PROPOSAL: INSTRUCTIONAL MATH TESTING SOFTWARE

by

Joel W. Duffin

A proposal for a project submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE (PLAN C)

in

Instructional Technology

Approved:

____
Major Professor
Andrew S. Gibbons

____
Committee Member
Don C. Smellie

____
Committee Member
Charles Stoddard

____
Head, Department of
Instructional Technology
Don C. Smellie

UTAH STATE UNIVERSITY

Logan, Utah

1998

AKNOWLEDGMENTS

I would like to thank Dr. Larry Cannon, Dr. Andrew S. Gibbons, Dr. Robert Heal, and Dr. Richard Wellman for the opportunities, encouragement, good ideas, and hard questions they contributed to make this project enjoyable and profitable.

CONTENTS

INTRODUCTION

The objective of the Instructional Math Testing Shells project is to design and create software which instructs students in solving math problems. The software is to accompany a college Precalculus course text book. The two main areas of focus will be the design of user interfaces (UIs) and intelligent tutoring systems (ITSs). UIs will be designed to support the cognitive processes of the learner. The UIs will require learners to externalize steps of their problem solutions. Not only will the UIs require learners to show the steps commonly shown in writing, but also the invisible steps normally performed only in a problem solver's mind, such as choosing goals and evaluating progress. The ITSs will model expert problem solving and give coaching and feedback messages to the learner. Both the UI and the ITSs will be parameterized to handle any problem of a given type. Shells will be created for two problem types: simultaneous linear equations and triangles.

PROBLEM STATEMENT

Statement of Problem

Practice is an essential part of learning any skill, especially Math problem solving. However, all practice is not equal. The amount and type of learner support given during practice affects the engagement of the learner, the time spent, and the resulting learning that occurs (Brown, Collins, Duguid, 1989). Depending upon how well a learner has mastered a given topic, different levels of support are most effective in continuing to motivate and challenge the learner towards further mastery (Glaser, 1977). Given these assumptions, the challenge of this project is to create software that operates in a variety of modes, each mode designed to support a different stage of learning.

In much traditional math practice, especially computerized math practice, mechanics is emphasized above process (Schank and Cleary, 1995). This seems to be caused by the reality that it is easier for both the grader and the computer to evaluate mechanics than processes. For this reason, one of the primary goals of this project is to create user interfaces that support problem solving by requiring the learner to externalize their problem solving steps including the selection of goals and evaluation of progress. It is believed that requiring learners to externalize goals will help them analyze, understand, and manage their problem solving processes. Research has shown that scaffolding the selection of problem subgoals helps students acquire problem solving skills (Corbett, Anderson, 1995). A parallel motivation for focusing on the user interface is to create user interfaces that allow expert models of problem solving to be externalized to the learners view. Attempts will be made to create UIs that externalize expert models of problem solving in ways that focuses learners’ attention on the essential elements of solving a particular type of problem. Research utilizing methods such as reciprocal teaching has shown that appropriate demonstrations of expert problem solving facilitate learning (Brown, Campione, Reeve, Ferrara, Palincsar, 1991). It is hoped that interfaces which allow the learner to externalize their problem solving processes will also allow expert models of problem solving to be externalized.

Another challenge for creating math practice software is to design human computer interactions that support learning. Interactions to be created include modeling of expert solutions, computer interjected or learner requested coaching, and feedback. ITSs will be created to implement these features for use in coherent instructional strategies. Evaluations of ITSs have demonstrated large positive effects from ITSs that included the following instructional features (Shute and Psotka, 1996).

PURPOSE AND OBJECTIVES

The purpose of this project is to create a software practice environment to accompany a college Precalculus Math book. In designing and creating this software we hope to gather information that will help answer the following questions:

- How do you design interfaces that focus learner's attention on problem solving processes without being so cumbersome that a student will not use it?
- How do you use embody expert problem solving knowledge in an ITS so that it can be used to provide instructional features?

Created software will be used in future research to examine the questions:

- Does externalizing problem solving process while modeling expert solutions to problems enhance learning?
- Does requiring learners to externalize their problem solving process enhance learning?

REVIEW OF LITERATURE

The following areas of research and instructional theory will be used to guide the creation of the software in this project: Cognitive Apprenticeship, Cognitive Task

Analysis, Intelligent Tutoring Systems, Feedback, and Learner Control. The following paragraphs briefly describe some of the key recommendations of these areas as they relate to this project.

Cognitive Apprenticeship

In their landmark 1987 article, Collins, Brown and Newman laid out principles of cognitive apprenticeship as a learning theory in which modeling of expert performance plays a central role.

> In order to make real differences in students' skill, we need both to understand the nature of expert practice and to devise methods that are appropriate to learning that practice. As designers of instruction we must recognize that for a learner to become an expert, they must understand more than what can be physically observed. We must first recognize that cognitive and metacognitive strategies and processes, more centrally than low-level subskills or abstract conceptual and factual knowledge, are the organizing principles of expertise. (Collins et al., 1987, p. 4).

Collins et al. state that expertise can be best taught "through successive approximation of mature practice" (ibid.). Cognitive, versus traditional apprenticeship focuses on the conveying of cognitive skills which are often unobservable. They go on to explain that:

> Apprenticeship highlights methods for carrying out tasks in a domain. The apprentice repeatedly observes the master executing (or modeling) the target process. The apprentice then attempts to execute the

> process with guidance and help from the master (coaching). A key aspect of coaching is the provision of scaffolding, which is the support, in the form of reminders and help, that the apprentice requires to approximate the execution of the entire composite of skills. Once the learner has a grasp of the target skill, the master reduces his participation (fades). The interplay between observation, scaffolding, and increasingly independent practice aids apprentices both in developing self-monitoring and correction skills. Observation aids learners in developing a conceptual model of the target task or process prior to attempting to execute it (advance organizer). (Collins et al., 1987, pp. 4-5).

Intelligent Tutoring Systems

Instructional Technologists and AI researchers have long pursued the goal of creating software programs that can interact with learners as expert instructors do. This would make quality instruction cost lest and be more available to students. While these original conceptions of ITS appear to be too ambitious, less ambitious applications of ITS are common in university research settings. Evaluations of some of these systems are very positive. (Shute and Psotka, 1996). However, there are very few ITSs currently used in schools.

Problem Solving Cycle

The author proposes a Problem Solving Cycle model for analyzing human problem solving activities. The model identifies four stages of a cyclical problem solving process: establishing goals, selecting and executing actions, observing outcomes, and evaluation of progress. When faced with a problem and a set of conditions, a person breaks down the problem into goal subgoals they believe will lead them to the solution of the problem. They proceed to select from relevant actions an action they believe will help them accomplish the most pressing goal. After acting, a person observes the results of their action and evaluates the effectiveness of their action towards accomplishing subgoals. The cycle then repeats with the problem solver modifying subgoals based on information gained. This cycle is similar to models of human cognition during problem solving put forth by cognitive science and artificial intelligence (Lovett and Anderson, 1996). A conclusion of research examining the ACT* AI model of human cognition is that communicating the goal structure of a problem to the learner is an important instructional objective (Anderson, Boyle, Farrell and Reiser, 1997).

This cyclic view of the problem solving process is similar to a cognitive troubleshooting strategy known as PARI. In the PARI method, troubleshooters systematically solve problems by analyzing existing conditions, acting based on their analysis, observing the results of their actions, and interpreting their observations (Hall, Gott, and Pokorny, 1990).

Feedback

Definitions of what constitutes feedback vary widely and there is disagreement about what type of feedback is most effective for a given situation.

> There is a fine line and balance between providing a learner with too much or too little support in their learning process. Provide too much support and the learner does well during the instruction, but does not

> retain. Provide too little, and the learning process takes longer than it needs too and misconceptions are learned (Dempsey and Sales, 1993, p. 33).

One review of 40 reports of research involving feedback found that feedback can promote learning if it is received mindfully, but may inhibit learning if it encourages mindlessness. In the studies examined, delayed feedback resulted in lower effect sizes than immediate feedback (Bengert Downs, Kulik, Kulik, and Morgan 1991). Additional studies are needed to increase our understanding of the timing and content of feedback most beneficial for given learning styles and stages in learning.

Learner Control

The issue of learner control has long been a question of heated debate, On one side of the issue there are those who favor giving all control to the learner. On the other side are those who favor tightly structured instruction where the learner has little control. Shute and Psotka (1996) point out that the issue of learner control is a complex one because it must take into account differences in learning styles as well as stages in learning. They claim that an ideal learning environment would allow the amount of learner control to be varied according to the learner and where the learner is in the learning process.

DESIGN PLAN

The project will attempt to use information gathered from an analysis of the literature in the mentioned areas.

Cognitive Apprenticeship

The software will externalize expert reasoning for learners to see. It will also require learners to make explicit subgoals they are pursuing, allowing the computer to provide feedback on their reasoning as well as their performance.

The software will provide different levels of learner support, allowing the learner to move between them as their ability improves. Students will be able to request and watch commented expert solutions of problems. In addition they will be request coaching and feedback during practice.

Intelligent Tutoring Systems

ITS will be created to provide instructional features. The created software will be used as part of an actual college math course. If successful, additional ITSs will be created and used in college math courses.

ITS will be used to model expert solutions of an infinite but bounded set of problems. In addition, the ITS will provide feedback on learners performance by comparing expert solutions with learners. The power of the ITS is enhanced by the requiring learners to externalize their selection of goals and evaluation of progress.

Problem Solving Cycle

User interfaces as well as the expert modeling and feedback will be designed based on an analysis of the targeted problem types using the proposed model of problem solving. Controls will be created to allow learners to make explicit their selection of goals and evaluation of the outcome of actions taken. When models of expert performance are given, these same controls will be used to externalize expert cognitive performance so that learners do not have to guess at the reasoning used by the expert in making decisions. When learners practice problem solving, they will be required to make explicit the goals they are attempting to accomplish and to evaluate their progress towards those goals. Requiring the learner to specify this information helps support the early stages of learning in which learners need to consciously identify salient features of a problem and the goals they select (Anderson, Boyle, Farrell and Reiser, 1997).

Feedback

Cognitive task analysis (CTA) will be used to elucidate the mental models used by experts when solving problems in the domains examined. The results of the analysis will be used to formulate customized feedback messages. An important aspect of being able to use the CTA to provide meaningful feedback messages is that the studied problem solving environments require the user to specify goals they choose and their evaluation of the results of operations they perform. It is hypothesized that feedback messages based on specified goals will be significantly higher than messages based solely on evaluation of operations and results.

Learner Control

As learner ability increases, learners will be able to switch to modes that do not require them to explicitly state goals and evaluate progress. The authors believe this functionality supports the automation of the problem solving processes by helping to lower them to the subconscious level. The switcheable nature of the instructional features in the proposed software will allow questions of learner control to be examined. The software will give the learner control over the amount and type of support given during problem solving activities.

PROCEDURES AND METHODOLOGY

The student will work with professors in both the Math and Instructional Technology departments to receive guidance and formative evaluation on the creation of the proposed software. In addition one-on one evaluations will be performed using members of the target audience for the product. The steps to complete the project will be:

1. Locate and analyze related literature and products.
2. Work with experts using speak aloud protocols to analyze their solutions of instances of given problem types.
3. Document the production rules an expert uses to solve the problem type.
4. Create written prototypes of the user interfaces and interactions to provide.
5. Create software prototypes of the user interfaces.
6. Present software prototypes to professors for concept approval.
7. Program fully functioning user interfaces.
8. Program fully functioning instructional features.
9. Create a database of problems to be provided with the shells.
10. Test and debug the software using the database problems.

The software will be created using Toolbook to author user interfaces and C++ to implement the intelligent tutoring system used to provide instructional features.

REPORTING PROCESS

The results of the project will be reported in written and oral form. The written summary will include screen shots and explanations of the created software. The oral report will include a demonstration of the software that highlights features of the user interface and the instructional modes. Discussion following the oral presentation will focus on the instructional principles on which the design of the software is based as well as how the product could be implemented as part of a math course.

REFERENCES

Anderson, J. R., Boyle, C. F., Farrell, R., Reiser, B. J. (1987). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), Modeling Cognition, Wiley.

Bangert-Downs, R.L., Kulik, Chen-Lin C., Kulik, J.A., Morgan, M. (1991). The Instructional Effect of Feedback in Test-Like Events. Review of Educational Research, pp. 218-238.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18(1), pp. 32-42.

Brown, A. L., Campione, J. C., Reeve, R. A., Ferrara, R. A., Palincsar, A. S. (1991). Interactive learning and individual understanding: The case of reading and mathematics. In Landsmann, L. T. (Ed.), Culture, schooling, and psychological development (pp. 136-170). Norwood, NJ: Ablex Publishing Co.

Collins, A., BBN Laboratories, Brown, J. S., Newman, S. E. (1987). Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics, Xerox Palo Alto Research Center, Technical Report No. 403.

Corbett, A.T., Anderson, J.R. (1995). Knowledge decomposition and subgoal reification in the ACT programming tutor. Artificial Intelligence and Education, 1995: The Proceedings of AI-ED 95. Charlottesville, VA: AACE.

Dempsey, J.V., Sales G.C. (Eds) (1993). Interactive Instruction and Feedback. Educational Technology Publications, Inc., Englewood Cliffs, New Jersey.

Glaser, Robert (1977) Adaptive Education: Individual Diversity and Learning. New York: Holt, Rinehart and Winston.

Hall, E.M., Gott, S.P., Pokorny, R.A. (1990). A procedural guide to cognitive task analysis: The PARI methodology. Armstrong Laboratory Technical Report, Brooks AFB, TX., Armstrong Laboratory, Human Resources Directorate, Manpower and Personnel Research Division.

Lovett, M. C., Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. Cognitive Psychology, 31(2).

Schank, R.C., Cleary, C. (1995). Engines for Education, San Mateo: Morgan Kaufman.

Shute, V. J., Psotka, J. (1996). Intelligent Tutoring Systems: Past, Present, and Future. To Appear in D. Jonassen, (Ed.), Handbook of Research In Educational Communications and Technology. New York: Scholastic Publications.

SUMMARY: INSTRUCTIONAL MATH TESTING SOFTWARE

by

Joel W. Duffin

A project summary submitted in partial fulfillment
of the requirements for the degree
of

MASTER OF SCIENCE (PLAN C)

in

Instructional Technology

Approved:

Major Professor
Andrew S. Gibbons

Committee Member
Don C. Smellie

Committee Member
Charles Stoddard

Head, Department of
Instructional Technology
Don C. Smellie

UTAH STATE UNIVERSITY
Logan, Utah
1998

INTRODUCTION

The objective of the Instructional Math Testing Shells project was to design and create software which instructs students in solving math problems. The created software accompanies a college Precalculus course text book co-authored by Larry Cannon a professor in the Utah State University math department. Creation of the software was funded as part of a Higher Education Technology Initiative (HETI) grant obtained by Dr. Cannon and Dr. Heal in the USU Math Department. In addition to working with Dr. Cannon and Dr. Heal, the candidate worked closely with Dr. Wellman from the USU Math Department and with Dr. Gibbons from the USU Instructional Technology Department.

Two main areas of focus were the design of user interfaces (UIs) and intelligent tutoring systems (ITSs). UIs were designed to support learning of cognitive processes by requiring the learner to show the steps of their problem solutions. Not only do the UIs require the learner to show the steps commonly shown in writing, but also the invisible steps normally performed only in a problem solver’s mind, such as choosing goals and evaluating progress. The ITSs are used to model expert problem solving, give coaching and feedback messages to the learner. Both the UI and the ITSs are parameterized to handle any problem of a given type. Shells were created for two problem types: simultaneous linear equations and triangles.

PROJECT DESCRIPTION

The candidate's involvement in the HETI project began when Dr. Gibbons introduced him to Dr. Heal and Dr. Cannon. Through a series of meetings the scope, possibilities, and expectations of the project were worked out. An initial problem type, Simultaneous Linear Equations (SLEs), was identified for which to create a problem solving environment. Dr. Gibbons and the candidate performed a cognitive task analysis (CTA) to identify subgoals, and operations used in solving the problem type. They designed an interface on which expert solutions of the problem type could be displayed and practice could be performed. During initial design period, the candidate began reviewing literature related to Math instruction, user interface design, and intelligent tutoring systems (ITSs).

After analyzing the problem type and designing a user interface, the candidate programmed a prototype of the problem workspace using Asymetrix Toolbook. He met with Dr. Heal, Dr. Cannon, and Dr. Gibbons to present the interface as well as ideas about the types of instructional interactions that could be provided with the interfaces. The candidate used feedback from the meetings to refine and create a number of alternative user interfaces for the SLE workspace. A conscious effort was made to design an interface that allows a user to easily specify goals and evaluate of their progress towards those goals.

Once a user interface was agreed upon, the candidate began working on an ITS that could provide demonstration, explanation, and feedback instructional features. He used information gathered from the CTA as a basis for which to document and program production rules for the ITS. The ITS for the SLE problem shell was programmed using C++ and connected with the Toolbook interface. Additional meetings between the candidate and math professors were held to get feedback on the interface and instructional features. In order to test all areas of functionality, the candidate created a database of problems that could be read into the SLE shell. The candidate continued working to find and fix bugs in the shell as well as to refine the implementation of the interface. He met with Dr. Soulier and other professors in the Instructional Technology Department to receive advice on the design of the UI.

When the SLE shell was nearly completed, the candidate began working with Dr. Heal on the creation of a workspace for solving triangle problems. They used the same process of analysis, design, UI prototype, and ITS development as was used to create the SLE shell. Additional human computer interaction issues were considered including drawing and labeling diagrams.

In addition to creating the described SLE and Triangle shells, the candidate consulted with the Dr. Heal, Dr. Cannon, and Dr. Wellman to give feedback, programming assistance, and design suggestions on software they were creating. He began work on a shell for solving algebra problems involving identities. The candidate wrote papers giving guidelines for designing multimedia using Toolbook and describing types and uses of human computer interactions. He assisted in the integration of Toolbook modules with an electronic version of the Precalculus book.

CONCLUSION

While working on the project, the candidate learned lessons about project management, user interface design, expert system creation, and instructional features.

Early in the project, the candidate aspired to creating as many as ten instructional testing shells. As work began, it quickly became apparent that it was taking longer than expected to create the shells. The hope was that expertise gained from creation of the first shell could be leveraged to greatly reduce the time required to create later shells. To some degree this occurred, however not to the degree expected. The candidate learned the importance of making realistic plans, carefully monitoring progress and modifying plans when necessary.

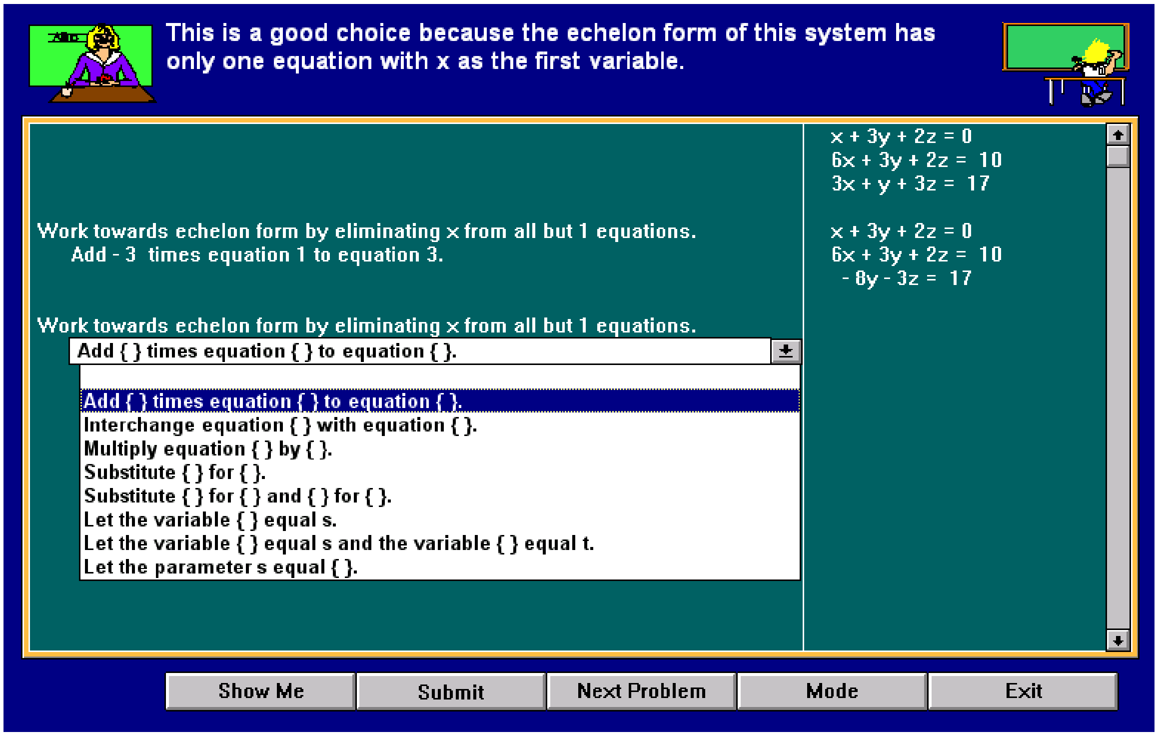
By designing and evaluating problem solving interfaces with users, the candidate learned lessons of user interface design: the simpler an interface, the better; if a UI element does not serve a purpose it should be removed from the screen; interfaces should model the processes they are designed to support.

Through the process of CTA, programming production rules, and debugging the ITS, the candidate learned that creating a useful ITS is complex. The candidate came away from the experience believing that modeling of expert problem solving should include explanations of subgoals pursued and strategies used. He is currently involved in conducting research to test these hypothesis. In addition, scaffolding learner’s selection of subgoals during practice can aid their learning of problem solving.

RECOMMENDATIONS

The candidate recommends that future candidates take time to learn about past efforts similar to theirs. They should leverage existing tools and products to create new products. If possible, form a multidisciplinary team including project managers, subject matter experts, instructional designers, programmers, and graphic designers. This allows each person involved to focus on their areas of greatest interest and strength.

FIGURES



This is a good choice because the echelon form of this system has only one equation with x as the first variable.

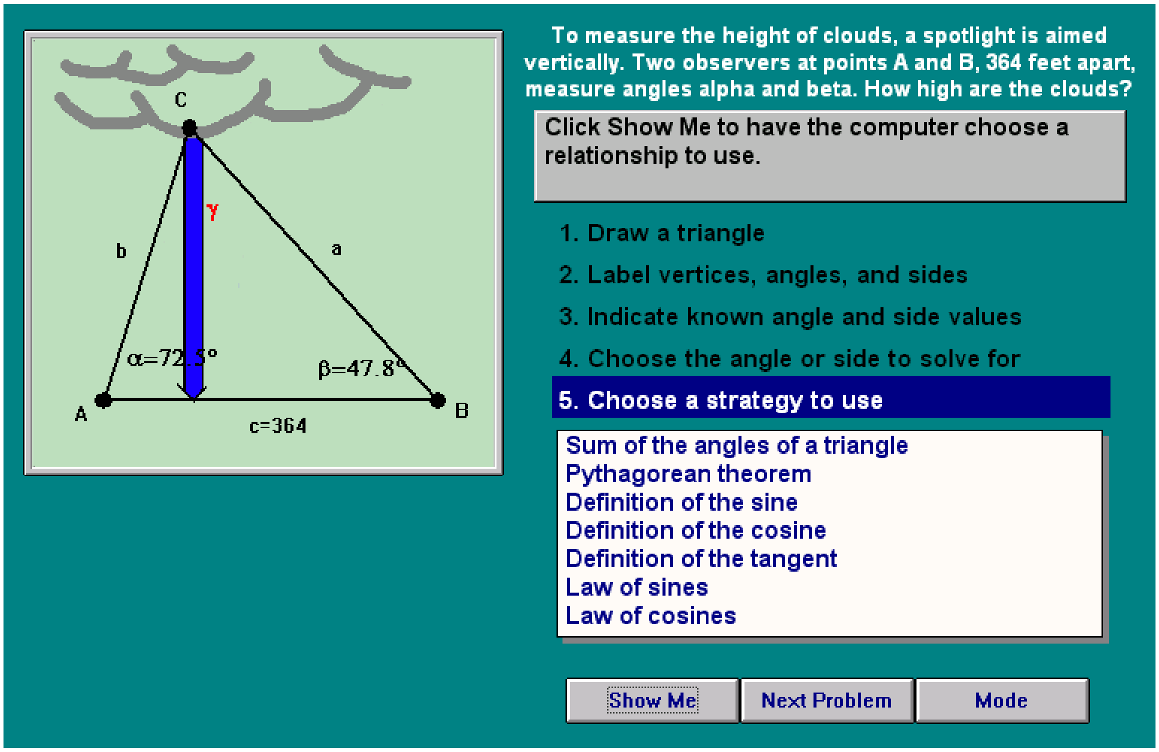
Work towards echelon form by eliminating x from all but 1 equations.
Add - 3 times equation 1 to equation 3.

Work towards echelon form by eliminating x from all but 1 equations.
Add { } times equation { } to equation { }.

$$x + 3y + 2z = 0$$
$$6x + 3y + 2z = 10$$
$$3x + y + 3z = 17$$

$$x + 3y + 2z = 0$$
$$6x + 3y + 2z = 10$$
$$-8y - 3z = 17$$

Figure 1: Simultaneous Linear Equations Testing Shell



To measure the height of clouds, a spotlight is aimed vertically. Two observers at points A and B, 364 feet apart, measure angles alpha and beta. How high are the clouds?

Click Show Me to have the computer choose a relationship to use.

1. Draw a triangle
2. Label vertices, angles, and sides
3. Indicate known angle and side values
4. Choose the angle or side to solve for
5. Choose a strategy to use

Figure 2: Triangle Problem Solving Shell