ED 428 677 IR 019 338

AUTHOR	James, Jeff
TITLE	Practical Issues in Interactive Multimedia Design.
PUB DATE	1998-06-00
NOTE	7p.; In: ED-MEDIA/ED-TELECOM 98 World Conference on Educational Multimedia and Hypermedia & World Conference on Educational Telecommunications. Proceedings (10th, Freiburg, Germany, June 20-25, 1998); see IR 019 307. Some figures may not reproduce clearly.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	Artificial Intelligence; *Computer Assisted Instruction; *Computer Software Development; Constructivism (Learning); *Courseware; Foreign Countries; Higher Education; Hypermedia; *Instructional Design; Interaction; Models; Molecular Biology; *Multimedia Materials; Teaching Methods
IDENTIFIERS	Hong Kong Polytechnic; Interactive Courseware; Learning Environments

ABSTRACT

This paper describes a range of computer assisted learning software models--linear, unstructured, and ideal--and discusses issues such as control, interactivity, and ease-of-programming. It also introduces a "compromise model" used for a package currently under development at the Hong Kong Polytechnic University, which is intended to teach students principles of molecular biology, incorporating three activities--concepts, practice, and assessment. Difficulties involved in trying to build intelligence into the practice sectors is also discussed, particularly the irony of designing software which can actually disadvantage good students. Eight figures present: an electronic book; unstructured hypermedia software; a cartoon of "the intelligent tutor"; a mild constructivist model made up of three-part units; a sub-menu of the molecular biology project; an example content page; an example activity; logic depicting possible feedback; and a more incremental and fair routine. (Contains 11 references.) (Author/DLS)

Practical Issues in Interactive Multimedia Design

Jeff James
Educational Development Unit
Hong Kong Polytechnic University
Hong Kong SAR, China
Tel: +852 2766 6290, Fax: +852 2334 1569, E-mail: etjjames@polyu.edu.hk

Abstract: CAL (Computer Assisted Learning) software can be designed using the same principles of teaching which apply to the design of traditional teacher delivery. Software can be a simple page-flipping delivery system, or can incorporate components associated with active learning. Likewise, CAL software can vary according to the amount of control and "advice" given to the learner. Compromises can be made concerning the complexity of the educational environment produced with corresponding differences in the complexity of the programming used to produced the software. A program intended to teach principles of molecular biology is an example of software which attempts to incorporate components of both receptive and constructivist learning. It is designed to incorporate a certain amount of intelligence but in doing so, also introduces the question of fairness to different users.

1. Introduction

While computers continue to increase in memory capacity, speed, and multimedia delivery, the most critical factors in educational software design are concerned with the incorporation of *interactivity* into the CAL environment. Just as university teaching can vary from uni-directional, didactic lecture presentations to student-centred environments in which the learners inquire, discuss, discover, and generally contribute to their own learning, educational software can be designed which to follow a similar range of models.

This paper describes a range of CAL software models and issues such as control, interactivity, and ease-of-programming. It also introduces a "compromise model", used for a package currently under development, which is intended to teach students principles of molecular biology, incorporating three activities- Concepts, Practice, and Assessment. The paper will also consider the difficulties involved in attempting to build some "intelligence" into the Practice sectors, particularly the irony of designing software which can actually disadvantage the "good" students.

2. A Linear Approach to Software Design

The lecture style has a long-standing tradition in university settings. This one-way didactic teaching method is based on a model where a subject expert *presents* information to students. This information can be relayed verbally or in text form, with the result of this information being transcripted into students' notes. Advantages of this expository teaching style include efficiency (often with a teacher-to-student ratio of one-to-hundreds) and the relative ease of preparation and delivery. However, it can be argued that a good text could replace this teaching method with the favourable end result of less errors in the students' resultant written material (lecture notes would rarely be edited to the accuracy of a textbook).

The simplest and by far the easiest-to-program CAL software model has a linear, sequential structure. It follows a book metaphor where typically, the reader begins on page one and reads pages in order until the end of the book is reached. A computerised book may have the advantage of incorporating multimedia objects on each page, but the essential structure of this model as shown in [Fig. 1] is the electronic equivalent of a lecture. Control is determined by the order of the content, the user simply clicking a "next page" button. Presentation software such as PowerPoint could be used for creating and delivering such a book. James et al (1998) describe how beginning educational software programmers tended to design linear systems.

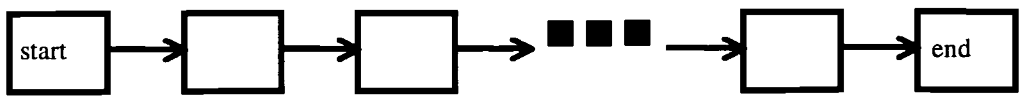


Figure 1: An electronic "book".

3. An Unstructured Model

The opposite approach to learning from the one mentioned above is one which gives students complete control of their information acquisition. Handing a student an encyclopedia or giving access to the World Wide Web are two examples of placing unrestricted learner-control tools in the hands of the student for a potentially rich, active, investigative environment.

[Fig. 2] shows graphically a typical hypermedia software model which, like the electronic book, is a simple point-and-click environment, but which allows complete user-control. The user can traverse any part of the structure, limited only by the buttons available at each node. A structure like this is still relatively easy to program using any authoring tool (such as HyperCard or ToolBook) which allows node (card or page) creation and linking between nodes.

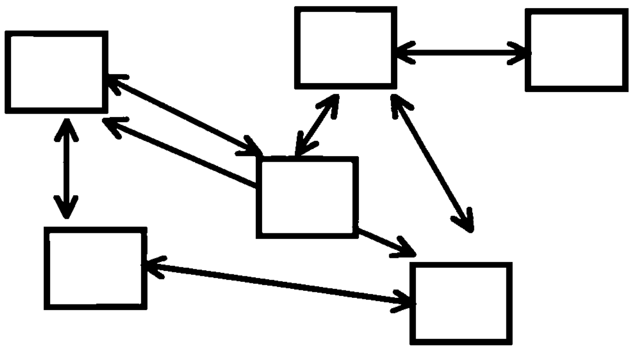


Figure 2: Unstructured hypermedia software

While the amount of control may be different for each of these teaching methods, the commonality is the *lack* of interaction between computer and user.

4. The Ideal Model

An idealistic teaching model would be based on a "tutoring" concept; the program would ask the user a question, accept *any* answer and give expert feedback, ask another question, etc. Unfortunately, this all-wise, infinitely-patient, intelligent tutor [Fig. 3] would require a level of artificial intelligence which would increase the programming complexity immeasurably.

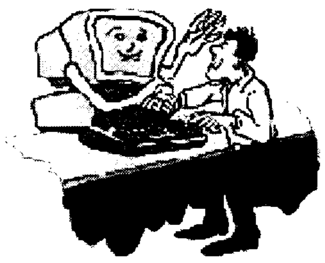


Figure 3: The Intelligent Tutor

It has long been recognised that there is more value in a "deeper" learning approach to education than that which is represented by the "expository" model. Laurillard (1993), in her comprehensive analysis of teaching in higher education, suggests that an active-engagement approach as advocated by educational psychologists Piaget and Bruner, is as relevant to university students as it is to younger learners. In practical terms, an environment such as this contains more learner-directed investigations (from the "hypermedia model"), combined with a significant amount of interaction (from the "tutor model").

In designing computer assisted learning tools, it is reasonable to attempt to design software which involves learners in the most powerful educational environment possible. Latchem et. al. (1993) advocate basing software design on sound pedagogical principles such as constructivism where students work in a knowledge-construction environment designed to be interesting, challenging, problem-oriented, and sometimes experimental.

The issue of control is fundamental to constructivist learning. In discussing the amount of control in CAL software, Schwier and Misanchuk (1993) caution that different learners have different amounts of control requirements. High-achieving students apparently benefit from having higher degrees of control whereas less-able and sometimes more naive students require more direction and structure to be administered by the computer program. It follows that educational computer software will be more adaptable to a range of students if it is flexible enough to allow different amounts of control by the users, although from a practical programming perspective, this is difficult to achieve.

Current interactive multimedia software often contains hypermedia knowledge structures. These are nodes of information (for example, screens of information), connected together by links (often in the form of buttons pointing to other screens). According to Muffoletto and Knupfer (1993), hypermedia enables students to explore information resources and can provide them with the opportunity to actively use newly-discovered information in the production of new knowledge. Jonassen et al (1991, p237) agree that "hyper environments" are "among the best examples of constructivist learning environments".

Assuming that more powerful learning environments as discussed above can be programmed into CAL systems, Csete and James(1996) have proposed a conceptual framework for designing an instrument to encourage learner control and engagement. This framework is in the form of a template with three components: (1) a window for the presentation of basic (and new) information, (2) a menu of learning options, and (3) a "safe" learning workspace for experimentation and the application of new concepts, principles, and skills in a non-threatening environment. Within the third template component, it is expected that learners will be encouraged to try new things, repeat activities, and explore "what if" scenarios. Rather than a completely free, open, and 100% learner-controlled environment, the viewpoint was that there will be a "guided discovery" situation where the program shares control with the learner. Lowyck and Elen (1991, p214) label advocates of this compromised position "mild constructivists",

> "Mild constructivists argue that, though self-regulated learning might be the ideal, most learning involves the interaction between internal (cognitive) and external (e.g. instructional materials) monitoring and that learning processes can be initiated both internally and externally."

A CAL program which has a modified hypermedia design, consisting of nodes connected by links including components of guidance from and interaction with the program, is consistent with this compromise.

[Fig. 4] shows a conceptual model of a CAL program which has a general sequential structure of units, but within each unit there is (1) information delivered to the student, (2) an environment where the student can investigate, and (3) "intelligent" interaction between the student and the program. A model such as this attempts to incorporate constructivist opportunities and interaction within a guided-delivery system.

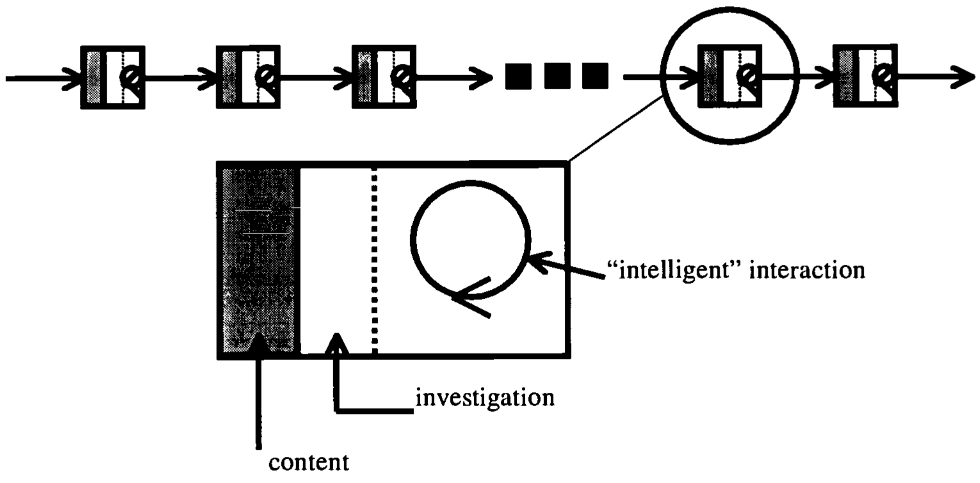


Figure 4: A "mild constructivist" model made up of three-part units.

The content component is similar to a sequence of one or more nodes in the "lecture" model, each node explaining a concept using text, diagrams, pictures, animations etc. The investigation component can vary, but typically will be an activity where the students make decisions based on what they understand of the content. It should be noted that activities such as these often require sophisticated design and programming skills.

The "intelligent" interaction consists of a "dialogue" which can be programmed using logical structures (e.g., if-then-else statements). This component addresses the concern expressed by Latchem et. al.(1993) that a majority of interactive multimedia systems are a "shallow" result of a simple menu-and-selection philosophy of interaction. They also see a potential for a more Socratic environment, posing questions and challenges. This view is supported by Larkin and Chabay (1992, p170) who propose, "Never tell when you can ask".

Larkin and Chabay caution, however, that designing interactive programs is a difficult task- the programming logic and possible pitfalls can be seen in the following example. The result of a well-designed comprehensive dialogue can be worth the effort, the alternative being what Bates(1995) describes as a lack of flexibility leading to student frustration when reasonable responses are not "recognised" by the computer.

5. Example Case

In early 1996, the Hong Kong Polytechnic University allocated Hong Kong University Grants Commission funds for projects which implemented interactive multimedia technology in teaching and learning situations at the University. A sub-project of one of the successful grant applications is called Molecular Biology which is intended to develop a teaching tool to aid third-year students to learn DNA technology. The philosophy behind the design of this project is to create CAL software which is consistent with the "mild constructivist" model given above. It incorporates high-resolution graphics and animation, and was authored using Micromedia Director. [Fig. 5] is a snapshot of a sub-menu of the program.

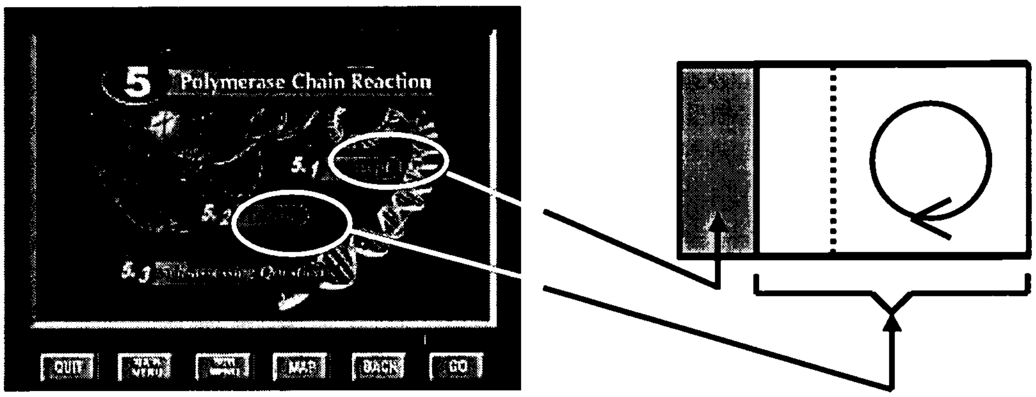
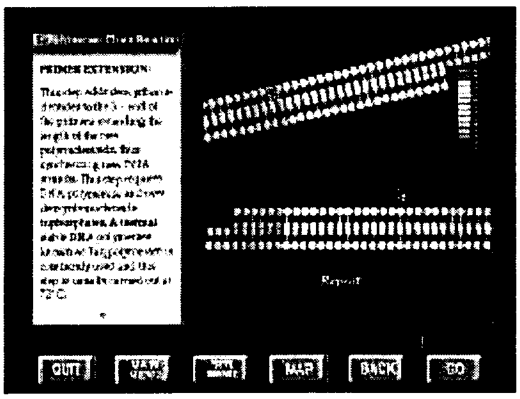


Figure 5: A Sub-Menu of the Molecular Biology Project.

The Concept part delivers subject content to the student; an example page is shown in [Fig. 6a]. This particular page gives an animation demonstrating how temperature affects DNA strand separation.



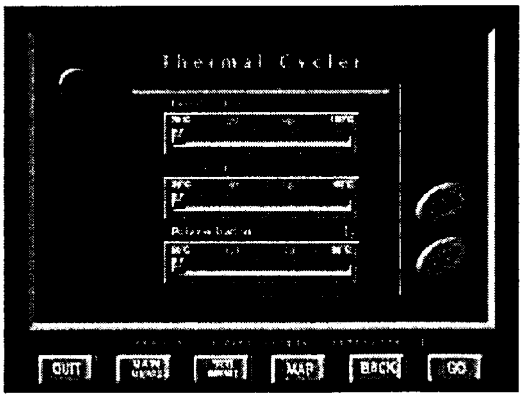


Figure 6a: Example Content Page. **Figure 6b:** Example Activity

The Practice option provides a problem-solving environment with appropriate user-computer interaction. [Fig. 6b] displays an activity where the student is expected to specify three temperatures using temperature slides. When the temperatures are set and the Start button pushed, the computer will report appropriate feedback to the student. This feedback is in the form of a written report. [Fig. 7] is a diagram displaying the logic the computer program will follow in determining feedback to the student's temperature settings (D, A, and P). Each of the seven possible reports will be a unique "intelligent" response to the temperatures which the student specified (note that in the actual package there are sixteen reports available, rather than seven). If the user sets all three temperatures correctly (D = 95, A is between 40 and 50, and P is between 70 and 75 degrees), Report 5 is given and control then leaves this routine, progressing to the next step in the program. If incorrect combinations of temperatures are specified, an appropriate report is given and the user has the opportunity to specify a new set of temperatures.

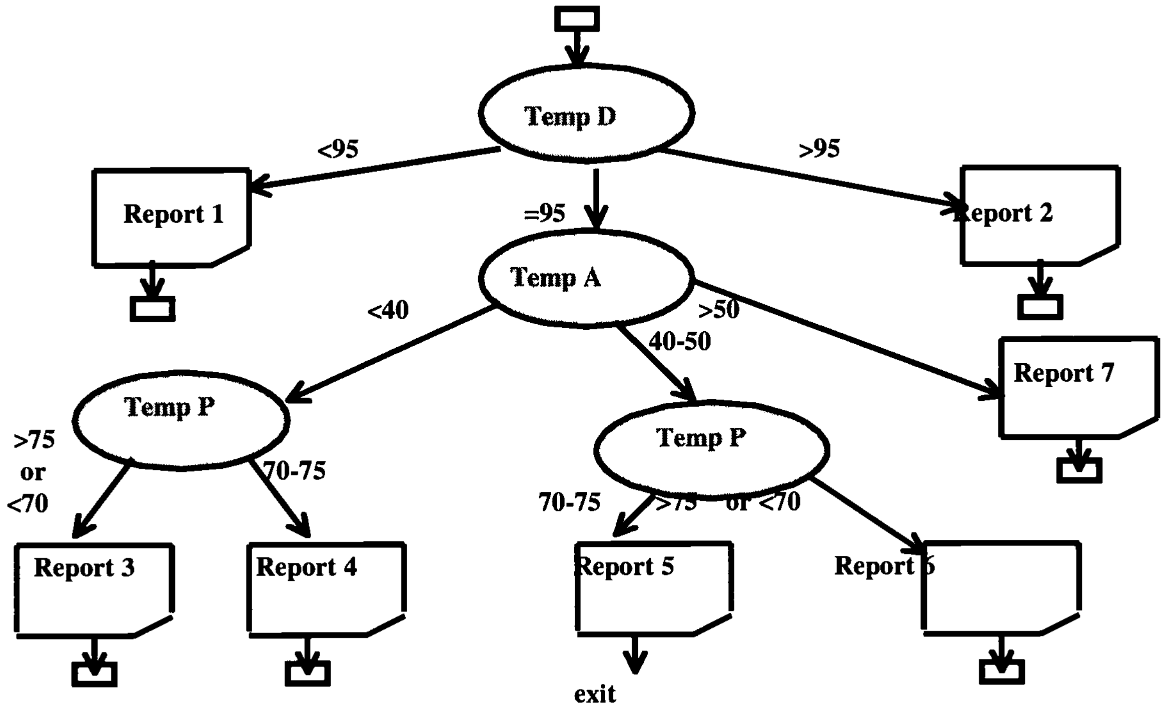


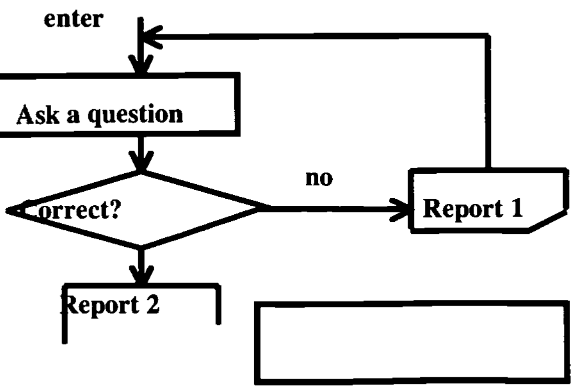
Figure 7: Logic Depicting Possible Feedback.

While the logic described above appears to be reasonable, the designers have a dilemma: assuming that the six "incorrect" reports are information-rich for each respective combination of temperatures, it could follow that the "good" student who gets the correct temperatures upon the first (or even second) attempt would have learned something else from some of the incorrect reports. If this is so, then the student was *disadvantaged* by leaving the routine following their correct input. Designers are cautioned therefore, that perhaps a more "fair" routine would look like that shown in [Fig. 8] where all students have the opportunity to gain from a variety of situations, in this particular case, from investigations arising from different incorrect temperature settings.

Example questions which would generate reports could be:

Report 1: What would you expect to happen if the Denaturation temperature was set to less than 95?

Report 2: Good! Did you also know that "de-oxy" means "removing oxygen"?



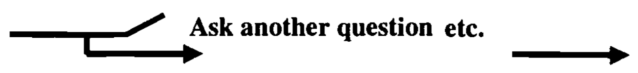


Figure 8: A More Incremental and Fair Routine.

The inevitable tradeoff with this structure however, is that all students must go through all possible settings, increasing the time of sitting by most students and more importantly, minimising the individuality of program runs.

In the spirit of the active learning approach however, guidance could be given in a more open-ended manner; for example "Investigate a wide range of temperature settings" could result in a variety of reported results as in effect, the student would be instructed to traverse the whole structure of [Fig. 7]. However, this would favour Schwier and Misanchuk's (1993) "high-achiever" students. A designer wanting to incorporate an "all things for all people" philosophy could cater for both types of students by adding another selection page with two buttons, one labelled "Sequential path approach" and the other "Open-ended approach".

6. Conclusion

Just as there are different teaching methods and styles, CAL software can be designed to reflect different teaching philosophies. Active learning approaches can be incorporated into computer programs to promote more effective learning, but there will be a tradeoff between more powerful learning tools and programming practicalities. Furthermore, it is not possible to build the "perfect" CAL program because of the differences between learners and hence, their learning styles. Perhaps the most serviceable educational software designs are accomplished by incorporating both guided and self-discovery components. In all cases however, interactive software designs must include components of intelligence and fairness to all learners.

7. References

[Bates 1995] Bates, A. *Technology, Open Learning and Distance Education* , Routledge, London.

[Csete and James 1996] Csete, J., and James, J. "The guided discovery template: a general model for using a constructivist approach in learning situations", conference proceedings for the *12th Annual Conference on Distance Teaching and Learning: Designing for Active Learning*, pp. 69-75; Madison, Wisconsin.

[Duffy et al 1991] Duffy, T., Lowyck, J., and Jonassen, D. (eds). *Designing Environments for Constructive Learning*, Springer-Verlag, Berlin.

[James et al 1998] James, Jeff, Csete, Josephine, and Kwan, K.P. A model for supporting subject-matter expert faculty in developing quality computer assisted learning software. To be published in the ED-MEDIA/ED-TELECOM 98 conference proceedings and presented at the World Conference on Educational Multimedia and Hypermedia in Freiburg, Germany, June 20-25, 1998.

[Jonassen et al 1991] Jonassen, D., Mayes, T., and McAleese, R. "Manifesto for a Constructivist Approach to Uses of Technology in Higher Education", in Duffy, T. et. al., *Designing Environments for Constructive Learning*, p231-247.

[Larken and Chabay 1992] Larkin, J., and Chabay, R. (eds). *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*, Lawrence Erlbaum Associates, New Jersey.

[Latchem et al 1993] Latchem, C., Williamson, J., and Henderson-Lancett, L. *Interactive Multimedia: Practice and Promise*, Kogan Page, London.

[Laurillard 1993] Laurillard, D. *Rethinking University Teaching: a framework for the effective use of educational technology*, Routledge, London.

[Lowyck and Elen 1991] Lowyck, J., and Elen, J. "Transitions in the Theoretical Foundation of Instructional Design", in Duffy, T. et. al., *Designing Environments for Constructive Learning*, p213-229.

[Muffoletto and Knupfer 1993] Muffoletto, R., and Knupfer, N. (eds). *Computers in Education*, Hampton Press Inc., New Jersey.

[Schwier and Misanchuk 1993] Schwier, R., and Misanchuk, E. *Interactive Multimedia Instruction*, Educational Technology Publications, New Jersey.

Acknowledgement

The author would like to thank Dr. Daniel Lee, the subject specialist for this project, for his valuable contribution.