Published in:
Proceedings National Educational
Computing Conference '86
San Diego, CA
June 4-6, 1986

A SAMPLER

OF EDUCATIONAL SOFTWARE

AT CMU

David Trowbridge

Center for Design of Educational Computing
Carnegie-Mellon University
Pittsburgh, PA 15213
(412) 268-8527

Abstract

Carnegie-Mellon University is in the midst of installing a campus-wide network of high-function workstations to support educational computing. The system, known as Andrew, is the result of a joint project between CMU and IBM. Developers of educational software have now demonstrated several prototype educational applications running on the Andrew system. This paper gives some background for these developments and describes several examples of work in progress.

Background

Over the past three years, the Information Technology Center (ITC) has been constructing a hardware and software base for educational computing at CMU. A network file system and user interface software have been in place for about a year now. The feasibility of educational software development is amply demonstrated by a number of independent projects carried out by diverse groups across the campus. The Center for Design of Educational Computing (CDEC), a sister organization of the ITC, is supporting these endeavors.

The present computing environment has aspects of both commonality and diversity. Some commonality was established when the University made several important decisions at the start of the project. First, a powerful class of personal workstations was chosen, having 2 to 4 Megabytes of RAM, 1000-by-1000 pixel displays and CPU's capable of executing a million instructions per second. The establishment of a fairly high minimum level of capability meant that less energy would have to be invested in accomodating hardware dependencies.

Second, the University decided to use the Berkeley 4.2 UNIX operating system. Thus, work on the network and the user interface could proceed without first having to build an entire operating system. Also, UNIX brought with it a large collection of powerful tools, and good prospects for portability.

Third, it was decided to build a network which supported a single integrated file system, and a common user interface. Thus, users could log in at any workstation on campus and the system would look the same as from any other workstation. From the beginning, the ITC has set itself the goal of building a user interface which is both powerful for the experienced computer user and easily accessible to the novice.

The system also allows for considerable diversity among different campus groups. Various programming languages are available. At present, most development is done in C, the natural language of UNIX. Pascal, Fortran, and LISP are also available. In addition, a special authoring language, C-MU Tutor, is now being used for implementation of courseware materials.

Another aspect of diversity stems from the University's multi-vendor policy, in which workstations manufactured by several different vendors are available in public clusters and for sale in the campus computer store. Presently, the Andrew system runs on three different brands of computers: the Sun 120, the DEC VAXstation II, and the IBM RT PC.

Software Base

A window manager program mediates the display of all other programs (known generically as "client programs") in user-controllable multiple display windows. Several programs may be running at the same time; each program may use one or more than one window.

Several subroutine libraries are available to programmers who are building educational applications. There are subroutine packages for displaying multiple font text and simple graphics, for handling mouse and keyboard input, for creating and manipulating textual material, for making pictures and for building data-base oriented programs.

If desired, client programs in Andrew may be written using the window manager alone. The window manager subroutine library has functions for defining and selecting fonts, for displaying text, for line drawing and filling areas with patterns. At present, the window manager does not support color. While all the basic tools for displaying text and graphics are available through function calls to the window manager, higher level tools are also available to aid in the creation of more sophisticated programs. Among these are the base editor, the layout manager and the "Grits" subroutine libraries.

The base editor library supports manipulation of documents with cutting, pasting and copying text within a document and across documents (including to and from documents in different windows), scrolling, and on-screen formatting of text into various typesetting styles such as italics, bold, and special fonts.

The layout manager partitions real estate within a window using "layouts," separate rectangles (either tiled or overlapped) which may be used for different purposes. For instance, one layout might be used for a document with a scroll bar, another for a dialog with the student, another for animation. Each layout has associated with it procedures for redrawing it, handling mouse clicks within it, and displaying menus that are specific to that layout. Layouts may be cleared and redrawn independently.

Grits is a data base facility available at both a command-language level and as a subroutine library. It provides functions for storing, organizing, retrieving and displaying user data.

Examples of Educational Software

To give some of the flavor of educational software development projects underway, we describe here a few programs which have been developed on the Andrew system.

Graphical simulations for Physics[1]

Three physics programs illustrate the use of graphical simulations in Andrew (Figure 1). Each of these was written in the C language (a few hundred lines of code), using the window manager subroutine library.

Graphs and Tracks
W. J. Hansen, ITC
and David Trowbridge, CDEC

Student is shown graphs of position, velocity or accelleration for simple examples of rectilinear motion. By adjusting the post heights on an arrangement of sloping tracks, and selecting values for initial position and initial velocity, the student is challenged to reproduce the given graphs. The student sees a ball rolling along the tracks as the corresponding graph is plotted.

Optics Bench
Bruce Sherwood, CDEC, ITC and Physics

The student selects among a collection of optical elements such as convex and concave lenses, mirrors, and apertures, and places them along an optics bench. A screen is also positioned as desired. Using the mouse to point, the student indicates an origin for a spray of light rays which then pass through the elements. An image forms on the screen and is displayed as a circle shaded according to the intensity of the incident light.

Equipotential Lines
David Trowbridge

The student constructs an arbitrary arrangement of charged conductors having various fixed electric potentials. The program computes values of the potential in the intervening regions and draws lines of constant potential.

Dialog[2]

The C-MU Tutor language has proved to be valuable for authoring dialogs (Figure 2). It enables non-expert programmers to develop educational programs. In addition to the control structures of a general-purpose language, it supports many capabilities which are particularly germane to the creation of educational applications. These include picture drawing, graph drawing, animation, answer judging and "fancy text." C-MU Tutor, as an incrementally compiled language, makes possible an extremely short revision/review cycle. Even with large programs, an author need not wait more than about 10 seconds to see the effects of changes just made.

Expressions for Waves
David Trowbridge

The student is engaged in a dialog on concepts of waves, including ideas of amplitude, frequency and wavelength. The student is asked to enter simple mathematical expressions that describe oscillations. The aim is for the student to develop skills in writing expressions for sinusoidal traveling waves. At the end of the dialog, a brief quiz is given. Problems are stated in words, with numerical parameters generated at random.

Graphing Tool

A general purpose graphing tool shows the use of the Andrew base editor (Figure 3). This program is somewhat more complex than programs using only the window manager.

Grapher and Calculator
Bruce Sherwood
and Robert Schumacher, Physics

The student may enter arbitrary parametric equations in standard algebraic form, including first-order ordinary differential equations. The program plots whatever variables the student chooses. Initial values and ranges may be changed at will. In calculator mode, the program computes values of user-defined expressions.

Historical Research Tool[3]

Great American History Machine
David Miller, History

This program will provide a graphical user interface to a large body of data from U.S. censuses and election returns dating back to the early nineteenth century. A prototype has been created using 1850 census data (Figure 4). It displays maps of states and counties with fill patterns corresponding to values of census variables. The user may select variables from a large collection, and choose ranges of variables to best illustrate the relationships of interest. Thus it is possible to generate and test hypotheses about historical events interactively. It gives the historian and the advanced student a convenient, easy to use tool for exploring research questions. In addition, the program will be used to construct exercises for students in a sophomore-level introductory American history course.

This program uses the Grits database facility and an accompanying subroutine package for generating graphical displays of the data. Physical features such as rivers and mountain ranges will soon be incorporated into the maps as well.

Structural Design Tool[4,5]

Graphical Aid for Structural Design
Mary Lou Maher, Civil Engineering

This program is being developed to help architecture students understand structural engineering. It provides students with an environment for synthesizing structural systems (Figure 5). It uses several windows. The windows include a working window for combining structural components, a substructure window for defining new components, a plane window for displaying any 2D plane

in the building, and a window for viewing the 3D display of the structure. The program analyzes and judges the student's proposed structure for its overall feasibility. After the structure meets the test of feasibility, the student chooses specific structural elements from a database of building materials. The program then gives additional feedback on the integrity of the structure.

Intelligent Tutor for Electrical Engineering[6,7,8]

Dr. Thevenin
Sarosh Talukdar and Rostam Joobbani.
Design Research Center

A research project using expert systems technology, Dr. Thevenin addresses a common problem encountered by science and engineering students: Given a moderately complex circuit (Figure 6) consisting of a current source, a voltage source, an arrangement of resistors and two terminal points (n1,n2), find the Thevenin equivalent from the viewpoint of the two terminals (n1,n2). In particular, find the value of the Thevenin resistance between the terminals and find the open circuit voltage. The tutor has six components:

- A Problem Generator, capable of producing essentially endless streams of problems to fit specifications set by the student.
- A Problem Solver capable of solving all the problems the generator can produce with all the methods that the student must learn.
- An Interface that allows easy and natural communications between the computer and the student.
- A Monitor capable of tracking a student's attempts to solve a problem, identifying any errors, assessing their severity and assigning a numerical grade to the student's efforts.
- A Teacher who will be responsible for developing models of the student, diagnosing student needs, devising tutoring strategies and making explanations.
- A Supervisor to coordinate the other components.

The tutor is written in OPS5 — a language for expert systems -- and has about 850 rules. The user interface, written in LISP and C, uses the Andrew window manager.

Future Directions

The present programming environment, while extremely powerful for expert programmers, is not well suited to the non-expert. To generate large large amounts of educational materials, some different kinds of tools are needed. The current agenda has the development of authoring tools as one of the top priorities.

One approach, being taken by Bruce Sherwood of CDEC, is the development of an authoring language, C-MU Tutor. This language has a number of commands which are often needed in tutorials and dialogs (facilities for paging forward and backward, utilities for analyzing string, numeric and pointing device input, commands for drawing lines and circles, filling regions, drawing graphs, etc.) It is an incrementally compiled language, so there is practically no delay between revising a program and seeing the effect of that revision.

Another approach is the development of on-line design tools which may eventually make it possible for non-programmers to specify the requirements and detailed design of a lesson, without being concerned with problems of implementation. Then, implementation can be done by skilled programmers, who need not have any background in the subject matter. In addition, efforts are underway to explore the possibilities of making implementation partly automatic, using sufficiently powerful software tools.

The high-function workstation provides many new opportunities for building authoring tools, on-line design aids and implementation tools. Already, it is clear that a 1000-by-1000 pixel display can greatly enhance productivity: it allows the programmer to see the source code, the last version of the running program, run-time and compiler diagnostics, debugger information, and documentation on the screen, all at the same time. The large-scale network enables widely separated individuals to review each other's work, both as source code and as running programs. The combination of powerful workstations and a high-performance network seems to have the elements necessary for educational computing to come of age.

References

1. Trowbridge, D., "Using Andrew for Development of Educational Applications." Proceedings, University Advanced Education (AEP) Projects Conference, IBM ACIS, Alexandria, VA, June 1985.

2. Sherwood, B. "An Integrated Authoring Environment." Proceedings, University AEP Conference, IBM ACIS, Alexandria, VA, June 1985.

3. Miller, D., "The Great American History Machine." presentation at the University AEP Conference, IBM ACIS, San Diego, April 1986.

4. Zhang, Weiguang, "Interactive Graphic User Interface for 3D Structural Design," unpublished Masters thesis, Department of Civil Engineering, CMU, Pittsburgh, PA, March 1986.

5. Maher, M.L., Zhang, W. and Oppenheim, I., "Educational Software for Structural Engineering," presentation at the University AEP Conference, IBM ACIS, San Diego, April 1986.

6. Joobbani, R., and S.N. Talukdar, "A Knowledge-Based Expert System for Tutoring in Electrical Engineering," Proceedings, 8th International Computing Symposium ICS-85, Florence, Italy, March 1985.

7. Joobbani, R., and S.N. Talukdar, "An Expert System for Understanding Expressions from Electric Circuit Analysis," Proceedings, 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, August 1985.

8. Talukdar, S.N., and R. Joobbani, "A Knowledge-Based Intelligent Tutor for Thevenin Equivalents," Proceedings, University AEP Conference, IBM ACIS, Alexandria, VA, June 1985.

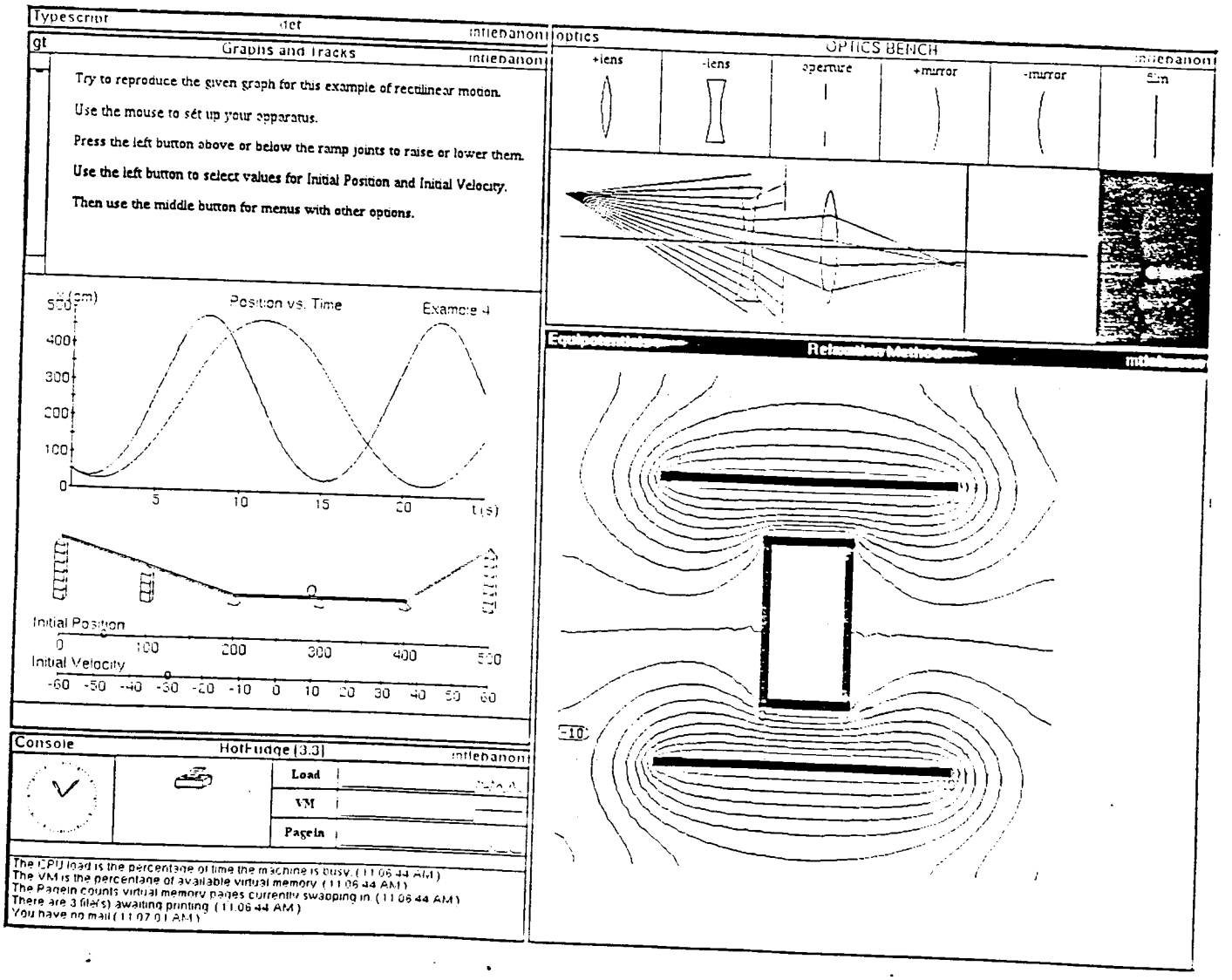


Figure 1. Graphs and Tracks, Optics Bench and Equipotential Lines: graphical simulations for Physics

waves /itc/itc/det/cmututor/waves.t mtiebanon

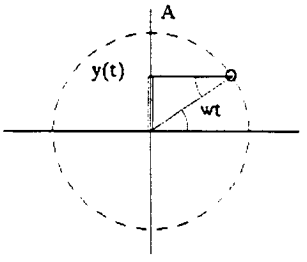
Suppose that at t=0,
the displacement is zero:
y(0) = 0
Write a function y(t),
in terms of A, w and t
that describes the subsequent motion.

y: displacement (cm)
A: amplitude (cm)
w: angular velocity (rad/s)
t: time since t=0 (s)

More
Contents
Introduction
Reference Circle
Oscillations
Traveling Waves
Quiz

y(t) = Asinwt

The vertical displacement, y
will equal the radius, A times
the sine of the angle, wt.



That's the right idea.
In the future, please use parentheses,
like this: Asin(wt).

Back Next

Figure 2. Expressions for Waves: a dialog

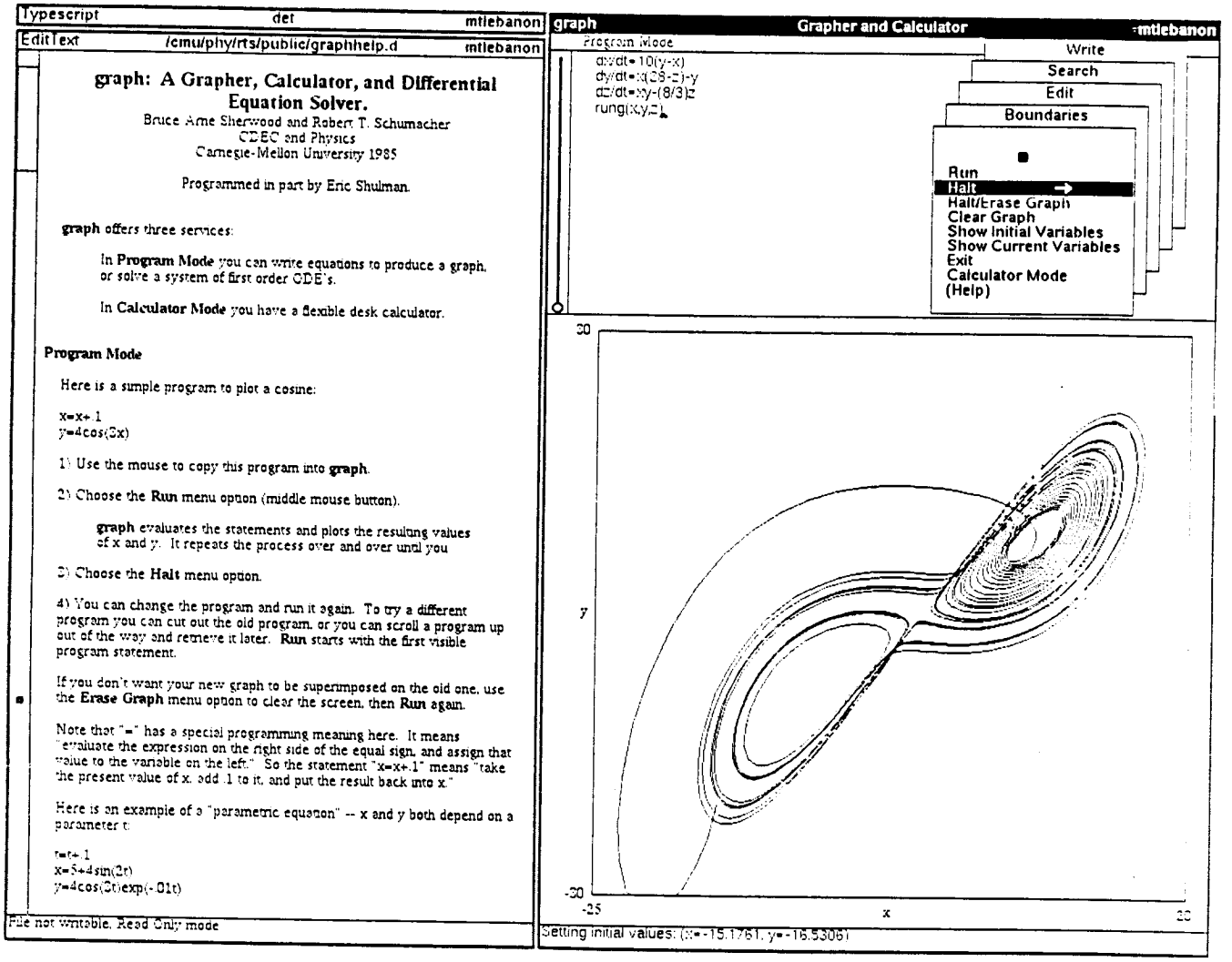


Figure 3. Grapher and Calculator: a graphing tool

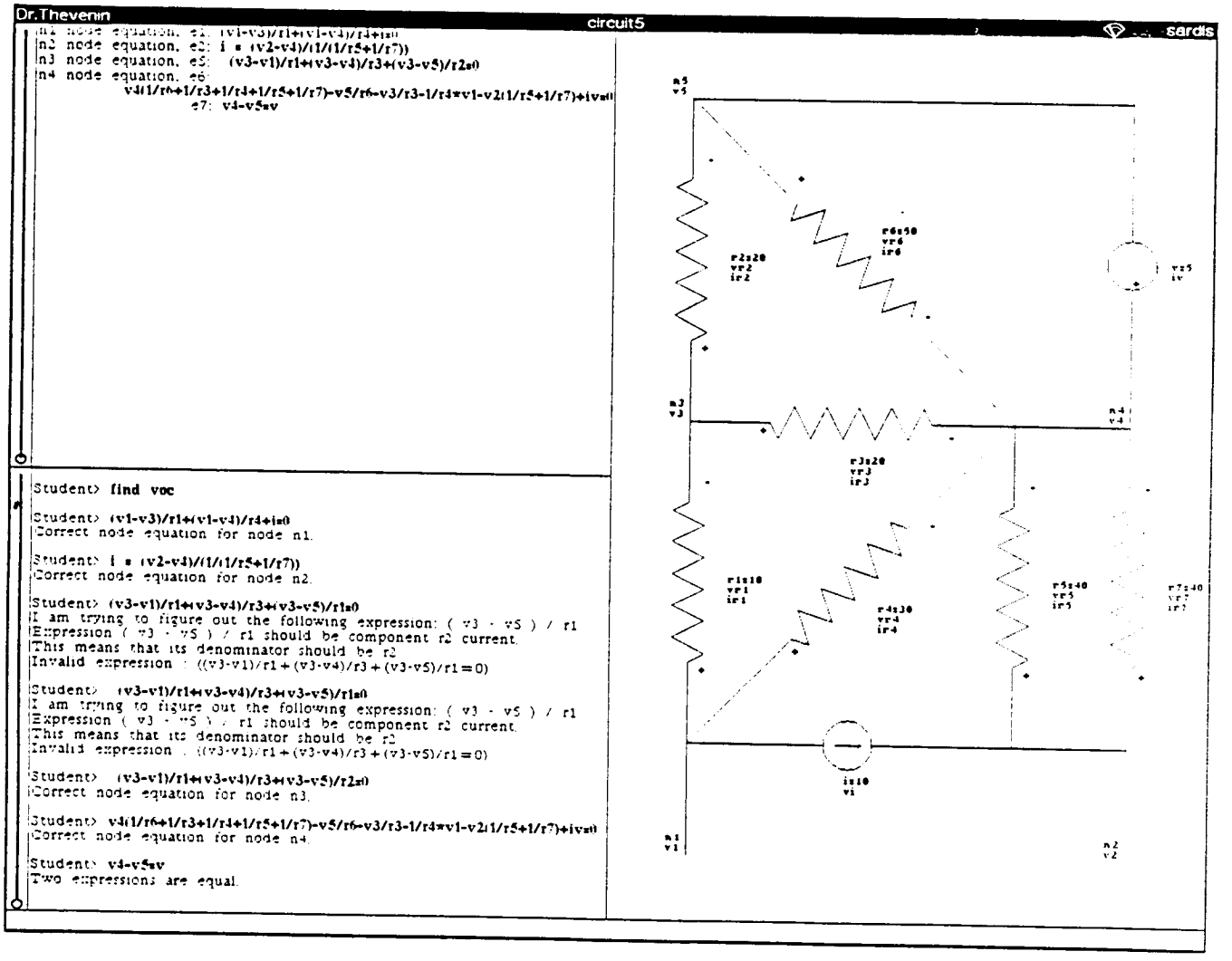


Figure 6. Dr. Thevenin: an intelligent tutor
for electrical engineering

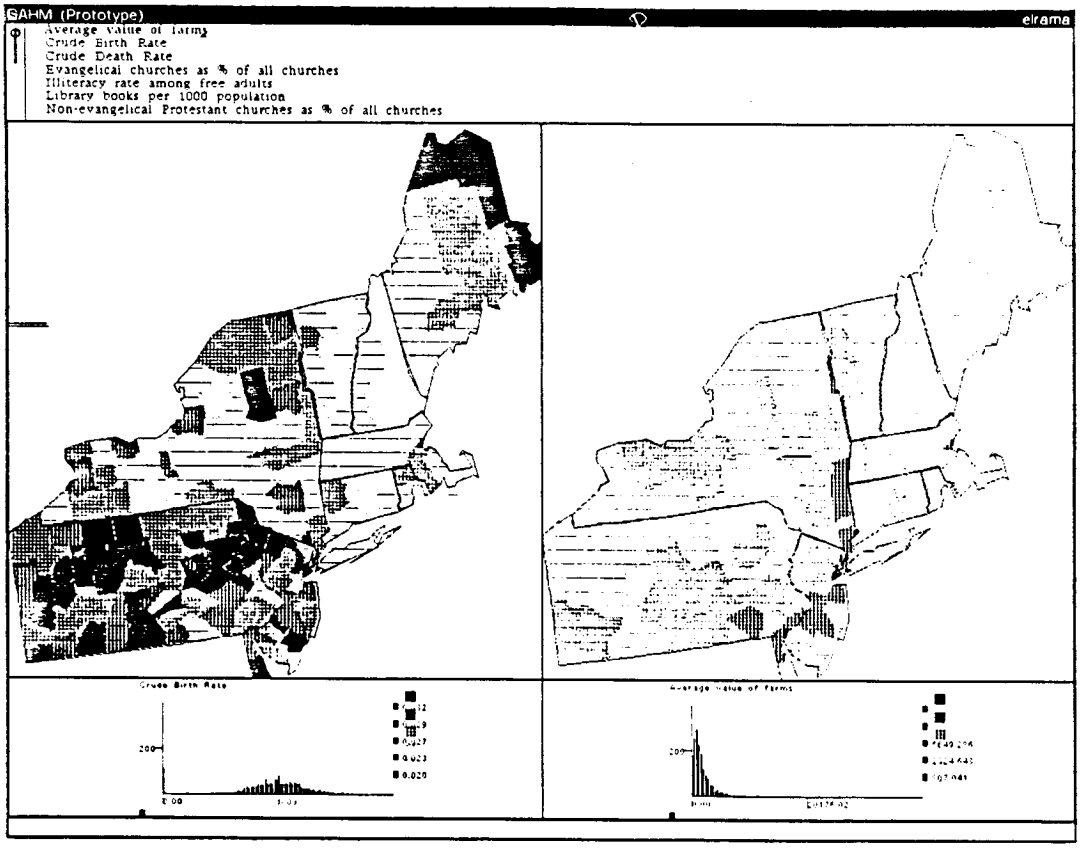


Figure 4. Great American History Machine:
a historical research tool

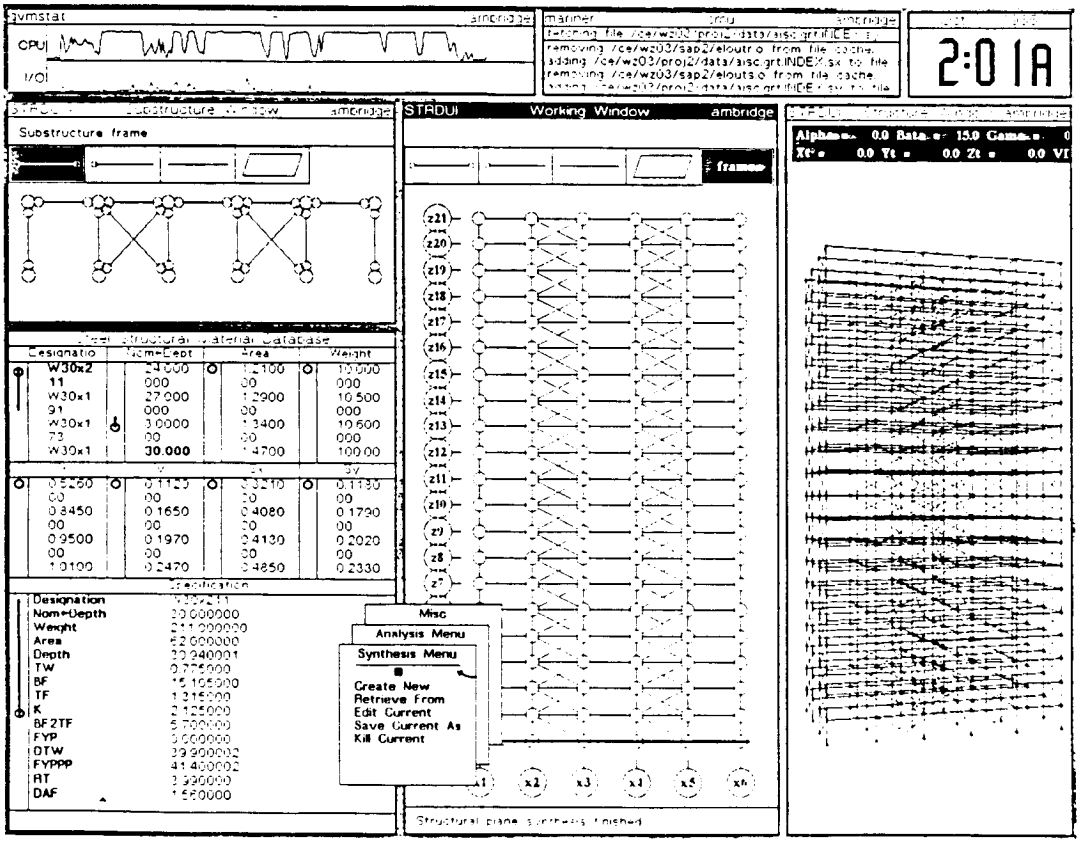


Figure 5. Graphical Aid for Structural Design:
a structural design tool