TOOLS FOR AUTHORING CONSTRUCTIVIST COMPUTER ASSISTED LEARNING RESOURCES: A REVIEW

Barney Dalgarno

School of Information Studies, Charles Sturt University,
Wagga Wagga, Australia
E-mail: bdalgarno@csu.edu.au
WWW: http://farrer.riv.csu.edu.au/~dalgarno

ABSTRACT

The development of Computer Assisted Learning (CAL) resources has recently become a serious option for many educators. For teachers who choose to undertake all, or part, of this development themselves, the choice of authoring tool is a very important one. At the same time as interest in CAL has grown, there has also been a gradual move away from traditional 'instructivist' teaching practices towards constructivist approaches, which have a greater emphasis on the learner being active in the learning process.

The question addressed by this paper is how much assistance do the presently available authoring tools provide for the development of constructivist CAL resources?

The paper documents a review of six authoring tools: Macromedia Authorware, Macromedia Director, Asymetrix Toolbook, Claris Hypercard, Microsoft Visual J++ and Microsoft FrontPage. In choosing these packages consideration was given to the need to include tools designed for CAL resource development, for multimedia production, for interactive WWW page development and for general purpose software development.

Many software reviews consist primarily of a list of the options available through menus, toolbars and dialog boxes. This can lead to an emphasis on the number of functions provided rather than on the ease of use for authentic tasks. To avoid this problem, the review involved the development of a prototype system using each of the authoring tools.

The review found that Toolbook provided the best support for the development of constructivist CAL resources, with Authorware also faring very well. Director was found to provide excellent support for the development of multimedia resources. Visual J++ was found to provide the best support for the development of simulations and microworlds.

KEYWORDS

Computer assisted learning, authoring, constructivism.

1. INTRODUCTION

The increasing availability and capability of computers, the pressure to provide alternative forms of educational delivery, and the increase in levels of computer literacy, has seen the development of Computer Assisted Learning (CAL) resources becoming a serious option for many educators. For teachers who choose to undertake all or part of this development themselves, the choice of authoring tool is a very important one.

At the same time as interest in CAL has grown, there has also been a gradual change in the view of the teaching and learning process held by many educators. Specifically, there has been a move away from traditional 'instructivist' views towards constructivist views, which have a greater emphasis on the learner being active in the learning process. This shift has significant implications for CAL resources and consequently for CAL authoring tools.

This paper examines the level of assistance provided by presently available authoring tools for the development of constructivist CAL resources. The capabilities of six authoring tools are analysed.

2. CHOICE OF AUTHORING TOOLS TO REVIEW

In choosing tools to review, various types of tool are worthy of consideration, including tools designed for CAL authoring, for multimedia production, for WWW development and for traditional software development.

CAL authoring and multimedia production tools typically provide an interface metaphor that allows the developer to organise the media elements and define the interactions that will be possible. Vaughan (1993) describes three distinct interface metaphors: the *card* or *page* metaphor (used by Hypercard, Supercard, Toolbook, Visual Basic, Oracle Media Objects and Apple Media Tool); the *flowchart* metaphor (used by Authorware and Icon Author); and the *time-based presentation* metaphor (used by Action, Cinemation and Director). More recently *object-oriented* features have become common in many tools (including Apple Media Tool, Oracle Media Objects, Director, Visual Basic and MFactory Mtropolis). In choosing tools for this review, other reviews, such as Dancer (1995) and Seachrist (1996) and descriptions of development projects, such as Phillips (1994), Hewer and Rogan (1993), Ellis and Browne (1996), Smith et al. (1996), and Marshall (1995) were considered.

The tools chosen to review were Macromedia Authorware Professional, Macromedia Director, Asymetrix Toolbook II Instructor and Claris Hypercard.

Authorware is one of the most widely used CAL resource development tools and is available for the Windows and Macintosh environments. Version 4 was reviewed. It uses a *flowchart* metaphor, with symbols (or *icons*) in the flowchart indicating the location of information, interactions, decisions, and media clips.

Toolbook is the most widely used CAL resource development tool on the Windows platform (it is not available for the Macintosh). Version 5.0 was reviewed. It uses a *book* and *page* metaphor, where a complete project is called a book and each screen of the project is called a page. A scripting language called *OpenScript* is provided.

Director is the most widely used multimedia production tool and is available for Windows and Macintosh. Version 6.5 was reviewed. It is based around a movie production metaphor. The development environment consists of a *cast*, containing the resources used in the project, a *score*, showing the cast members present in each *frame*, and a *stage*, where the frames can be viewed and edited. A scripting language called *lingo* is provided.

Hypercard is the original CAL resource development tool and has very widely used for small projects by non programmers. It is available for Macintosh only. Version 2.3 was reviewed. It uses a *stack* and *card* metaphor, where a project consists of a stack containing multiple screens called cards. A scripting language called *HyperTalk* is provided.

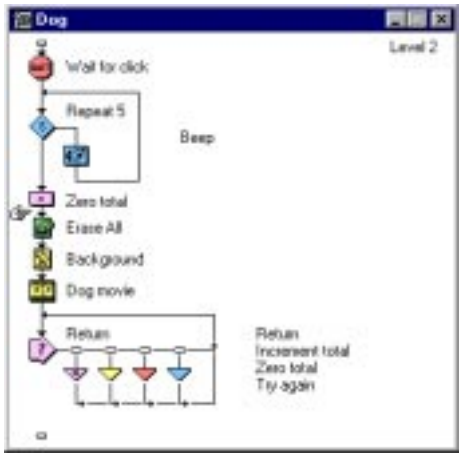

Figure 1: A section of a Macromedia Authorware flowchart

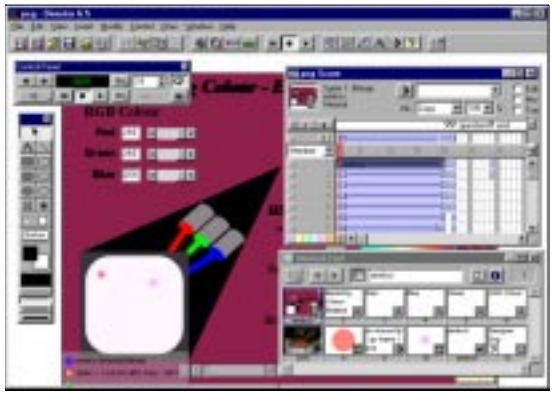

Figure 2: Macromedia Director, showing the cast, the score, the drawing tools and the control panel

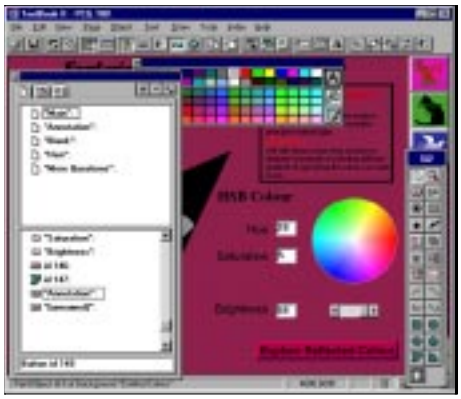

Figure 3: Asymetrix Toolbook, showing the drawing tools

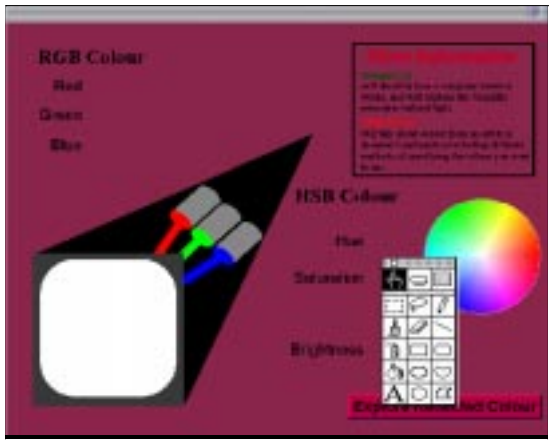

Figure 4: Claris Hypercard, showing the tools palette and the object browser

Much recent CAL resource development has focussed on WWW delivery, and consequently the inclusion of tools designed specifically for this purpose is appropriate. Such tools include those that facilitate the creation of Hypertext Markup Language (HTML) pages (such as Adobe Pagemill and Claris Homepage) and those that also support the inclusion of Java Script, Java Applets or plug-ins for enhanced interactivity (such as Microsoft FrontPage and Macromedia Dreamweaver). FrontPage has been chosen for this review.

FrontPage is available for Windows and Macintosh. Version 98 was reviewed. The review focuses on the capabilities of FrontPage for producing HTML enhanced with the use of Java Script. Proprietary features that work only on Microsoft browsers, such as ActiveX components, and features that rely on specific server software are excluded.

Software development tools are designed to facilitate the development of software using a specific programming language. As well as tools for editing and managing source code files and for testing and debugging code, they usually include graphical tools for creating forms, dialog boxes and menus. Well known examples include Microsoft Visual Basic, Borland Delphi, Microsoft Visual C++, Borland C++ and Symantec C++ (Grehan, 1996; Linthicum, 1995).

More recently, environments that allow development in the Java language, designed for Internet based software, have emerged, including Sun Java Workshop, Microsoft Visual J++, Borland JBuilder and Symantec Visual Café (Dragan, 1997).

Consistent with the recent emphasis on WWW delivery of CAL resources, Visual J++ has been chosen for this review. Visual J++ is available for Windows only, but the resources created (*Java Applets* and *programs)* can be used on any platform. Version 1.1 was reviewed.

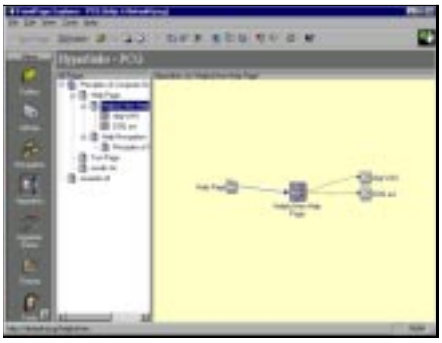

Figure 5: Microsoft FrontPage, showing the web explorer

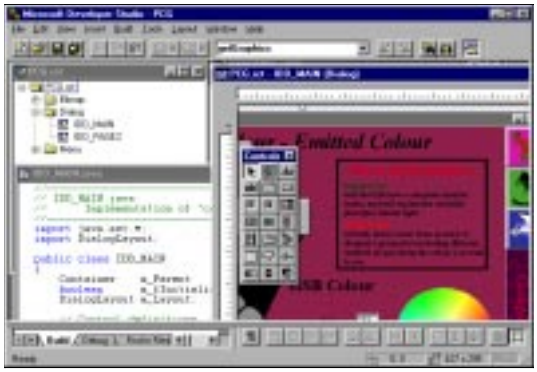

Figure 6: Microsoft Visual J++, showing the dialog editor and the resource browser

3. METHOD OF REVIEW

Many software reviews consist essentially of a comparison of the options provided by the packages reviewed. This review, however, involved an analysis of the capabilities of the each tool during the development of a prototype CAL resource. The prototype resource, titled *Principles of Computer Graphics*, was initially designed as an example of a constructivist CAL resource, and consequently includes many learner-computer interactions consistent with a constructivist view of learning. (For a detailed discussion of the characteristics of constructivist CAL resources, see Dalgarno, 1996).

During the development of the prototype resource using each of the tools, tables were compiled comparing the capabilities of the tools for developing resources, managing project elements and cross-platform delivery. This paper focuses on the capabilities of the tools for the development of constructivist CAL resources, looking specifically at the following:

- Use of media elements, including importing of files, and positioning and sizing of elements;
- Use of interactive elements, including the ease of providing interaction without programming;
- Use of questions with feedback; and
- Use of simulations and microworlds.

For each of these areas, a table comparing the capabilities of the six tools has been compiled. The criteria used in each table and the results obtained are discussed in the following sections. The tables are presented in a summarised form. More detailed versions of the tables are available from the author's home page.

4. RESULTS OF THE REVIEW

4.1 INCORPORATION OF MEDIA ELEMENTS

Broadly, this section focuses on the process of creating or importing media elements and placing them onto pages or screens. Considerations include the variety of media formats accepted and the tools available for editing and exact positioning and sizing of the media. When working with multiple media elements, the ability to align, evenly space and group the elements is important. Typically a number of screens will use the same media elements so the ease of duplicating elements or including elements on a range of pages is important.

There are also a number of additional considerations in working with specific types of media. For example, in working with text, the ease of applying standard character and paragraph formatting, and the ability to import formatted text are all important. In working with images, tools are required for creating, importing and manipulating images in bitmapped and vector format.

Table 1 summarises the capabilities of the tools for the incorporation of media elements.

Table 1

Incorporation of Media Elements

	Authorware	**Director**	**Toolbook**	**Hypercard**	**FrontPage**	**Visual J++**
Positioning and Sizing	good-excellent	good	excellent	fair	fair	good
Duplication of objects, pages	good	excellent	excellent	excellent	good	good
Text	excellent	good-excellent	excellent	fair-good	good	poor
Photos and Bitmaps	fair-good	excellent	good	good	good-excellent	good
Drawings	excellent	good	excellent	poor	poor	poor
Animation	good	excellent	good	good	fair-good	fair
Sound	good	good-excellent	excellent	excellent	good	fair
Video	excellent	excellent	excellent	good-excellent	good-excellent	poor
Transition Effects	excellent	excellent	good	excellent	poor	poor
Colour and Palettes	good	excellent	excellent	good-excellent	good	good
Window creation, sizing and positioning	fair	fair	good	poor-fair	good	fair

The review suggests that the four conventional authoring tools provide the best capabilities in the use of media elements, with Toolbook and Director coming out ahead, followed by Authorware then Hypercard. Toolbook provides particularly well for the positioning and sizing of objects, the duplication of objects within and across pages, for working with text and sounds and for creating and manipulating drawings. Director provides very well for duplication across pages and for the use of photos and bitmaps, as well as providing the most animation options, including straight and curved paths and automatic ‘inbetweening’ of size and position.

Authorware’s strengths are in working with text and drawings and in the provision of transition effects. Hypercard’s use of backgrounds which can be shared by multiple pages make the duplication of objects easy. Hypercard also provides good tools for working with sounds and for creating transition effects.

FrontPage suffers from the limitations of the HTML file format, which does not allow the degree of flexibility provided by the conventional authoring tools, in positioning of objects and in text layout and formatting. Visual J++ faired particularly poorly in the provision of tools for both importing and manipulating media elements and for positioning these elements on the screen.

4.2 INCORPORATION OF INTERACTIVE ELEMENTS

Constructivist CAL materials typically provide a number of different methods of navigation between pages or screens, to cater for different learning styles and differences in prior computer experience. Interactive elements for navigation include hot keys, menus, buttons, icons, hot spots and hypertext links. Additionally, there is a need for tools to allow for control of elements on a page such as scroll bars, and sound and video controls. If the learner is to answer questions or articulate their own ideas, then elements allowing for text or numeric input may be required. More complex interactive elements will be required if graphical simulations or microworlds are included, but these are discussed explicitly later in this paper.

In analysing the capabilities of the tools for allowing the incorporation of interactive elements, there are two issues to consider. The first is the positioning of the elements, and the second is the specification of what action will occur when the learner uses an element. If the action is simply a jump to a specific page, for example, it is desirable to be able to specify the action without any scripting or programming.

In providing for the use of text or numeric fields, the ability to validate the input, for example to ensure that numeric fields actually contain numbers, is important. In providing support for the use of menus, the ability to create both pull-down menus and lists of options on the page is required. In providing for the use of buttons, it is desirable to be able to create buttons with a 3D appearance and additionally to be able to create buttons with graphical labels.

Table 2 summarises the capabilities of the tools for the incorporation of interactive elements.

The review found that Authorware and Toolbook have the best support for providing navigation options, with both providing excellent tools for creating hot spots and hypertext. Authorware's tools for using hot keys, and menus, and Toolbook's tools for creating buttons and icons and for controlling media are noteworthy. Both provide support for jumping to another page and for searching for text, with Toolbook coming out ahead in support for the creation of standard back, next and previous buttons.

Probably the next best in support for navigation is FrontPage, which through HTML allows the easy creation of hypertext links and hot spots, linked to specific pages. Additionally, a standard back button and a home button are provided automatically by the web browser. Both Director and Hypercard are able to provide most of the navigation options, but scripting is required. J++ faired the worst in this area, with programming required for hot keys, menus and buttons and particularly complex programming required for hypertext, hot spots and media controls.

In providing for text and numeric input fields, Toolbook and FrontPage fared best, both providing templates (termed 'widgets' by Toolbook and 'FrontPage Components' by FrontPage) for validating the values entered.

Table 2

Incorporation of Interactive Elements

	Authorware	Director	Toolbook	Hypercard	FrontPage	Visual J++
Text Fields	**fair**	**fair-good**	**excellent**	**fair-good**	**excellent**	**fair**
Hot Keys	**excellent**	**fair**	**good**	**fair**	**poor**	**fair**
Menus	**good-excellent**	**fair**	**good**	**fair-good**	**fair-good**	**fair**
Buttons and Icons	**excellent**	**good**	**excellent**	**fair**	**good**	**fair**
Hot Spots	**excellent**	**good**	**excellent**	**fair**	**good-excellent**	**poor**
Hypertext	**excellent**	**fair**	**excellent**	**fair**	**excellent**	**poor**
Media Controls	**good**	**good-excellent**	**excellent**	**good**	**good**	**poor**
Ease of jumping to page	**excellent**	**good**	**excellent**	**good**	**excellent**	**fair**
Ease of creating Back, Next and Previous functions	**good**	**good**	**excellent**	**excellent**	**excellent**	**poor**
Searching	**excellent**	**poor**	**excellent**	**excellent**	**good**	**poor**

4.3 QUESTIONS WITH FEEDBACK

One of the most important component of any constructivist CAL materials is the practice exercises that provide opportunities for the learner to obtain feedback about their own knowledge constructions within the domain. These may take a number of forms, the most common being multiple choice or single word answer questions. These are typically implemented using buttons and text fields respectively. Another type of question involves the learner matching or grouping words, phrases or images. This can require more complex interface techniques, such as dragable graphics, or provision for the learner to draw lines with the mouse.

Feedback to responses can be provided in a number of ways. The simplest is to jump to another screen containing the feedback. Alternatively, an invisible text field containing feedback can be made visible, or text can be written to the screen under program control. If questions are to be used for summative rather than formative reasons, the system needs to provide for scoring of the learner's attempts.

Table 3 summarises the capabilities of the tools for the use of questions with feedback.

Table 3

Questions with Feedback

	Authorware	Director	Toolbook	Hypercard	FrontPage	Visual J++
Multiple Choice	fair-good	fair-good	excellent	fair-good	fair	fair
Text answer	excellent	fair	excellent	fair-good	fair	fair
Numeric answer	good	fair	good	fair-good	fair	fair
Graphical response, such as dragging and connecting	good	fair-good	excellent	fair	fair	fair
Feedback for each response	good	fair	excellent	fair	fair	fair
Scoring	good	fair	excellent	fair	poor	fair
Control over attempts	good	fair	excellent	fair	poor	fair

Toolbook provides by far the best capabilities for the provision of questions with feedback. Through the use of templates (termed 'widgets') multiple choice, text answer, numeric answer, and graphical response questions can be created without any scripting. Toolbook also provides for scoring of responses, again without scripting.

Authorware is the only other tool to provide any support for questions with feedback, without requiring scripting. It does not, however, provide as much flexibility as Toolbook, in the input methods for multiple choice questions and in the use of graphical questions. The generation of feedback for given learner responses is also not quite as easy in Authorware as it is in Toolbook, and the degree of support for scoring and for control over the number of attempts allowed is also not quite as good.

Director, Hypercard and J++ all allow questions with feedback to be created, but in each case extensive scripting or programming is required. FrontPage fared worst in this category. This is because graphical response questions are difficult to create using HTML and Java Script and because hiding of answers and scoring is virtually impossible.

4.4 SIMULATIONS AND MICROWORLDS

Simulations and microworlds are arguably the most important constructivist CAL techniques, as they allow for the learner to manipulate and experiment with ideas and consequently to construct their own knowledge representation. They are also the most difficult to provide from an authoring perspective, and consequently are often not used. Simulations and microworlds can be graphical, allowing the learner to directly manipulate visual representations of ideas and objects with a mouse, with events occurring within the simulated world depending on the actions of the learner. Alternatively they can be text based, with the system displaying information about the 'world', asking the learner to specify actions to be carried out, and then calculating and displaying updated information.

Simulations (or microworlds) typically involve the internal representation of a 'world' using complex data structures, with changes to this world computed using calculations on this data. In the case of graphical simulations (or microworlds), the user may be able to select and drag objects around the screen and possibly also draw lines with the mouse. In order to update the visual appearance of the simulated 'world', it must be possible to move objects around and draw new objects under program control.

Table 4 summarises the capabilities of the tools for creating simulations or microworlds.

Table 4

Simulations and Microworlds

	Authorware	Director	Toolbook	Hypercard	FrontPage	Visual J++
Selecting objects	fair	fair	fair	fair	fair	fair
Dragging objects	excellent	fair-good	excellent	fair	fair	poor-fair
Drawing	poor	poor	good	poor	poor	fair
Storing a “world”	fair	good	good	fair	poor	good-excellent
Updating “world” display	fair	fair	good	fair	poor	good
Generating “world” changes and Processing data	good	good	good	fair-good	fair	good-excellent
Saving and loading “world”	good	fair	good	poor	poor	good-excellent

As discussed, graphical simulations typically require complex user-input, such as the dragging of objects or drawing with the mouse. Toolbook and Authorware provide excellent support for learner dragging of objects, including the specification of responses depending on what object is dragged where. The other tools require complex scripting for this. Toolbook provides good support for the drawing of lines connecting objects, but free form drawing monitored by the system is much more complex. There is no support for learner drawing in the other tools, with J++ allowing it only with complex programming.

The most important issue in developing a simulation (or microworld) is the storing and manipulation of the data structures that form the ‘world’ and the visual representation of this ‘world’. This invariably requires significant programming, including the handling of complex data structures. Consequently, the tools that provide the richest programming language are likely to be the most appropriate. J++ provides the best support for storing complex data structures, for carrying out complex arithmetic operations and for producing a graphical representation of a ‘world’. The conventional authoring tools are a fair way behind, with Toolbook a marginal leader, ahead of Director.

5. OVERALL PERFORMANCE AND SUITABILITY TO TASK

Having looked in detail at the capabilities of each of the tools, it is worth looking at how each of the tools fare for the authoring of specific types of CAL resources. Four types of resources have been identified. At the most basic level are resources that contain only text and still graphics, which allow learner controlled navigation, but which have no other interactivity. Next are those that additionally include other forms of media such as sounds, videos and animations. The third type are resources that include questions with feedback and the fourth are resources that include simulations or microworlds.

Table 5 looks at the suitability of the authoring tools for producing each of these types of resources.

Table 5

Suitability to Task

	Authorware	Director	Toolbook	Hypercard	FrontPage	Visual J++
CAL resources, with text and still graphics, with limited interactivity	good-excellent	good-excellent	excellent	good	good	fair
Multimedia CAL resources with limited interactivity	good-excellent	excellent	excellent	good-excellent	good	poor
CAL resources including questions with feedback	good-excellent	good	excellent	fair-good	fair	fair
CAL resources including simulations or microworlds	fair-good	fair-good	good	fair-good	fair	good-excellent

Toolbook provides the best support for producing basic resources containing text and still graphics with basic navigation options. Director and Authorware are the next best, with Director providing good support for the presentation of information and Authorware for the provision of navigation. Hypercard and FrontPage are both quite adequate for producing these types of resources, with Hypercard providing more support for the provision of information and FrontPage providing the better navigation options. J++ is the most difficult of the tools to use for the production of this type of resources, both because its capabilities for presenting text and graphical information are limited and because the provision of navigation requires programming.

Toolbook and Director provide the best support for producing resources that additionally contain multimedia elements. Toolbook provides particularly good support for the use of video and for the provision of media controls. Director provides particularly good support for animations, videos and transition effects. Authorware and Hypercard also provide good support for producing CAL resources containing multiple media, with Authorware providing particularly good support for the use of video and transition effects and Hypercard providing particularly good support for the use of sounds and transition effects.

Toolbook provides the best support for producing resources containing questions with feedback, with the templates (termed ‘widgets’), allowing questions of various sorts to be created simply by entering information in dialog boxes and placing text and graphics on the screen, with feedback automatically provided without any scripting. Authorware provides the next best support, with support provided for the provision of a range of question types, but with some logic required to provide feedback to responses.

Resources containing simulations or microworlds are by far the most complex to produce. The storing of the data that describes the worlds and the displaying of this information in either graphical or text form requires complex scripting or programming and consequently J++, the tool that provides the richest programming language, provides the best support. Because of the limitations of J++ in the positioning of static information, however, it is likely that a tool like Toolbook, which also has a powerful scripting language but which also provides good support for the provision of static information would be equally useful. Director, Hypercard and FrontPage all provide a rich scripting language and Authorware allows the complex manipulation of data through variables and flowchart logic, but all are limited in their provision of support for graphical operations under program control.

6. REFERENCES

Dalgarno, B. (1996). Constructivist Computer Assisted Learning: Theory and Technique. In A. Christie, P. James & B. Vaughan (eds), *Making New Connections, Proceedings of the Thirteenth Annual Conference of the Australian Society for Computers in Learning in Tertiary Education*. Adelaide: University of South Australia.

Dancer, H., (1995). Authoring Tools. *Australian Personal Computer*, 16(10).

Dragan, R.V. (1997). Java Tools Get Real. *PC Magazine*, 16(1) [online]. URL http://www.zdnet.com/pcmag/features/javatool/_open.htm

Ellis, A. & Browne, C. (1996). Staff-Student Cooperation in the Development of CBT Multimedia Materials. In A. Christie, P. James & B. Vaughan (eds), *Making New Connections, Proceedings of the Thirteenth Annual Conference of the Australian Society for Computers in Learning in Tertiary Education*, (pp. 181-193). Adelaide: University of South Australia.

Grehan, R. (1996). We rounded up DOS/Windows-based C++ compilers and found Microsoft and Watcom are tops, but where's Borland's upgrade? *BYTE*, March, 1996 [online]. URL http://www.byte.com/art/9603/sec11/art10.htm

Hewer, P. & Rogan, S., (1993). Multimedia and Computer Assisted Learning at the University of Wollongong. *Sharing the Vision, Proceedings of the 11th Annual Australian Computers in Education Conference*, (pp. 187-192). Sydney: NSW Computer Education Group.

Linthicum, D.S. (1995). Rapid application development promises applications without programming. Does it deliver? *BYTE*, August, 1995 [online]. URL http://www.byte.com/art/9508/sec7/art3.htm

Marshall, D.R. (1995). Art History and Multimedia: an ArteFactual Approach. In J.M. Pearce & A. Ellis (eds), *Learning with Technology, Proceedings of the Twelfth Annual Conference of the Australian Society for Computers in Learning in Tertiary Education*, (pp.368-375). Melbourne: University of Melbourne.

Phillips, R., (1994). Producing Interactive Multimedia Computer-Based Learning Projects. *Computer Graphics*, 28(1).

Seachrist, D. (1996). BYTE Software Lab Report / How Multimedia Multitools Compare. *BYTE*, Nov, 1996 [online]. URL http://www.byte.com/art/9611/sec8/art1.htm

Smith, P.K., Lines, D., Forsyth, K.D., Fardon, M., Stoll, P. & Martin, A. (1996). Childhood Seizures CD-ROM. In A. Christie, P. James & B. Vaughan (eds), *Making New Connections, Proceedings of the Thirteenth Annual Conference of the Australian Society for Computers in Learning in Tertiary Education*, (pp. 487-490). Adelaide: University of South Australia.

Vaughan, T. (1993). *Multimedia, Making it Work*. Berkeley: Osborne McGraw Hill.