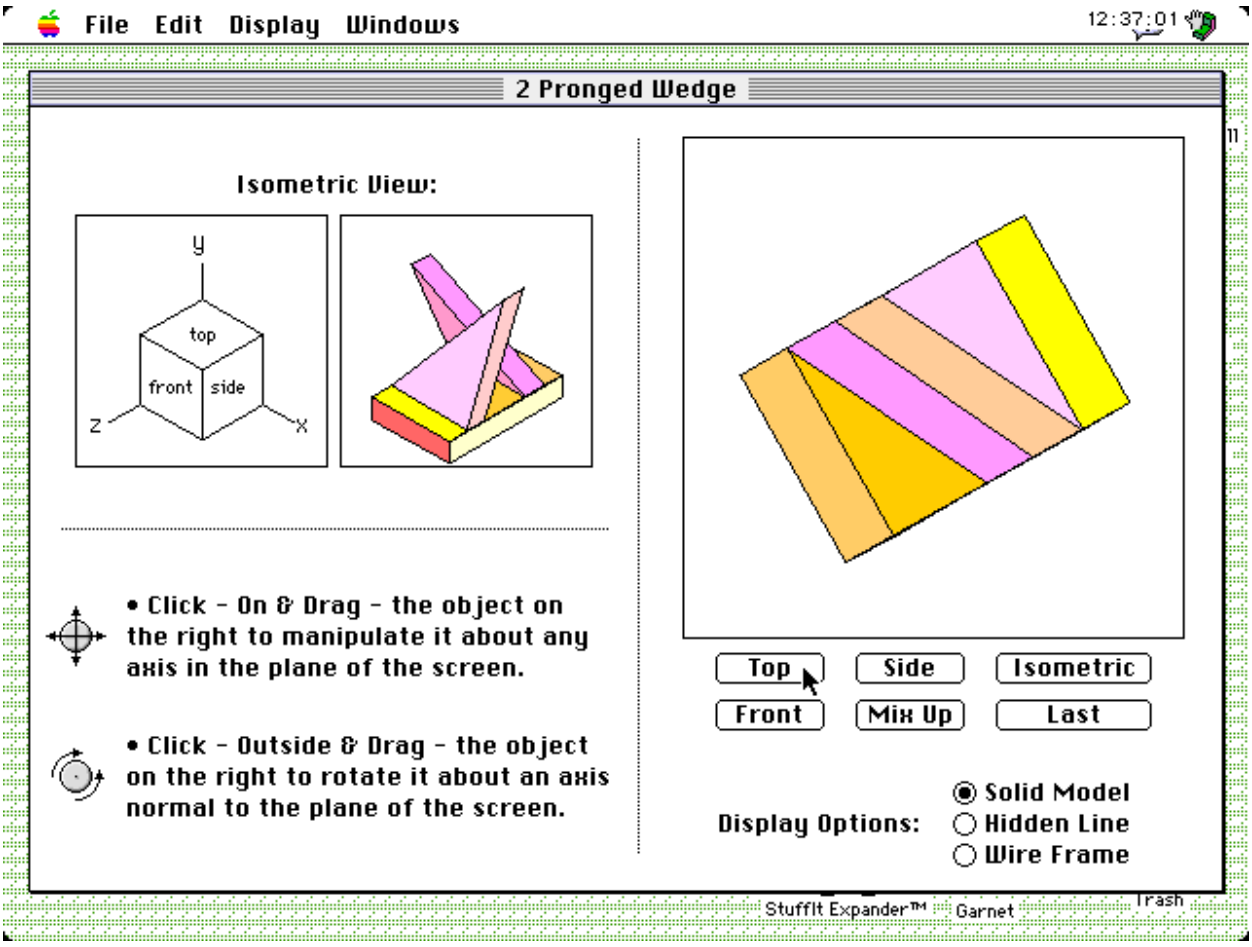

File Edit Display Windows
12:37:01
2 Pronged Wedge
Isometric View:
y
top
front
side
z
x
• Click - On & Drag - the object on the right to manipulate it about any axis in the plane of the screen.
• Click - Outside & Drag - the object on the right to rotate it about an axis normal to the plane of the screen.
Top
Side
Isometric
Front
Mix Up
Last
Display Options:
Solid Model
Hidden Line
Wire Frame
StuffIt Expander™
Garnet
Trash

Figure 8: Scores on spatial reasoning pretest and posttest for males and females enrolled in graphical communication course

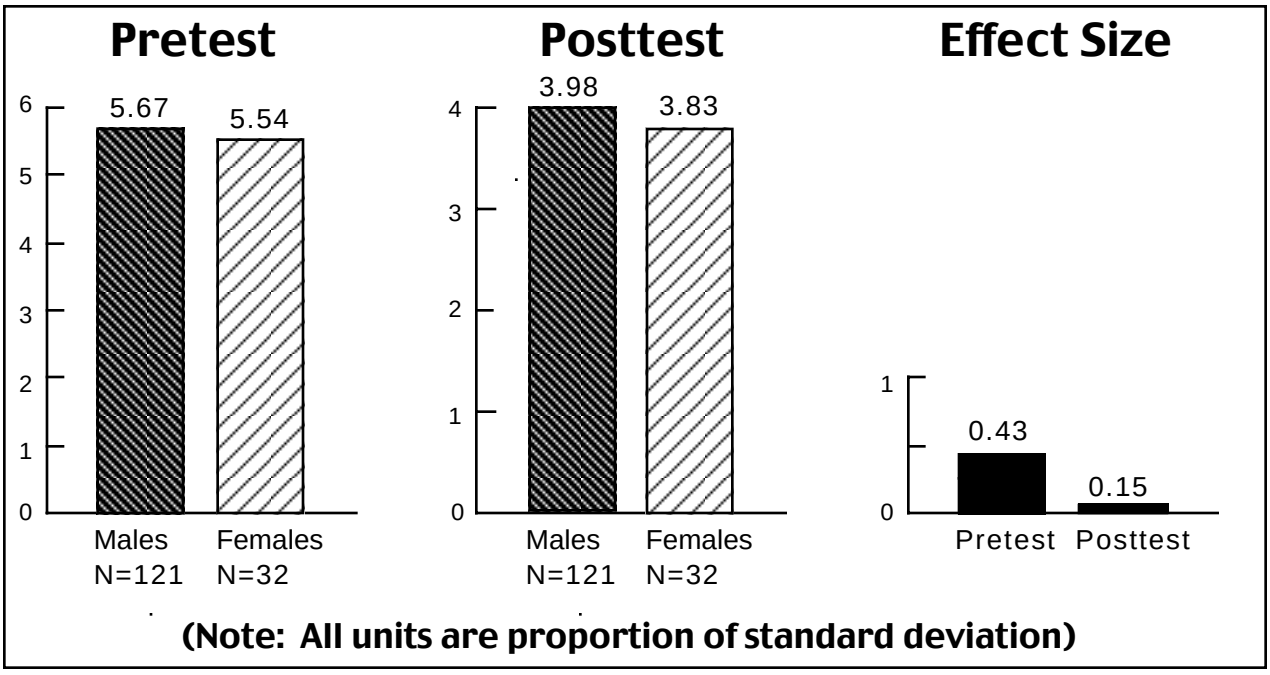


Table I

Grades in Computer Science Course by Gender:

Effect of improved instruction

	Males			Females			Effect Size
	μ	SD	N	μ	SD	N	
Semester 1	136	33	65	131	30	34	0.20
Semester 2	127	41	75	121	36	28	0.16
Semester 3	129	37	93	130	30	36	-0.06
Semester 4	134	33	63	133	34	33	0

Table II:

Place table II here (Must print as separate document to accommodate page orientation)