

Visual aspects of courseware engineering

J.W.A. Lanzing and I. Stanchev

University of Twente, The Netherlands

Abstract This contribution focuses on visualisation of information in courseware and in courseware engineering. An overview of advantages and disadvantages of visualisation is given, considering psychological and instructional, cross-cultural, and technical aspects of visualisation. Cross-cultural aspects are especially significant because of their strong relation with issues of courseware portability. Finally, an exploration of possibilities to improve courseware and courseware engineering tools by applying visualisation is included, with some practical examples of visualisation derived from authoring systems.

Keywords: Authoring; Authoring tools; Courseware; Courseware engineering; Information visualisation.

Introduction

The Persian prophet Mani (216-276 AD) was a gifted painter and presented his religious ideas in pictures to convince illiterate people. Nowadays computer enthusiasts are discovering the possibilities of graphical user interfaces. Presenting information in visual, non-textual form, is what is meant when we speak of visualisation. The non-textual symbols, pictures, graphs, images, and so on conveying the information will be called visuals.

In this paper we will discuss the possibilities for improving courseware and the courseware engineering process using visualisation.

The idea of using visualisation is not far fetched. Some people tend to make better use of text than of the same information presented in a visual form, others work the opposite way. Anyway, since repetition is one of the most practiced and effective learning techniques, displaying the same information in both modes of presentation could certainly produce positive effects on it's own, irrespective of the student's preferences.

Advantages and disadvantages of visualization

To be able to use visualisation to its maximum potential we need to understand what characteristics of visualisation cause it to be advantageous. This way we can estimate when visualisation will be the most beneficial.

Psychological and instructional aspects

The acceptance of individual differences leads to three possible approaches. The first approach is to make understanding as easy as possible for each individual by supporting both text and visual modes of presentation. The second approach is to use only the mode that fits the individual student best. Thirdly one can argue that everyone should be proficient in reading both text and visuals, so making it too easy would not stimulate the reader/viewer to train his abilities; consequently only one mode should be used at a time, but not necessarily the one that the student prefers. We expect that the solution to this dilemma lies in the nature of the instruction. The question whether or not to teach visual literacy is strongly related to questions of teaching common knowledge; it does not explicitly belong in the school curriculum but it is expected from good general education. Training and re-schooling of adults has less general and more specific goals and the designer of learning materials for this audience may therefore try to optimise learning outcomes by neglecting such goals as visual literacy and optimise the learning by teaching in the mode most suited for the individual learner.

Associative linking

A disadvantage of visualisation of the learning content is that most of the times the link between the symbol and its meaning are more or less arbitrary, based on associations of the mind. Because of individual differences one can never be certain someone else will have the same associative link and will therefore be able to interpret the symbols used correctly.

Formalization of symbols is, to a large extent, not the way to proceed. There is no reason to start six thousand years of language development all over again by creating a new set of hieroglyphs.

Another way to go would be to make use of visuals which have a high degree of logically deducible meaning. This has been done when a message was constructed to be sent into deep space to contact extra-terrestrials. But the interpretation of this message would take a very, very intelligent mind and a lot of time to test the many hypotheses necessary to decode the message. And then the message was not much more complicated than 'Hello, we are humans, this is what we look like and this is where we are'. Clearly this is not the way to go for information presentation in courseware.

On the other hand comprehension can be facilitated by adding context to the symbols. A lonely rubbish bin on a screen could mean a lot of things (time to put your rubbish bin out for collection, what you have just typed is rubbish, etc.), adding it to the rest of the desktop concept (icons for disks and files, the windows and menus) could easily lead to the conclusion that the rubbish bin may be used to dispose of files and discs!

Concreteness, realism and metaphors

Several instructional theories focus on the dilemma of either working from abstract to concrete or the opposite way. Romiszowski (1981) describes, for

instance, Bruner's classification of three levels of representation — enactive, iconic and symbolic, advancing from concrete to abstract (p. 173) — and Landa's concern for the rule/example sequence (RUL-EG versus EG-RUL) (p. 179). Examples are usually used to present information realistically, but text is an abstraction itself. To make the best use of examples they should be as realistic as possible. An obvious way to do this is to make use of visualization.

Metaphors are used when a topic has no intrinsic realism. To make the topic more accessible an analogy is found between the topic and a less abstract phenomenon. The metaphor may be visualised and therefore given more realism. In terms of Bruner this will enable students to gain iconic experience with symbolic topics. Direct manipulation (Shneiderman, 1990) of abstract information is possible by visualisation of the metaphor. An example of such possibilities is available in the Macintosh operating system and similar graphical user interfaces (GUIs), where files and directories, which have no physical existence whatsoever are moved, copied, deleted, etc. by moving their visual representations across the screen, using the mouse.

Parallelism of visuals

A major difference between text and visuals from a psychological point of view is the difference between the sequential nature of text and the parallel nature of visuals. For text a predefined direction of processing the chain of characters and words makes it a sequential mode of presentation. For visuals there is no predefined direction of reading, therefore it is possible to display items in such a way that it is easy to view the items 'at the same time'. A verbal description of a route you have to drive will only work if this is exactly the route you have to take. Imagine you have to find your way in a big city by looking it up in a file listing route descriptions of all possible ways to drive through your city. You will find it better to use a street map.

This extreme example shows the importance of the parallel nature of visuals. A less extreme example is the preference of authors to use visualization to give an overview of a theory, for instance. The details will be explained in text mode, the complex interrelations between the central concepts of the theory are more easily communicated using visuals. As soon as there are several ways to move through the information source, visualization becomes an important way of improving the surveyability of all the information.

Motivation

The motivating aspect of visualisation is clear. Programs that look nice are fun to work with. Of course one runs the risk of distracting and therefore decreasing the learning effects, but in general the more motivated the learner is the better he learns. Although motivation derived from the appeal of visuals is extrinsic, and less beneficial than intrinsic motivation, it can still be

advantageous: the general attitude towards the learning task, towards the program and towards computers are positively influenced by such feelings as the comfort of working with the program and the perceived user control (Hartley & Lovell, 1984).

Cross-cultural

Since the beginning of written language people have realized that communication using the arbitrary rules of language is not equally understandable for all. The chance of not understanding the associative relations between symbol and meaning increases when cultural differences between sender and receiver become larger. In general one can say that the fewer assumptions made about the rules of communication, the greater the probability of understanding. Communication by text requires language, and, from this point of view, language is nothing less than an enormous number of communication rules. By using text the sender assumes the receiver will be aware of these rules and will know how to apply them (quickly enough) to process the message. Language is part of culture; therefore often cross-language will also mean cross-cultural. Cross-cultural communication by text is therefore a problem unless sender and receiver can agree to use one set of rules (one language), provided they possess knowledge of a common set of rules (speak a common language).

Pictograms are iconic symbols. Using graphical symbols one can create an icon language, which is again a set of rules, but the associative link between symbol and meaning should make the icon language more culturally independent than text. For example, think about the pictographic language for traffic regulations or the icon language for organisational and functional focusing of areas and facilities in communication and meeting environments (airports, railway stations, conference centres, etc.). The discussion by Ossner (1988, pp. 8-11) follows the same lines of reasoning and reaches the same conclusions.

To increase the portability and effectiveness of educational software we have to investigate and to apply the full range of cross-cultural advantages of using visualisation (Lanzing, 1993). All these advantages have to be transformed into practical principles and guide-lines for software engineers and instructional designers involved in the development and implementation of educational software.

Universal understanding or equal unease?

There is however one less positive remark about the advantages of visualisation. The possibilities of visuals to express verbal information detailed enough to communicate whatever you would want to communicate are limited. This disadvantage may partially be circumvented in several ways previously mentioned in our discussion of associative linking.

What we have been trying to say using pictograms has been relatively easy to say; 'go there', 'exit over here', 'no smoking allowed', etc. To express

more complicated things using graphical communication might not make understanding easier, but at least everybody is equally unfamiliar with the symbols used, which could by itself increase mutual understanding. You may compare this to a situation where two persons having a different mother-tongue communicate the best when both use their second language, instead of just one person trying to speak the other's language. In this case more allowances are made for each other's mistakes.

Technical

Every computer is well equipped to display characters, but only a few of them offer enough built-in aids to make the programming of an interactive system on a graphical screen easy. Also the absence of one standard graphics adapter requires the programmer of graphical software to invest large efforts to make the courseware run on most computers. To overcome this problem one needs to use either an authoring system which supports graphical interaction or a library with adequate graphical functions or objects.

Without the right tools the development of a program that uses visualisation will take considerably more time and effort and therefore costs will be higher. Depending on the goals, the available tools and experience using them, the difference between the production of courseware in text and in graphical mode will decrease. In the end it might even become cheaper to produce graphical courseware due to the improved authoring tools.

The most important limitation in the design of graphical software is the skill needed to produce acceptable graphics. To guarantee an acceptable standard of graphical design one really needs the services of a professional designer. No student can be fooled to work with courseware that looks amateurish, however good the concept, the instructional design and programming may be, users look at the screen and if they see amateur graphics they will not be confident in the quality of the rest of the program. Of course this significantly increase the costs of courseware production.

Application of visualisation in courseware engineering

Visualisation in courseware engineering can contribute in two ways. The first and most obvious is visualisation in courseware itself, to improve the learning process. The second is the use of graphical representation, for instance in authoring tools, to improve the courseware engineering process.

Visualisation for learning

The advantages mentioned before indicate the value of visualisation for learning. Visualisation techniques from the domain of multimedia and graphical methods of visualisation can be used. The effort required to implement visualisation in courseware will depend largely on the development system used, but also on the availability of scanning hardware

and on the capabilities of the development team in producing visuals of high quality or on the availability of legally usable visual materials.

Depending on the learning content of the courseware, visualisation can serve several functions. Molitor, Ballstaedt and Mandl (1989) offer the following useful classification of functions.

- *Representation.* The representational function serves to transmit information, which is difficult or impossible with text. The use of the representational function of visuals depends largely on the instructional content because it has to be concrete, visible. Whilst a verbal description of the face of a person may be detailed enough to serve the educational purposes, it cannot be denied that a picture of a face is much easier to interpret and remember.
- *Organisation.* The organisational function provides an overview or macro structure of the text content. In this way visualisation will probably have the advantages of an advanced organiser, but with the extra advantage that it can be easily referenced. Knowledge navigation is one of the major problems in the educational use of large databases such as hyperdocuments used in discovery type learning packages. The overview provided by the macro structure may also serve as a kind of navigation tool. The macro structure or the 'map' of the program may be provided with 'buttons' or links to access parts of the program directly.
- *Interpretation.* The interpretation function helps the reader to understand parts of the text for instance by visual analogies and metaphors. An example of an interpretative use of visuals is the depiction of sets using Venn-diagrams or the use of trend displays to show the development of the value of a variable over time.
- *Transformation.* The transformation function offers a mnemonically useful form re-coding, being a kind of visual mnemonic. This is the function pictures often have in the 'keyword' or 'hook' technique' (Pressley, Levin & Delaney, 1982) for learning vocabulary in foreign language.
- *Decoration.* The last function, decoration, is used to beautify the text. The decorative function is a concern for Molitor, Ballstaedt and Mandl (1989), because they believe that it could produce negative effects. We think the decoration function itself increases motivation and will therefore produce a positive effect, if not overdone. But we agree that an illustration serving only a decorative function will probably be a distraction because the student will start to wonder about the educational value of the illustration and will spend his time trying to discover the deeper meaning.

Molitor *et al.*'s categorisation of visualisation functions should not be confused with a categorisation of visuals.

Visual aids for courseware engineering

Project management with Gantt diagrams (Federal Electric Corp., 1973) to clarify the complex relations and timing of different actions in the process of

courseware development and the design of programs with flowcharts, or Nassi-Shneiderman diagrams to clarify the program structure during the design phase are examples that illustrate the usefulness of visualisation for courseware engineering. But the use of graphical representation in courseware engineering should not be limited to representations on paper.

Visualisation has advantages that are specifically suited for improving the design of courseware. The nature of courseware sometimes requires non-linear program structures. The 'programmed instruction' method with its linear structure has proved to be de-motivating for most students, because they like to be in control of their instruction. Purely textual representation is no longer sufficient to represent a non-linear structure in a way that is clearly and unambiguously understandable. Several software publishers have realised this and started to design visual CASE (Computer Aided Software Engineering) tools, like visual authoring systems and even 'visual programming systems'.

Visual programming systems are systems where programming is done through visualised instructions. Many parts of system design can be visualised (Shu, 1988, p. 68).

- 1 system requirements,
- 2 program function,
- 3 program structure,
- 4 communication protocols,
- 5 composed and typeset program text*,
- 6 program comments and commentaries,
- 7 flow of control,
- 8 structured data,
- 9 persistent data, and
- 10 the program in the host environment.

Although this is an extensive set of parts of system design that can be visualised, most of these are not that relevant for the majority of courseware engineering projects. System requirements (1), communication protocols (4), data structures (8,9), and the program in the host environment (10) seem to be out of the scope of most courseware engineering projects. But program structure (3), composed and typeset program text (5) and the flow of control (7) are aspects of courseware engineering that may benefit from visualisation.

Tanimoto and Glinert (1986) distinguish four program elements and their classification is simpler and more useful for courseware engineering than that of Shu. This classification contains:

- 1 flow of control;
- 2 operations described by icons;
- 3 graphical display of data;
- 4 human-machine interaction.

Because of the very limited means of most educational institutions and the limited frequency of courseware use, courseware engineering should be as

* The way that the text appears to the (courseware) user should also include illustration.

efficient as possible. Therefore the aspects that can be usefully visualised will become increasingly important.

The visualisation of program structure has already been implemented in several commercial authoring systems. As an example we will describe and discuss the different ways in which *Authorware Professional* (formerly *Course of Action*) and *HyperCASE* have used visualisation techniques to facilitate easier program development (Authorware, 1987; Koenig, *et al.*, 1991). Both these systems allow direct manipulation of program components which also eases the learning effort and decreases the frustrations of designing visual software which frequently occur using higher programming languages (Shneiderman, 1990). The designer will therefore benefit from the advantage of the parallel nature of visuals.

Authorware Professional contains several possibilities for visualisation. The most apparent use of visualisation is in the so called 'program line'. This program line is a straight line from the top of a program creation window to the bottom, on which different icons can be placed. These icons represent different program operations or functions, like displaying, erasing or animating pictures or text on the screen, user-interaction through the question icon with different sorts of answer options, branching, etc. There are also icons to play external music files and to control a videodisc player. This way the user creates some kind of a flowchart which clarifies the program structure and flow of control.

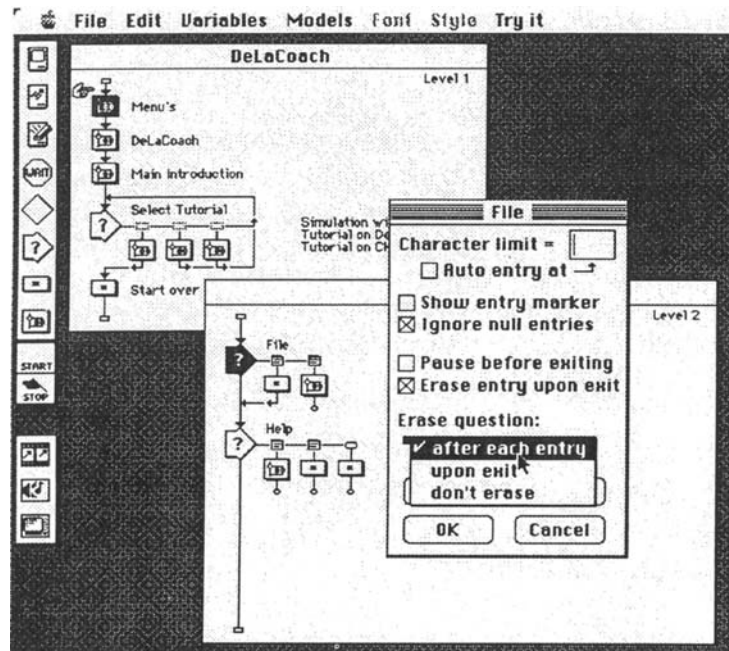


Fig. 1. Screen presentation of the *Authorware Professional* programming environment

Figure 1 shows the programming environment of *Authorware Professional*. It consists of: (a) the standard Macintosh menubar at the top of the screen, (b) two panels on the left with small icons (c) two windows with the program line and (d) a window called 'File' with a number of options.

The user-interface of both the authoring system and the courseware products developed in *Authorware Professional* makes use of the visualised interface of either the Apple Macintosh operating system or Microsoft Windows 3.0 on MS-DOS machines.

HyperCASE has so far only been implemented on MS-DOS. It works without Microsoft Windows but implements its own graphical user-interface. The representation of flow of control or program structure within *HyperCASE* is more parallel compared to *Authorware Professional*. In *HyperCASE*, a program is represented by a 'tree' of components, each component representing a function or a module. The components available are 'screen', 'window', 'module', 'icon', 'program', 'switch' and 'video'.

Figure 2 shows the programming environment of *HyperCASE* with (a) a menubar at the top of the screen, (b) a panel of icons of program components on the right and (c) two windows with the 'program tree'.

When at a certain moment in the program there is an possibility to branch, for instance when multiple buttons appear in a window, component icon connection points appear at the bottom of the window for each button to which other blocks can be linked. In this way pushing different buttons gives access to different parts of the program tree and the program will behave differently.

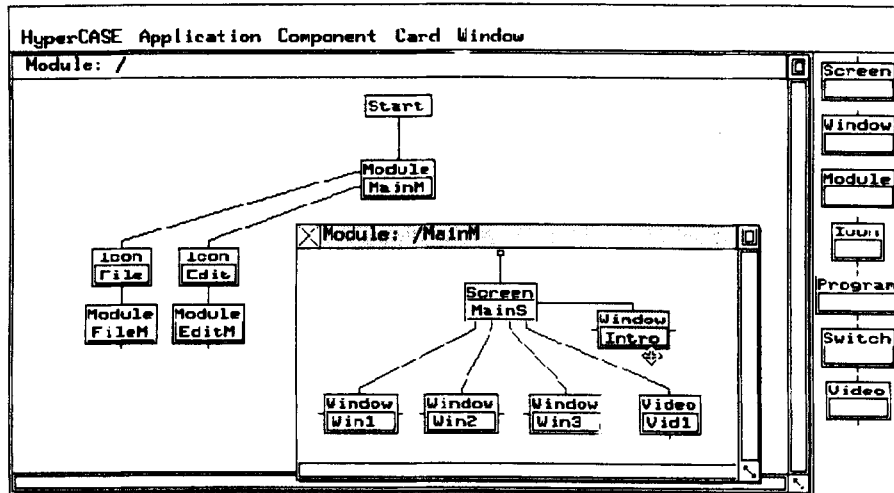


Fig. 2. Screen presentation of the *HyperCASE* programming environment

When using courseware developed in *HyperCASE* it is not necessary to close one window before moving to the next, so multiple windows can be on the screen at the same time. Switching between windows in *HyperCASE* means moving the focus of the execution mechanism from one component in the program tree to another (of course this is invisible to the student), allowing the student to follow another branch of the program tree. So there is an essential difference between this tree representation and an ordinary flowchart: 'because windows components stay open and 'run', a dead-end on a branch of the program tree does not mean that the application has terminated' (Koenig & Bryant, 1991). The main advantage of this representation of the program structure is that it is much easier to present information to the student simultaneously from different parts of the program. From an instructional point of view this allows the student to make his own combinations of facts more easily, so enhancing discovery learning types of instruction.

Adaptation and rapid prototyping

Advocates of authoring and visual programming systems often claim that 'non-programmers' should be able to develop their own software. However we do not consider programming by teachers who are not proficient in programming a good approach. The effort required to make good educational software is much too large to fit within the frame of lesson preparation. We do, however, recognise the desire of teachers to influence the instruction delivered by courseware products. Adaptability of courseware will become more and more important over time when teachers start to integrate courseware in their normal curriculum. As with ordinary school-books, teachers will want to make additions after some years of usage. Courseware will therefore be more readily adopted when making changes to content and learning paths (De Diana & de Vries, 1990) can be facilitated. Visualisation techniques may help to build systems, that enable teachers to make minor changes to existing courseware, such as removing a paragraph or adding one, replacing a picture, etc.

ResEdit for Apple Macintosh computers might serve as an example for teacher-adaptable systems. *ResEdit* makes good use of the unique characteristic of the Apple Macintosh computer that a program consists of two parts. The first part is the compiled code, the control structures. The second part contains data in a number of standard ways, such as strings, pictures, icons, cursor shapes, etc. This second part is known as the 'resource fork' and can be edited using the resource editor, *ResEdit*. *ResEdit* was originally developed for professional programmers, but soon became popular with many other Macintosh users. Unfortunately this tool has too many degrees of freedom and some of the resource items are too difficult to start with, therefore making a general tool like *ResEdit* too confusing to be used by teachers for lesson adaptation. A useful tool for teacher adaptation of courseware should include the possibilities to alter the learning route

through the content, to change, add and delete paragraphs and questions, to change the correct-answer alternatives and to replace pictures.

Rapid prototyping tools have become increasingly popular in recent years. They are systems that allow experienced programmers to make a mock up version of software to be developed to discuss and test the program before it is actually being programmed, usually in some higher programming language. The role of visualisation for rapid prototyping tools is obvious, because the major function of such tools is to produce programs that look like the final version of the courseware under development. It should give all the people involved in the courseware development process an idea of the functioning of the, usually, graphical user interface and the educational interaction.

Conclusions and directions for future research

The perceived importance of visualisation of courseware can be shown by looking at the number of authoring systems offering possibilities to make visual courseware. Tyre (1989) analysed the specifications of 31 authoring systems and all except two of them supported 'true' (better than character-based) graphics.

It is difficult to draw specific conclusions that prescribe some way of courseware design. Visualisation may produce positive learning effects, but should still be used cautiously. It is still better not to make use of visualisation in courseware than to do it incorrectly. The efforts required to visualise courseware and the quality of visualisation depends for the larger part on the courseware engineering tools used.

Courseware engineering tools can be enhanced using visualisation. Especially the visualisation of program structure and direct manipulation of program elements seem to improve efficiency, but care should be taken not to limit the degrees of freedom of the authoring system and therefore the possibilities for the programmer. This, as we have seen in the two examples of authoring systems, relates at least partially to the type of structural representation used.

For improving courseware design methodology it is necessary to study the relationship between type of software and the functions of visualisation, both for authoring such courseware and in the courseware itself.

References

- Authorware (1987) *Course of Action: reference manual*. Authorware, Minneapolis.
- De Diana, I.P.F. & de Vries, S. (1990) Adaptable educational courseware: an antidote to several portability problems. *Journal of Research on Computing in Education*, 23, 2, 225-241.
- Federal Electric Corporation (1973) *Netwerkplanning volgens PERT [Network planning according to PERT]*. Universitaire Pers, Rotterdam.
- Hartley, J.R. & Lovell, K. (1984) The psychological principles underlying the design of computer-based instructional systems. In *Instructional Software: Principles and*

- perspectives for design and use* (eds. D.F. Walker & R.D. Hess) pp. 38-56. Wadsworth, Belmont, CA.
- Koenig, J., Bryant, R. & Greenberg, V. (1991) *HyperCASE: The visual application development system*. Interactive Image Technologies, Toronto.
- Lanzing, J.W.A. (1993) *Visual aspects of the cultural portability of educational software*. [unpublished doctoral literature survey] University of Twente, Enschede.
- Molitor, S., Ballstaedt, S.P. & Mandl, H. (1989). Problems in knowledge acquisition from text and pictures. In *Advances in psychology, Vol. 58. Knowledge acquisition from text and pictures* (eds. H. Mandl & J.R. Levin) pp. 3-36. North-Holland, Amsterdam.
- Ossner, J. (1988) *Transnational symbols: The role of pictograms and models in the learning process*. Pre-DELTA Project TF III task e (paper 5).
- Pressley, M., Levin, J.R. & Delaney, H.D. (1982) The mnemonic keyword method. *Review of Educational Research*, 52, 1, 61-91.
- Romiszowski, A.J. (1981) *Designing instructional systems: Decision making in course planning and curriculum design*. Kogan Page, New York.
- Shneiderman, B. (1990) Direct manipulation: A step beyond programming languages. In *Visual programming environments: applications and issues* (ed. E.P. Glinert) pp. 317-329. IEEE Computer Society Press, Los Alamitos, CA.
- Shu, N.C. (1988) *Visual programming*. Van Nostrand Reinhold, New York.
- Tanimoto, S.L. & Glinert, E.P. (1990) Designing iconic programming systems: Representations and learnability. In *Visual programming environments: applications and issues* (ed. E.P. Glinert) pp. 330-336. IEEE Computer Society Press, Los Alamitos, CA.
- Tyre, T. (1989) Authoring packages continue to mature. *Technological Horizons in Education Journal*, 17, 3, 10-18.

CONFERENCE NOTICES:

New media for Global Communication:

multimedia, hypermedia and virtual reality (MHVR'94)

September 14-16, 1994 ICSTI Moscow

The aim of the conference is to provide a discussion and demonstration platform for researchers and practitioners involved in the theory, development, application and analysis of new computer media: multimedia, hypermedia and virtual reality. The conference will serve as a forum for the exchange of information and ideas between Eastern and Western scientists.

East-West Conference

on Computer Technologies in Education (EW-ED'94)

September 19-23, 1994 Crimea, Ukraine

This is the third in a series of conferences designed to report the best research in the field of Computer Technologies and Education and to provide opportunities for the exchange of information and ideas between Eastern and Western scientists.

For further information:

Dr. Peter Brusilovsky, plb@plb.icsti.su (fax: +7 095 943 0089)