2001

An Interactive Visual Environment for Scientific Problem Solving

Georg F. Mauer
University of Nevada, Las Vegas, georg.mauer@unlv.edu

An Interactive Visual Environment for Scientific Problem Solving

Georg F. Mauer
University of Nevada, Las Vegas

Abstract

Science and engineering students do not typically receive explicit training in scientific problem solving, i.e. applying science principles to specific situations. Students' problem solving skills often show little improvement throughout their course of studies.
This paper describes a structured, graphical, interactive (GUI) learning environment, which presents problems and tools for analysis in systematic and logical order, and encourages the student to develop the solution path in the manner of an experienced expert. The learning environment's subject area is Engineering Dynamics, which was selected for its systematic structure and its early (usually sophomore) place in the undergraduate curriculum.
The software presents the concepts required to solve homework problems, organized along book chapters. First, the student is prompted to analyze the problem statement, i.e. to extract relevant information from the text and classify the problem. Free-body diagrams are developed interactively on-screen. The problem solution is then developed conceptually by applying the problem information to the current (and preceding) chapter's laws as appropriate. The conceptual solution is complete if the number of variables in the problem matches the number of equations in the conceptual solution set. Lastly, the quantitative solution is developed in Mathcad, using the applicable laws from the conceptual solution and the data given. The problem solving software thus creates and reinforces a pattern for problem solving which is typically absent among novice students: they tend to start the solution process with the numbers at hand, and then try to find an equation that yields the desired result. Over time, the software thus is expected to train students in systematic problem solving. Context-sensitive help throughout explains laws, procedures, and their possible connections to the problem at hand.

I. Introduction

Scientific problem solving in science and engineering education is a skill acquired by intensive and often frustrating training. Even when students understand the pertinent scientific theories and mathematics, no clear path or general strategy is typically visible to the beginner. There is ample evidence that teaching excellence (such as well designed presentations of sample solutions) and students' subject knowledge do not *per se* translate into the acquisition of problem solving skills. Rather than solving by applying principles and laws, students often find it expedient to emulate sample solutions.

The Importance of Problem Solving: The difficulty of teaching what H.M. Paynter (1961) in a classical paper termed *'the art of modeling'* is discussed, among others, in Hogan (1991) and Stein (1991). Problem solving involves the focused application of the learned science and math principles to a specific scenario. Problem solving requires additional skills, especially an in-depth analysis of the problem structure and a mapping of the path to the answer: see e.g. Caillot (1983), Pankratius (1990), Bucciarelli et al. (2000). Scrivener et al. (1994) show that instruction targeted at problem solving increases student performance and satisfaction significantly. The lack of problem solving skills extends also to social and behavioral sciences (Woods et al., 1997). The Chemical Engineering Department at McMaster University restructured its curriculum to create four new undergraduate courses dedicated to problem solving and related issues such as motivation and self-confidence (Woods et al, 1997). Barrett et al. (1998) describe a visual environment for teaching and designing digital systems, which presents guidance and tools as needed for each step. The authors report a quantitative improvement in student performance resulting from the availability of information as needed during the design process.
Several key findings from the thorough case study at McMaster University by Woods et al. (1997) are quoted below:

> "(Faculty) … worked problems, supplied sample solutions, and showed a variety of problem solving heuristics. However, when tested, our students were unable to recognize, transfer or apply the skill for the process of solving their homework problems. In-class faculty demonstrations of the faculty's skill in solving problems did not transfer skill to the students."
>
> "The faculty asked students to show how they solved problems by having the students work problems on the board. We hoped that students should develop problem solving skills from these activities. Yet, students did not develop skill in problem solving. The students still could not solve problems when faculty changed the conditions slightly. They continued to depend extensively on sample solutions, even when these were shown to be inapplicable."
>
> "Despite individual professors' dedication and efforts to develop problem solving skill, "general problem solving skill" was not developed in the four years in our undergraduate program. Students graduated showing the same inability that they had when they started the program. Some could not create hypotheses; some misread problem statements. During the four-year undergraduate engineering program studied, 1974-1978, the students had worked over 3000 homework problems, they had observed about 1000 sample solutions being worked on the board by either the teacher or by peers, and they had worked many open-ended problems."
>
> " In other words, they showed no improvement in problem solving skills despite the best intentions of their instructors."

II. Approach

2.1 Didactics and Underlying Issues: In problem solving, students must map complex, multi-step sequences of mathematical operations. Among difficulties encountered are:

- The significance and usage of a law are not immediately apparent from its mathematical

formulation, and must be learned in addition to the laws themselves.
- From the large set of laws governing the domain, an applicable subset must be selected and adapted to the problem at hand (modeling).
- Problems are generally of a multi-variate nature, and the solution often requires solving simultaneously for all variables, e.g. using matrix methods.
- Problem solving is error-prone and frustrating. Each operation must be executed judiciously.

It is well established that the fundamental difference between novice problem solvers and expert problem solvers is that the experts have a well-organized, hierarchically arranged, easily retrievable knowledge base (Pankratius, 1990). Expert problem solvers in physics, for example, spend more time classifying problems than novices; however, the classification is made according to deep underlying principles and goes a long way towards solving the problem. Novices, on the other hand, classify problems according to their physical features and very quickly attempt to solve them using what may be called means-ends analysis. Thus, before one can select and apply a principle of physics (such as Newton's second law) one must know the principle and its relationship to other principles and laws in the hierarchy of the organized knowledge base. It would thus seem that difficulties with problem solving are not merely rooted in a lack of conceptual understanding, but also in the difficulty of building a systematic knowledge base.

2.2 Skills Development Software - The interactive GUI software described below has been designed specifically for the development of problem solving skills within an existing science course in engineering physics (Dynamics). For each section of a standard dynamics textbook (e.g. Hibbeler, 1992, Riley and Sturges, 1996, Beer and Johnston, 1997), problem solving units are being developed through which the student can learn and practice the methods to solve the problems of that section. Presently, units for kinematics (rotation and translation) and point mass dynamics are available, with additional units under development for inertial dynamics and energy methods.

The interactive GUI (Graphical User Interface) framework for the mapping, organizing, and conceptual solving of problems structures the solution process, shows possible solution paths, and guides the student towards developing insight into the nature of problem solving and to gradually develop experience and self-confidence.
Software Structure: The software consists of GUI analysis sections as outlined below, combined with symbolic computing software (choice: Mathcad). The software contains a database of problem assignments. The student can select a chapter topic and solve a problem within the chapter's context. Content-sensitive help explains laws, procedures, and their possible connections to the problem at hand. The software presents the learning topic systematically and requires the student to proceed to the solution in a logical sequence. However, except for completed sample problems, the software does not provide solutions. Students must execute each step on their own. By structuring the solution path and rewarding the student (in most cases) with the comparatively rapid and successful completion of an assignment, the software gradually develops the ability to approach and solve problems systematically.

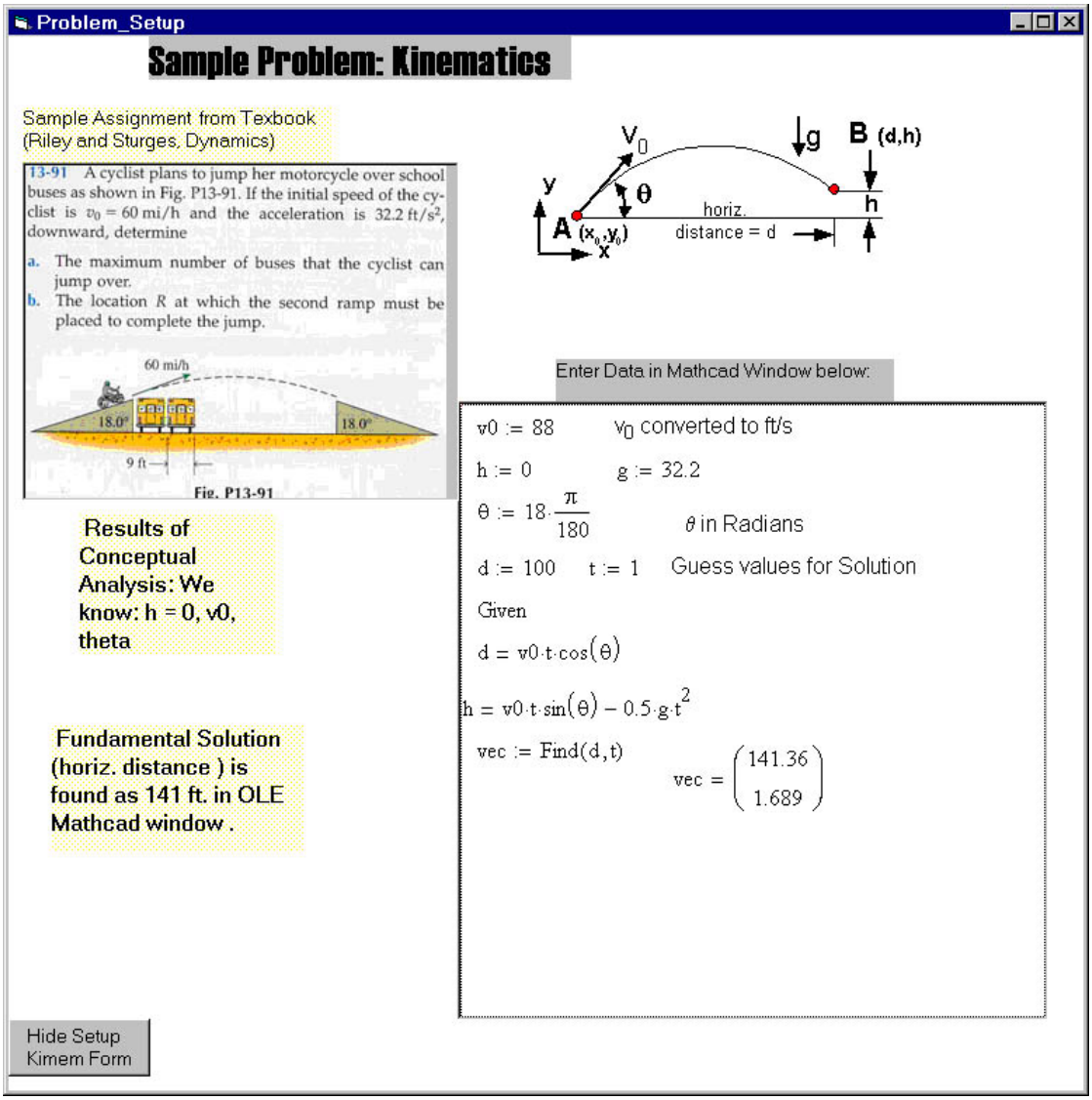

Figure 1 Problem Statement and Mathematical Solution of Kinematics Problem. The Mathcad *Solve* block finds horizontal distance traveled and travel time.
Student Entries: *Data and Parameters in Mathcad Window, Solve Block. Here, we solve for variables d and h after completing the conceptual solution of Fig. 2*

III. Application Example: Kinematics

For each problem, the software requires the student to follow the sequence below:
Step 1: Problem Definition: The *Opening Window* loads the selected problem text and its illustration. The student must identify known facts, e.g. constants, variables, geometry, desired result. Fig. 1 shows an example pertaining to curvilinear kinematics (quoted from: Riley and Sturges, 1996). In its upper right corner, the opening window also displays in a small graph the terms used in the analysis, and a Mathcad window where the student may enter data and equations.

Step 2: Conceptual Solution: A second '*Conceptual Solution Window'* (Fig. 2) is displayed next to the opening window of Fig.1. The equation set applicable to our kinematics problem is shown in the window labeled 'Equations' in Fig. 2, with five parameters initially undefined and positioned to the right of the Equations window. Known parameters are declared in the checkboxes labeled '*Check Known Parameters*' in Fig. 2, and the software moves icons for declared parameters to the equation's left side, thus providing visual feedback to the student. In addition, the 'Message Window' at the right of the checkbox offers context-sensitive assistance and status information

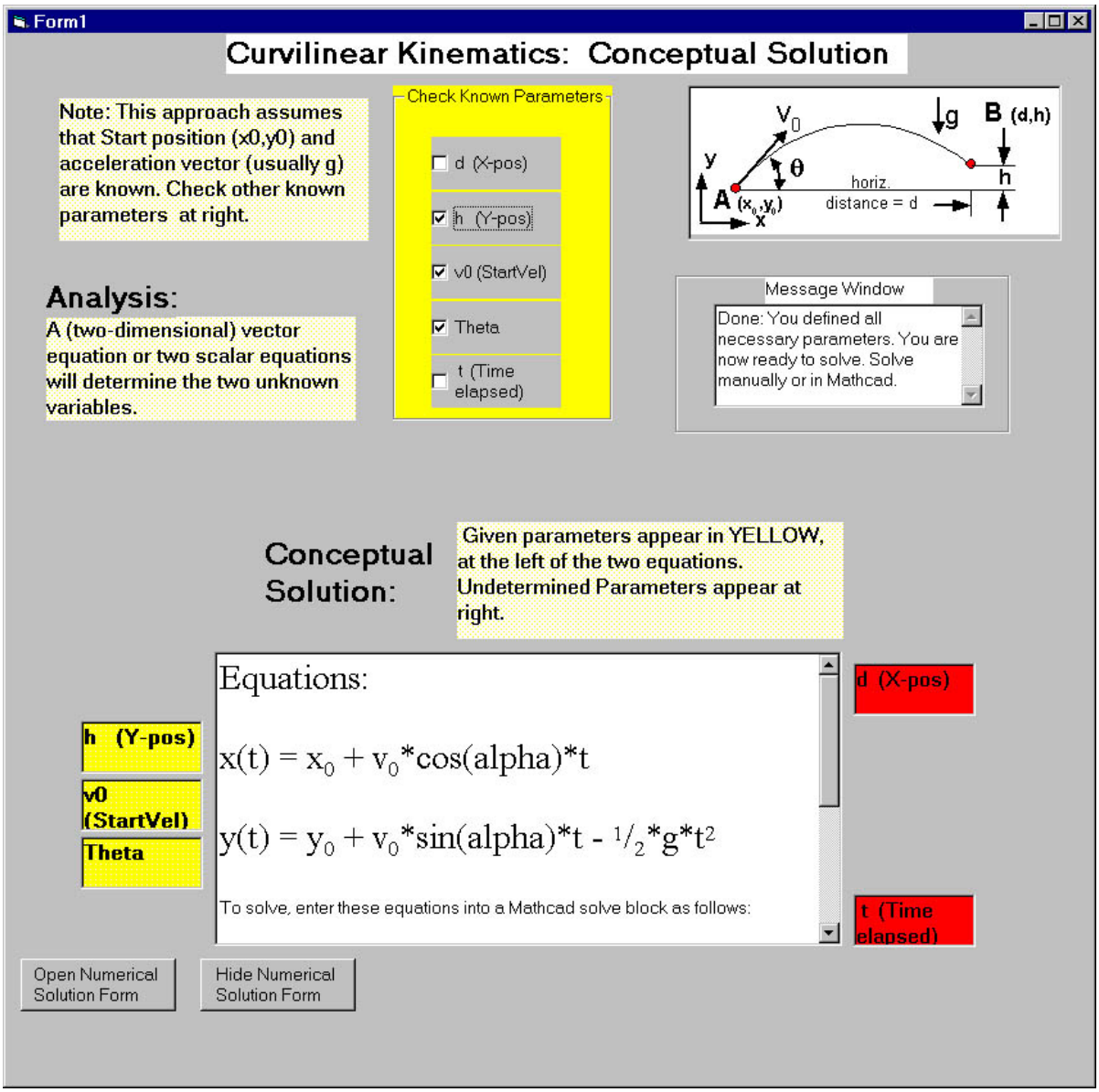

Figure 2 **Conceptual Solution Window,** Curvilinear Kinematics Example. The conceptual solution shown applies to the sample problem of Fig. 1.
Student Entries*: Given parameters are identified by clicking on checkbox. The message window and the position and color of each selected parameter provide visual feedback.*

through the conceptual solution process. The analysis is conceptually complete when the number of equations equals the number of remaining unknown variables. In the example of Fig. 2, only two scalar variables remain at the output side of the two-dimensional vector equation, i.e. the solution vector (time t and horizontal travel distance d) has been conceptually found.

Step 3: Adaptation: The student matches parameter names in generic laws (as formulated in '*Equations*') in terms of the pertinent, problem-specific variables and constants defined in Step 1. Example: In the upper part of the Mathcad window in Fig. 1, parameters in the *Equation Set* are matched with those in the problem statement of Fig. 1.
Step 4: Solution: From the conceptual map of Fig. 2 and from the set applicable laws (here the *Equations* in Fig. 2), The student is asked to copy applicable symbolic equations to the Mathcad window of Fig. 1. Here, the two equations cited completely describe the horizontal and vertical motions. Once the definitions, their correspondences with the generic equation terms, and the symbolic equations have been entered into the Mathcad window, Mathcad can solve for the unknown variables if a solution exists. The example of Fig. 2 identifies these variables as time t and

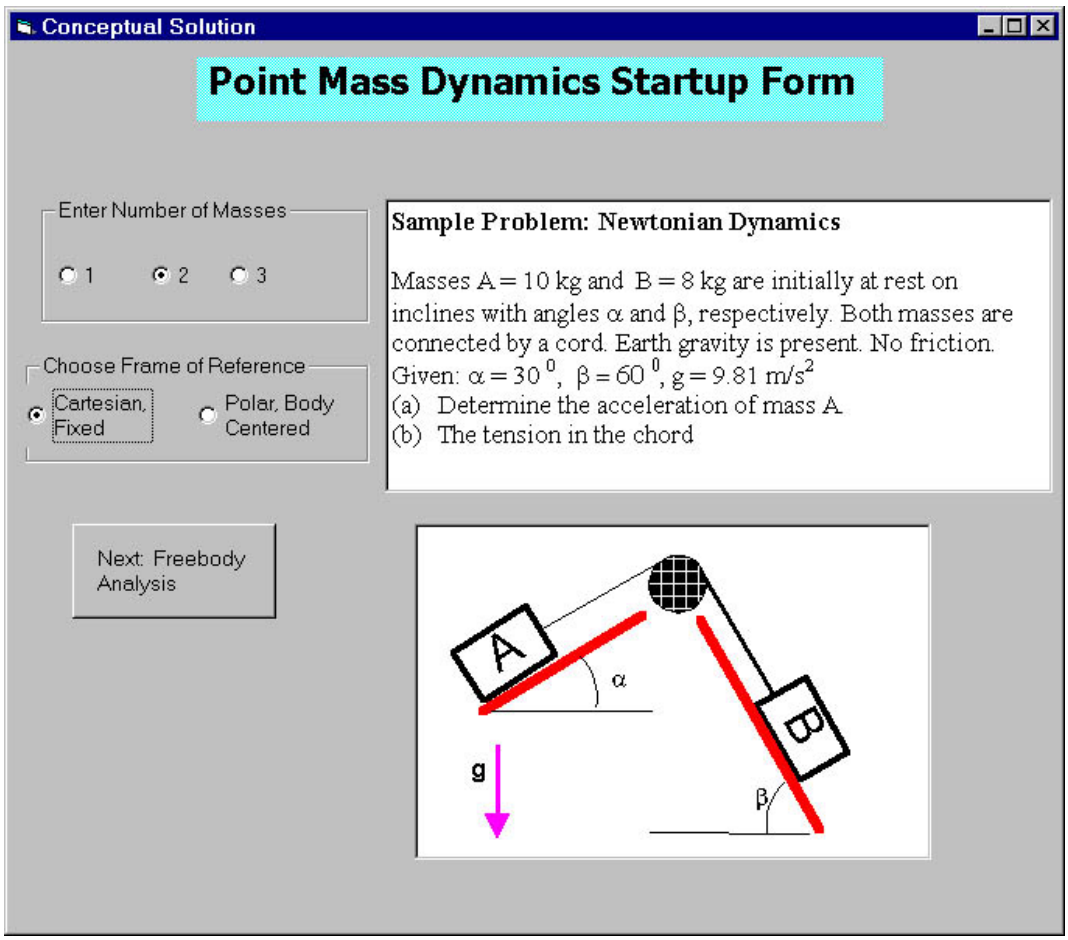

Figure 3 Newtonian Dynamics Example: Motion Analysis of Two Coupled Masses. Opening window for Conceptual Analysis.
Student Entries: *Make selections on Radiobuttons.*

horizontal travel distance d. The Mathcad command *Find(t,d)* then produces the symbolic and numerical results.
While the software environment presents possible solution paths graphically, it never mandates a specific path. The student working on an assignment must make all choices and selections, and will complete the assignment only by solving the problem on a conceptual level.

IV. Application Example: Point Mass Dynamics

Dynamic motion analysis typically requires the student to define a frame of reference, to perform a free-body analysis of the system, and to apply Newton's law to each mass element. In addition,

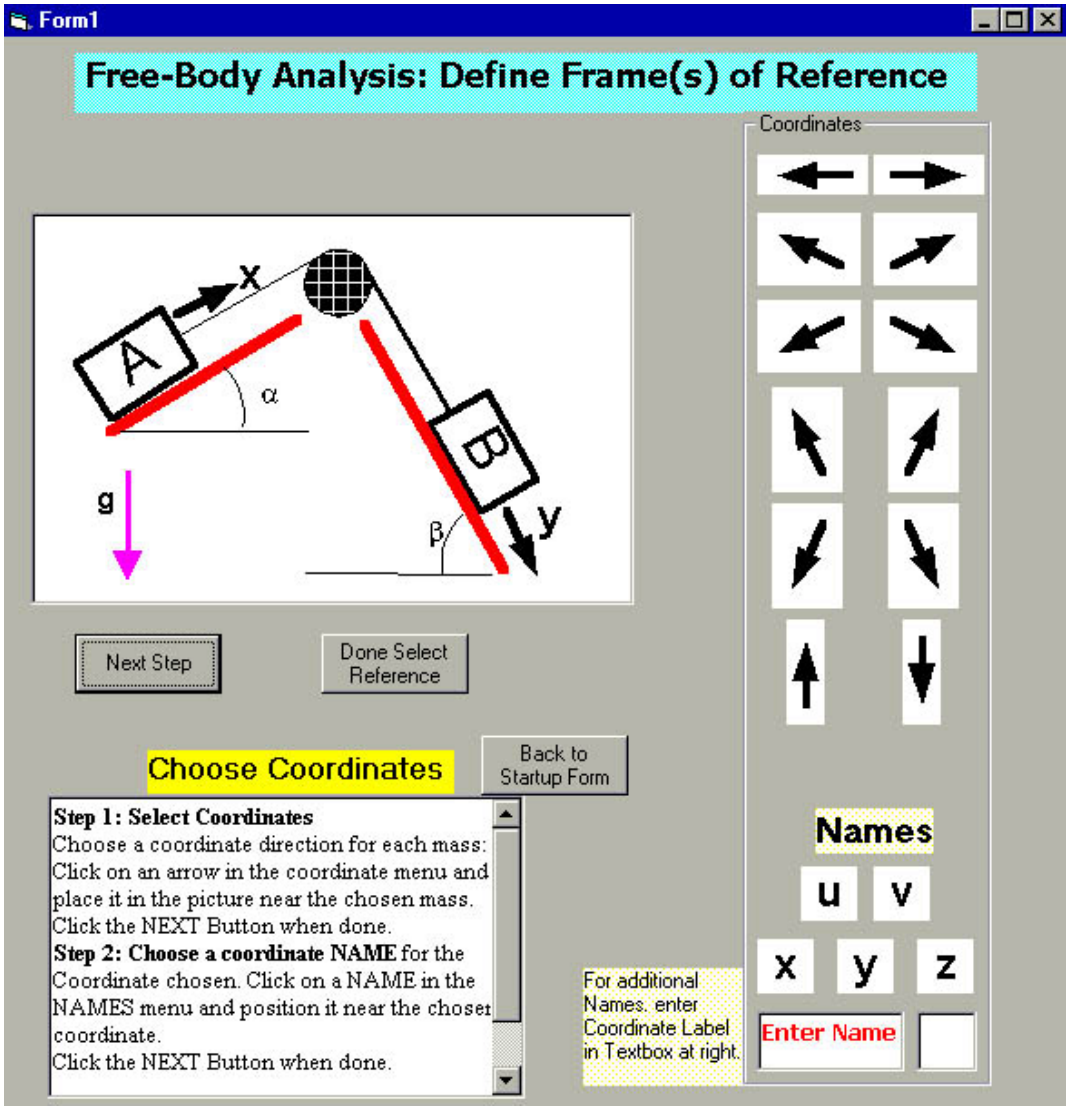

Figure 4 Newtonian Dynamics Example: Defining the Frame of Reference.
Student Entries: *Select frame of reference for each mass by dragging and dropping appropriate icons from the menu at left onto the picture.*

constraint equations must be found whenever the number of coordinates chosen exceeds the systems number of degrees of freedom.

Figure 3 shows a typical example. The problem is defined by a problem statement and an illustration. The latter serves as a backdrop for the free-body analysis. The software prompts the user to list the number of masses (used in subsequent windows to check for incomplete free-body analyses) and the frame of reference. Here, a Cartesian frame is appropriate. The student then proceeds to defining the appropriate coordinate system in a new window (Figure 4). By clicking on any of the arrow or *Names* icons on the right side of Fig.4, the selected item appears in the illustration at left, where it can be freely moved with the mouse or replaced by another user-selected icon as desired. The frame is complete after placing two arrows and two names onto the illustration. The subsequent two windows (Figures 5 and 6) require the student to add embedded

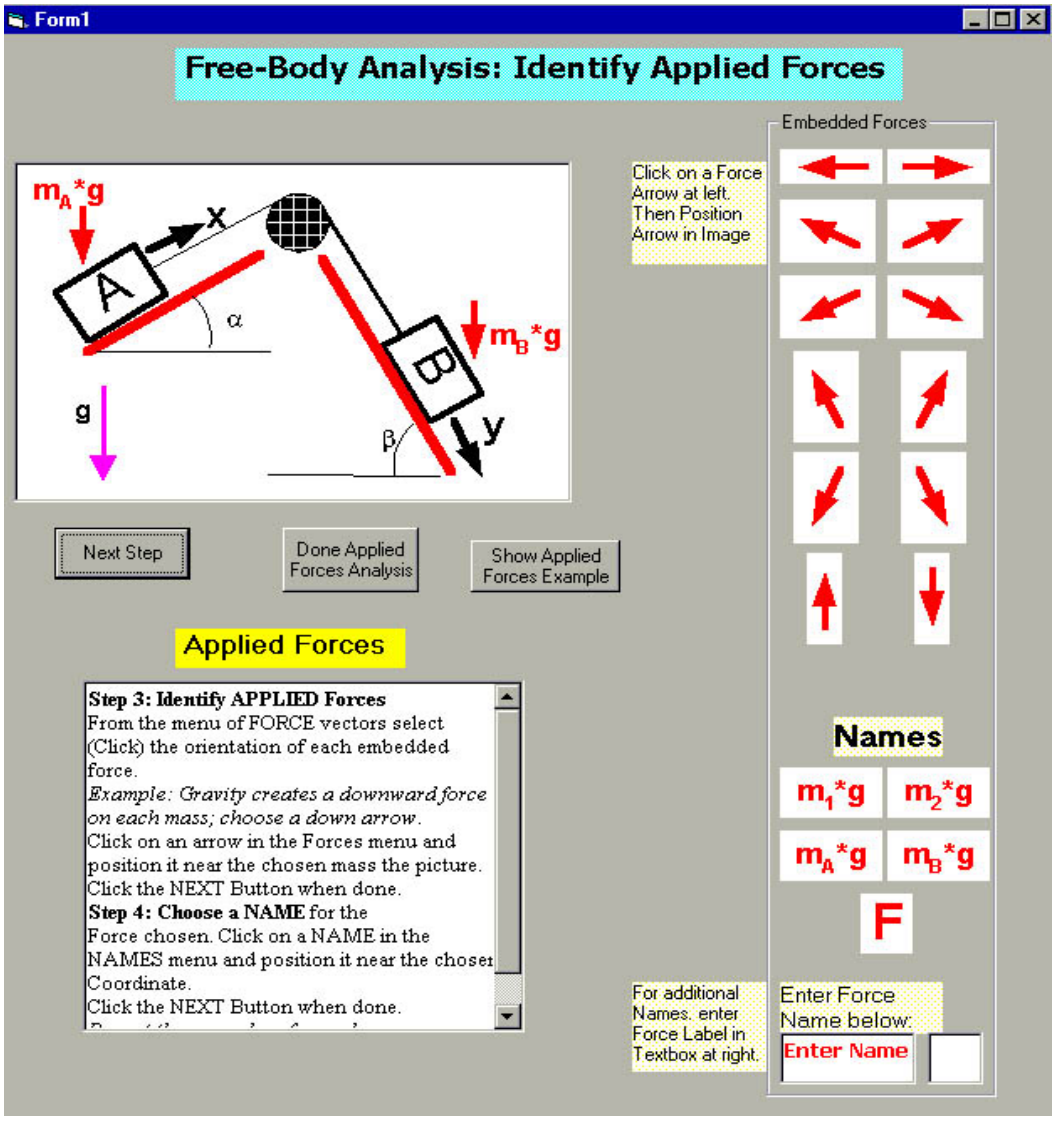

Figure 5 Newtonian Dynamics Example: Defining the Applied Forces
Student Entries: *Identify and name each applied force by dragging and dropping appropriate icons from the menu at left onto the picture.*

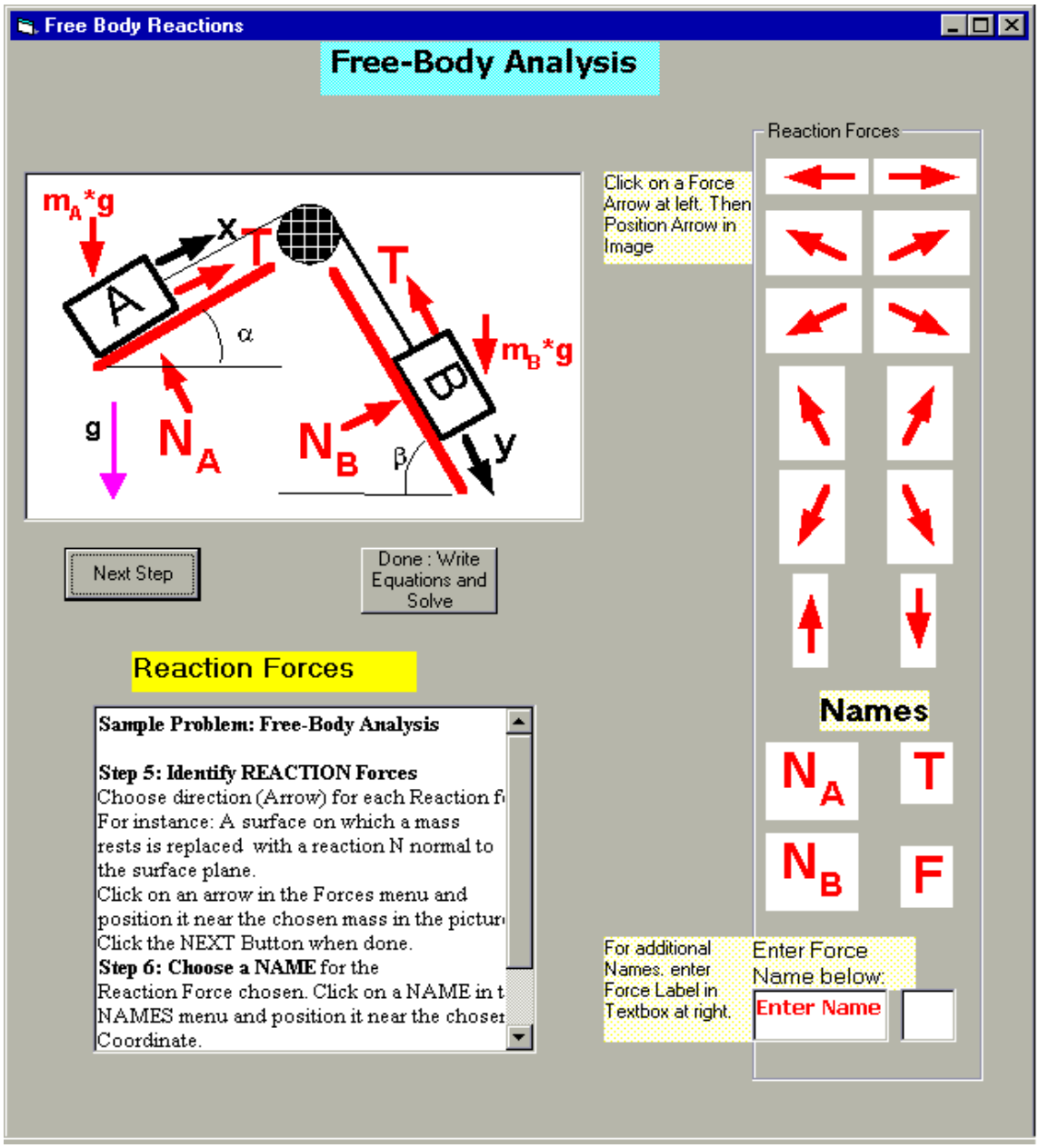

Figure 6 Newtonian Dynamics Example: Defining the Reaction Forces
Student Entries: *Identify and name each reaction force by dragging and dropping appropriate icons from the menu at left onto the picture .*

forces and reaction forces, respectively, to the illustration. The complete free-body diagram then guides the student to complete the analysis by applying Newton's law in each coordinate direction, and to define the constraint equation. The complete mathematical solution is shown in the Mathcad window of Fig. 7. The equation set consists of one equation each in x- and y-directions, respectively, and one constraint equation stating that the accelerations in x- and y-directions are the same, i.e. $\ddot{x} = \ddot{y}$.

Downloadable Demonstration Software – The interested reader is encouraged to examine the examples described here by downloading their code from the UNLV web site. Both programs can be found at: http://www.me.unlv.edu/coursenotes/egg207/meg207.htm
The code was developed with MS Visual Basic and runs on Windows 95 and later systems. The Mathcad window is not active in the downloadable versions of the software.

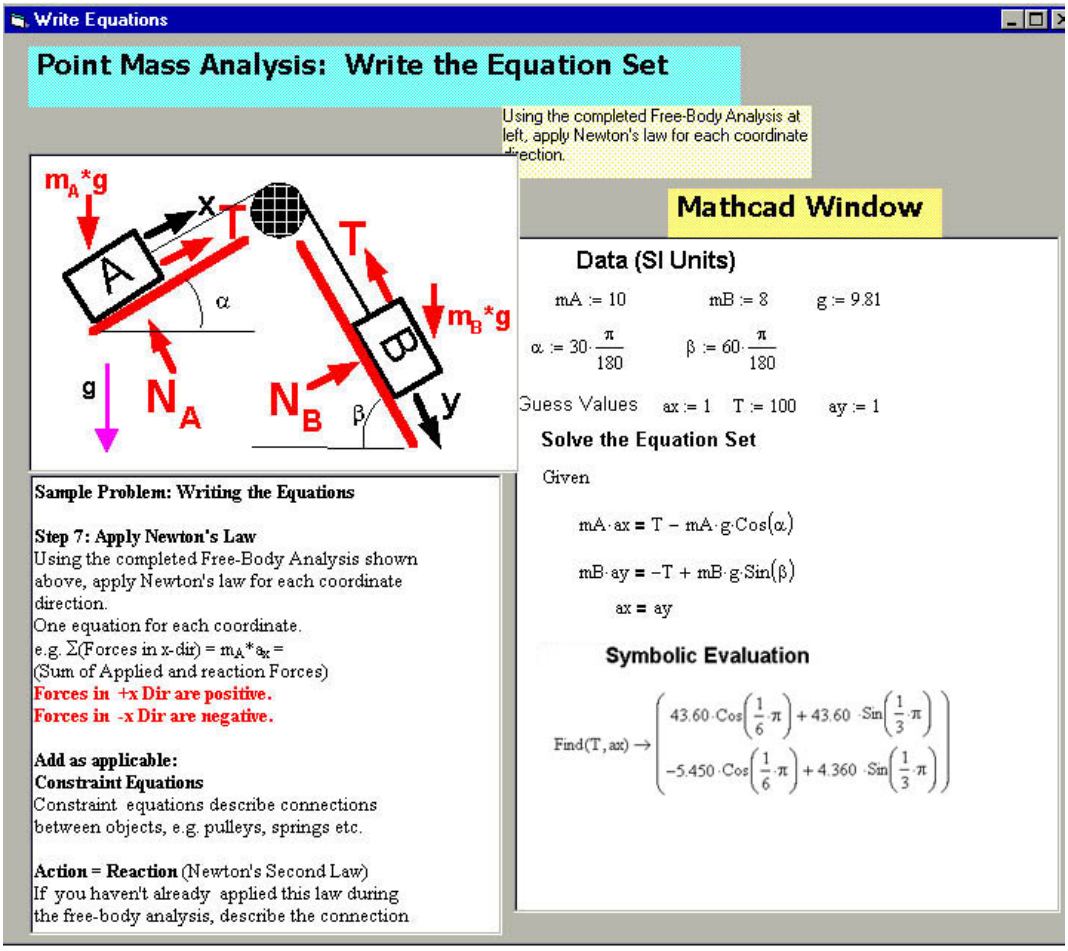

Figure 7 Newtonian Dynamics Example: Completing the Solution in Mathcad
Student Entries: *Data and Parameters in Mathcad Window, The equations in the Mathcad Solve Block (Newton's Law) express the sum of free-body forces for each*

V. Conclusion

A structured approach to training engineering students by means of a software-based concept for training problem solving skills in applied scientific problem solving has been presented. Beginning students regularly experience frustration and anxiety because they have not yet learned to organize newly learned material systematically. The extensive use of GUI tools provides convenient user interaction and continuous visual feedback concerning current status and further required solution steps. The GUI software environment structures the solution process and offers step-by-step guidance, yet requires the student to make all choices. The

student is thus encouraged and required to understand and actively practice the concepts leading to successful completion of assignments. The presentation of pertinent facts and concepts in combination with a GUI environment for their immediate application should facilitate their adaptation and practice and facilitate the training of students in practical scientific problem solving.

References

Barrett Steven F., D. J. Pack, G. W. P. York, P. J. Neal, R. D. Fogg, E. Doskocz, S. A. Stefanov, P. C. Neal, C. H.G. Wright, A. R. Klayton (1998) "Student-centered Educational Tools for the Digital Systems Curriculum," Proc. 1998 ASEE Annual Conf., Session 1620.

Beer, F.B. and E.R. Johnston (1997) "Vector Mechanics for Engineers," Sixth Ed. McGraw Hill Publ.

Bucciarelli, L.L., H.H. Einstein, P.T. Terenzini, and A.D. Walser (2000) "ECSEL/MIT Engineering Education Workshop '99: A Report with Recommendations," ASEE Journal of Engineering Education, p. 141-150, April.

Caillot, M. (1983) "Problem Solving Research in Elementary Electricity at LIRESPT," Problem Solving Newsletter, vol. 5, no. 3, p. 2.

Hibbeler, R.C. (1992) "Engineering Mechanics," Sixth Edition, McMillan Publ.

Hogan, N. (1991) "Teaching Physical System Modelling and Modern Control Together: The Need for an Integrated Approach," Proc. ASME 1991 Winter Annual Meeting, DSC-Vol. 36, p. 5 - 10.

McGraw-Hill and MSC Corp. (1999) "Beer & Johnston, Statics and Dynamics with Working Model," CD ROM to accompany Beer & Johnston's textbooks.

Pankratius, W. J. (1990). "Building an organized knowledge base: Concept mapping in secondary school physics," Journal of Research in Science Teaching. 27(4), 315-333.

Paynter, H.M. (1961) "Analysis and Design of Engineering Systems," MIT Press, Cambridge, MA.

Riley, W.F. and L.D. Surges (1996) "Engineering Mechanics: Dynamics," John Wiley & Sons Publ.

Scrivener, S., K. Fachin, G. R. Storey (1994) "Treating the All-Nighter Syndrome: Increased Student Comprehension Through an Interactive In-Class Approach," ASEE Journal of Engineering Education, p. 152-155, April.

Stein, J.L. (1991) "The Art of Physical System Modeling: How can it be Taught?," Proc. ASME 1991 Winter Annual Meeting, DSC-Vol. 36, p. 11 - 17.

Woods, D. R. et al. (1997) "Developing Problem Solving Skills: The McMaster Problem Solving Program," ASEE J Engineering Education, p. 75–91, April.

GEORG MAUER

Georg F. Mauer is a Professor of Mechanical Engineering at The University of Nevada, Las Vegas. Dr. Mauer is active in instructional computing, as well as in research on Automatic Control, Robot Sensors and Control. He graduated as a Diplom-Ingenieur from the Technical University of Berlin (West) in 1970, and completed his Ph.D. degree, also in Berlin, in 1977.