Australian Journal of Educational Technology

2002, 18(1), 89-109

Constructivist approaches to authoring

John G Hedberg and Barry M Harper
University of Wollongong

This paper discusses the thinking behind *MediaPlant*, an authoring tool which has been designed to embody several constructivist ideas in its development. It begins with some comparisons with some commonly employed tools and suggests that how the tools are designed to be used poses limitations upon the learning tasks which are designed. The chapter concludes with some examples of how the tools have been used and the types of products that have resulted from its use.

Development of educational software has had a long history of use of authoring environments that have enabled instructional designers, rather than programmers, to design and develop applications. The advantage of these tools has been that the designer did not need to be highly skilled in high level languages, but could use a simpler construction set of pre-programmed modules, often supported by a simple scripting language. The disadvantage was that the developer was limited to the pre-programmed modules available and to the underlying assumptions of the structured instructional design models adopted by the tool. The designer also had to work within the visual and procedural structures employed by the tool designers to represent the design process within which learning activities could be constructed.

However, the move toward pre-programmed modules, based on specific pedagogies, has not been universally viewed as a disadvantage. Designers have also taken the view that users of such tools may not have instructional design skills, and the more constraining the tools is toward particular instructional strategies, the more likely that users with limited understanding of learning will build pedagogically sound experiences for learners.

Multiple conceptualisations of authoring tools

Additionally, authoring tools have been modified or designed specifically for learner use. Researchers such as Seymour Papert (1993 p.142, 1980)

have long called for more open environments based on a theoretical view of learning he termed constructionism. Constructionism is based on two different senses of 'construction'. It is grounded in the idea that people learn by actively constructing new knowledge as well as asserting that people learn with particular effectiveness when they are engaged in 'constructing' personally meaningful artifacts in the context of resource rich 'virtual worlds' containing embedded tasks, designed for, or as a result of, open ended construction by learners.

Interactive learning environments, if well designed, can support learner construction of knowledge through problem solving experiences or through more creative expression. Such frameworks are based upon arguments that learners should be placed in authentic environments that incorporate sophisticated representations of context through such constructs as "virtual worlds". The assumption is that within these environments, the learner is supported by visual metaphors which are specially constructed to represent the information structure to which they have access and how the "world" operates. Within these learning environments, students are often given a rich set of resources to construct artifacts, which represent their solutions to problems and tasks that they undertake within the world. Designers construct these environments assuming the resource base includes the data needed to resolve the problem posed and the operation of the "world" supports appropriate propositions and argumentation for each solution. However the transfer of skills and learning from the 'virtual world' to the real world is not always well supported, especially when the learning environment is limited and constrained to reduce the difficulties of representation or functionality. This latter approach is true of environments which are based on the building of discrete elements and linking them together, rather than creating a more holistic task which has the core elements of the task to which the learner must transfer.

To simplify terminology amongst authoring environments, in this paper we will refer to the authoring application as a "tool" and the resources, which are collected together for display as a "project". Several authoring tools can construct standalone "projects" (for example, *Director* creates a "projector") which can be distributed to learners individually. Some tools also focus upon record keeping aspects of the system and create student files that are the student's responses to the embedded questions. Some tools work directly with the visual display, such as *Multimedia Builder* and *Hyperstudio*, all changes and design decisions are represented as visual changes to the display. Other tools provide a more comprehensive view of the multimedia objects, for example, *iShell* shows the details of each object

in terms of attributes and actions employed, as well as a runtime display window.

Visual metaphors in practice

When comparing key features of authoring tools, various authors have proposed classification mechanisms. In particular, Murray (1999) in his analysis of authoring tools at that time argued strongly that tools fell into two broad categories; pedagogy oriented and performance oriented. The problem with this categorisation is that it ignores the conceptual structuring and understanding of the tool that the user must adopt and feel comfortable with, before even beginning to think about how a particular learning design can be implemented within the structure of the authoring tool. A more useful construct for determining application of a tool would be to consider the characteristics of the tools that determine how they support users when they develop effective learning environments. Table 1 outlines a comparison of four quite different tools in what we contend is a more useful comparison, considering the key characteristics of metaphor, media support, built in tools, interactivity and scripting,

No matter what authoring tool is used, software designers have to work within the prevailing metaphor of the tool. For example, if a designer employs *Director*, the prevailing metaphor is a theatrical pageant and

Table 1: A comparison of four tools

	iShell	***Multimedia Builder* (MMB)**	***Director***	***HyperStudio***
Platform	PC and Mac	PC only	PC and Mac	PC and Mac
Metaphor	Object oriented	Page based, object oriented.	Score based with visual display and objects with attached code	Screen based, card metaphor and objects embedded.
Structure	An iShell project contains one or more documents (.k files) which include layout and object information.	A MMB project consists of pages into which objects are placed. Supports Master pages both under and over current page.	A director project contains all the elements to which it refers except for QuickTime movies	All elements are included in the one file except for QuickTime movies

Media support	Media supported include rich text files, and any graphics, audio or movie file format supported by QuickTime. Also includes html 2.0 (without forms), animated gifs, streaming movies, movies with chapter tracks, flipbook images, Quick-Time VR. (Most media elements are linked as external files.)	Media objects include text, graphic (bmp, jpg, gif, pcx, png, tif), video (avi, mpeg, mov, videocd) and animated gifs. Simple	Most formats are supported including sprites which can have their own sets of behaviors	Media objects are simple, with attributes that can be changed through dialog boxes. Most media formats supported.
Built in tools	Simple text field and box elements. Support for tabular formatting from a .tdl file. Can add a scroll bars as separate element.	Includes simple text fields, paragraph text object (type into or copy and paste), text buttons, image matrix object, static and dynamic effects, simple shapes and lines, and hotspots.	This product is quite sophisticated in terms of the range of tools embedded and the other tools which can be used to edited elements of the cast if so desired.	Includes a series of basic elements, which can be viewed in the context of each screen. The product does provide a meta-view in terms of a series of screens which can be shuffled around
Interact-ivity and Scripting	Each element has attributes depend-ing on their type (e.g. position, size, duration etc.) Events may be att-ached to elements triggered by user or run time actions (e.g. mouse down, enter key). Some events are specific to particular types of elements, others may be contained in any element.	Each elements has a set of simple properties depending on their type (label, color etc.) Actions may be enabled (e.g. go to another page, hide/show another object). A script object can be included in a page and can capture keyboard prompts. Properties of a	Complete scripting language which can be used instead of the attribute setting which can also be undertaken for simple projects	Objects can have their attributes set through dialog boxes. Limited scripting in a language based on Logo

	Commands act on elements (e.g. the drag command changes the position of an element). The project file contains overall settings (e.g. screen size etc.)	page include background image and music. Project menu allows general settings to be configured (e.g. window size, palette etc.).		
Scripting required for basic production - adding media, layout and simple navigation	Moderate. Understanding of commonly used events is needed.	Easy to use without scripting.	Drag and drop of construction and the score does enable the project to be displayed using the time dimension as the basic layout.	Basically you assign attributes to the elements within the project. Simple navigation based on the screen metaphor is easily placed on each screen (card).
Comments	Quite different to other authoring tools and terms used are unconventional. This means that the functionality is not intuitive even for those familiar with other authoring tools. The interface is more complicated than other tools with multiple palettes. Seems easy for novices to start using the tutorials and then difficult for most to go further on their own. Readily extended to larger more complex projects. Web site includes showcase of sample products.	Interface is simple and uses familiar functions (accessed by clicking on icons with pop up descriptions). Pop up windows for each object allow user to enter and alter properties and enable simple actions. A scripting language allows interactivity to be extended beyond these simple functions.	This product rapidly becomes quite complex if sophisticated behaviors are to be included. The number and variety of windows which are used to create a project requires two monitors for effective authoring	The product is extensively used in the K-12 arena. The simple interface allows young children to develop presentations including a wide variety of media. The product is not used to construct commercial product because of the speed of implementation and the lack of a runtime tool.

is conceived as a time dependent display. Events occur on a stage and are managed by a score, which dictates the depth and movement of each object over time. By contrast, *HyperStudio* employs a card metaphor where links between discrete representations of objects are embedded in a Hypertext relationship. *Authorware* (which is not now in common use) created a complex flow chart (algorithmic) structure, which was used to design the experience, but was not displayed to the learner. Most implementations required learners to work through the pre-determined paths very reminiscent of the traditional concepts embodied in programmed instruction.

From these brief examples it is clear that the visual metaphor will create a vehicle in which the instructional designer must work and conceive their project. It follows that there will be a symbiotic relationship between the instructional strategies and the way the tool enables the designer to think about the task. Of the many common tools, the older have taken a more structured approach borne out of behavioral learning theory. The more recent tools have striven to reduce the need for time or procedural structure to create an environment of intelligent objects. To represent this process the tool authors have created different ways of changing and visualising the relationship between creation of a learning environment and using these environments. Thus options that will enable time, hierarchical and spatial display are all possibly needed by designers as the learning tasks and project demand.

While most of these tools have become more complex over the past few years, with a corresponding increasing in the strength of the project which can be undertaken with them, there has also been a trend to the production of small single purpose tools that can be useful for small tasks. For example, it is possible to organise resources using the Mac only tool called *iView Multimedia* which the authors claim can:

- Instantly find on any of your disks and CDs that special photo, movie, sound, clip art, image, font, that you need for a project.
- Organise your media files into catalogs containing previews and information that can be viewed even when the original files are no longer on a mounted drive.
- Present your media as a continuous audio/visual slide show.
- Re-use your media with your preferred application, or use iView's own set of tools. You can easily print reports and export your media as QuickTime™ movies, HTML galleries and more.
- Use your catalogs as media palettes side-by-side with your favorite application. iView Multimedia supports Drag & Drop integration with the finder and any drag & drop savvy application.

- Examine and edit your media annotations, including caption, keywords, categories, digital camera photographic information and much more.
- Import media straight from the Web, and connect back to the URL with your favorite browser. (iView User Manual, 1999, p3)

Developing a new authoring environment

The designers of an authoring environment must make assumptions about the instructional design models that the tool will support and the potential end users. Some tools are designed for developers and others for learners to construct their own ideas. Several key writers have called for a reassessment of instructional design models used for the development of technology supported learning environments that assume constructivist views of learning. Hannafin and Land (1997) have suggested that we should be aiming for open ended learning environments, Jonassen and Tessmer (1996-1997) argued that we should be aiming at new learning outcomes, and Duffy and Cunningham (1996) have described a range of metaphors which structure how we teach. Additionally, Savery & Duffy (1996) have elucidated several principles that characterise this philosophical view in technology based learning environments.

Supporters of constructivist learning theories (Jonassen and Reeves, 1996) have criticised authoring tools that have some pedagogical support or constrain the designer for a particular pedagogy, such as intelligent tutoring systems. They argue that such systems are based on instructivist models of learning. However, Murray (1996) has proposed that such systems do acknowledge concepts such as intrinsic motivation, context realism and social learning contexts, but the authors who argue for pedagogical support in tools see them as 'not important, or as being too complex or incompletely understood to incorporate into instructional systems'. However, not only is it important, but examples of these environments have been developed based on the constructivist design principles and have been proven to be effective learning environments, especially when the problems that are posed are ill-structured and require more than simple factual responses. Moreover, they require the collection of evidence and a case being made for the proposed solution (see for example, Jonassen 1997; Hedberg et al, 1998; Herrington et al, 1999).

Jonassen and Tessmer (1996-1997) have also questioned the commonly used taxonomies of learning that are the basis of our instructional design models, proposing that engagement with a greater range of learning outcomes is essential for meaningful learning. They have suggested a new

framework for specifying the types of learning outcomes that modern learning environments should be developing.

Given the changes to hardware technology options, we wanted to provide design frameworks and visual metaphors that could be employed by highly skilled designers to use the tool for project creation. We also wanted a tool that could support young learners, with access to rich resources, to construct representations of their own ideas using such a tool. In response to these design suggestions and the changes that have occurred in hardware technology options, we have proposed a design model (Figure 1) that is cognisant of constructivist approaches to instructional processes, and addresses many of the above suggestions for reassessment of instructional design models (Hedberg et al, 1994).

Phase one of the model takes the basic information derived from an assessment of needs, and describes the parameters of the Project space. This is the information which is to be included in the materials, how it is structured, what the target audience understands about the information and how it might be structured for the audience. A possible structuring device might be a concept map of the ideas and links that are to be included in the project. Whatever the device used, the key idea at this stage is to begin a holistic structuring of the information and how it is going to be manipulated, that will eventually form the basis of an organising visual metaphor.

The second phase reviews the basic description and seeks to combine the structure and access to all the component elements through an appropriate instructional or interaction strategy. It also seeks to identify metaphors which help both the design team and the final presentation of the information structure. The outcome of the second phase would be a formal description such as a design brief. The detail would enable the reader to understand the underlying knowledge structures and the ways it is proposed to link them conceptually and intuitively. The key to this process is the reversal of access to the information. By this we mean in traditional designs we analyse the elements and sequence them into a presentation sequence. In this model we are trying to provide access to the data in a legitimate form, in the same way as the individual user would have access to and manipulation of the resources in the real world (see the arguments by Lave & Wenger, 1991). Thus the visual metaphor and structure must provide for information rich presentations, but ones that are extensible and

possibly able to be unfolded as the learner needs less support and scaffolding in undertaking the tasks.

The third phase can be considered a third pass at the same material, this time with the express goal of linking the design ideas into a potential presentation structure. One output of this phase would be an interactive mock up of the resources using an authoring tool to illustrate not only static display of information but also the graphical and visual metaphors used to create understandable links. The information included in this prototype may include motion and static graphics, sound and data landscapes, as appropriate to the concept under development.

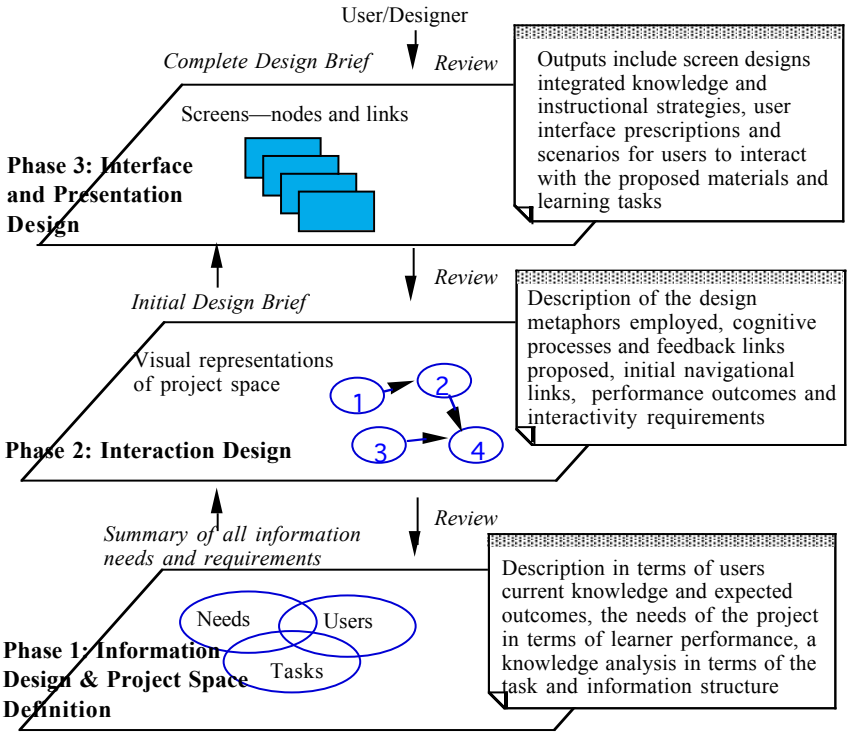

Figure 1: The design process used as the basis for interactive multimedia project development (Hedberg et al, 1994).

Implications for a software environment

The key implications for an authoring environment to support the design model would be that the environment includes:

- facilities for rapid prototyping of design ideas and restructuring of ideas simply and efficiently through a range of flexible interfaces.
- design elements that can be added through either menu selection, drag and drop, or copy and paste facilities.
- design elements that can be edited, re-used, and re-purposed simply and efficiently through a WYSIWYG interface.
- a visual representation of the presentation of the learning strategy options available at all times.
- global highlighting of an object and all of its occurrences in other views of the project (in all design and runtime views).
- extensibility so that new features can be added when necessary through extensions to the authoring tool and bridges to high level languages.
- networkability so that alternate storage options for graphic, video and audio resources can be used (for example, data files can be made available on CD-ROMs, hard disks, fileservers, or distributed across all three. Media elements such as pictures and text can be either stored internally or as external files, and in the future, it should be possible to use media and files stored on the Internet.)

In addition to these attributes for an authoring tool, the choice of an appropriate visual representation or metaphor will be critical for supporting a range of authoring tool users. This visual metaphor will either support the development of easily learned skills to enable users to express their ideas in multimedia forms, or, if unsuccessful will require users to 'learn' how the tool can be used to create projects. This second requirement is typical of most tools currently available. Key to any authoring environment is the way it sets up interactions and supports their simple execution by the tool user.

As we considered the issues we have raised for developing an authoring tool and the attributes that flow from these issues, in developing *MediaPlant*, we have struggled with questions such as:

- What visual support structures are required to create a learning environment? Is realism enough to make it an authentic context or is there some other way of interacting with the objects so that the task

of manipulation is simple? When would a 3D view be required, how should a navigable space be represented and displayed?
- Under a constructivist framework how should the user/learner create the interaction, rather than the learner having the interaction devised for them?
- What useful 'views' are there for project construction? Does it differ for learners, professional designers, or programmers?
- What general support functions should be available in an authoring tool? What cognitive support structures are necessary for learners? (For instance, Ferry et al, 1997 have demonstrated the importance of using cognitive mapping tools to generate understanding of complex knowledge domains.)
- What organisation and support structures (annotation devices, media specific views) are needed to assist in the development of projects that have potentially thousands of items to track and integrate? Should a level of customisation be available to the project developer so they can 'tailor' the visual organisation of their work to a world more in tune with the content and interactions they are developing?

MediaPlant as an authoring environment

With the framework of our design model and the questions we have raised, the Interactive Multimedia Learning Laboratory team set out to develop a software development environment. It was envisaged that this development environment, *MediaPlant*, would facilitate the production of complex cross platform learning environments for commonly available configurations, i.e. this would be an authoring tool that offered complete flexibility in the design process and also high level performance on entry level machines.

The authoring environment consists of a development tool and a runtime or 'player' program. The development program is used to construct and test the project, which is then distributed with the runtime program. The development and runtime applications enable project construction on both the Macintosh and Wintel compatible platforms, and the project files can be shared across both operating systems. The software environment is based on a C++ application framework tailored for large scale multimedia development.

The visual representation of the tool consists of a design window which functions as a 'drag and drop' construction space with additional views

that include a media window, a tools window and attribute windows for individual screen objects. A meta-view of the initial development program is based upon a tree file structure. This is being extended to present visual representations of the design by flowchart and Hypertext visual organisation.

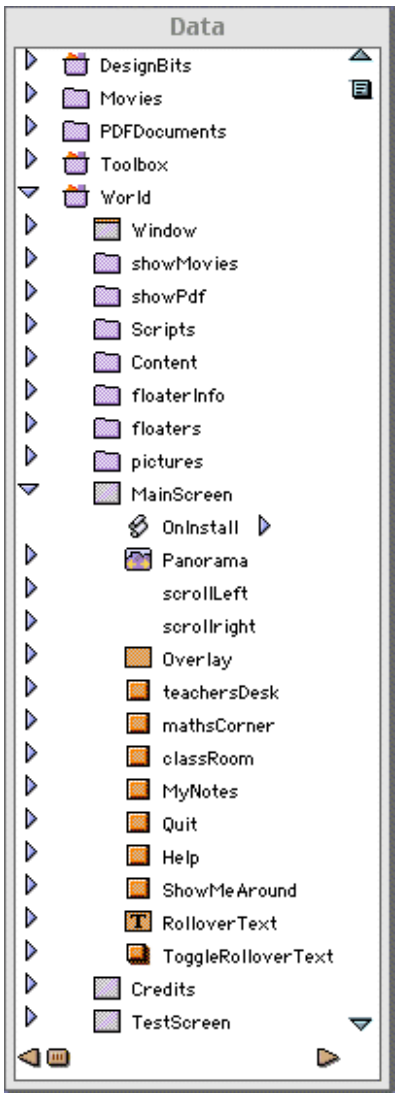

Figure 2: Tree file visual structure used in *MediaPlant* for the authoring process in *123 Counting with me*

Figure 3: The classroom metaphor and the Design window which is the result of the meta-structure of Figure 2

The file structure metaphor (Figure 2) provides a similar interface to the Macintosh Finder or the Wintel directory hierarchy, allowing easy access to all of the media, screens, screen objects and scripts within a project. By using aliases, files may be accessed wherever they may reside over a network and they may be moved without the link being broken. Projects are constructed using drag and drop, as well as copy and paste, which makes adding, editing and reorganisation of elements within a project simple. A designer has considerable freedom in how media are organised, including a choice between using external files, or storing media within *MediaPlant's* containers. Runtime screens can be displayed whenever desired, allowing easy graphical editing of items on each screen, as well as execution and testing of the product.

The object oriented nature of the design metaphor offers considerable extensibility for the authoring environment. Cognitive tools for learners, such as note pads, help windows, simulations, etc, once developed, can be simply dragged between projects, allowing designers to re-use

programmable objects that support learners in their exploration of information and construction of knowledge.

Example: *123 Count with me*

In exploring the application of the authoring tool, *MediaPlant*, to a specific project, an interactive multimedia development team used the tool to develop a CD-ROM project, *123 Count with me.* The design team had experience in developing multimedia products using simpler authoring tools and adopted the design process outlined in Figure 1.

The resulting package, *123 Counting with me,* provides a dense information landscape of resources based on general issues in professional development for elementary school teachers implementing a new curriculum in mathematics.

The information landscape uses spatial metaphors: a Classroom (Figure 3) and a navigable panorama. On entry to the environment, users are led to the main points of the package, these are presented as a programmed set of links and the palette window remains after the initial 15-second description. The package design has "learned" from previous products (Wright, et al, 1998) in that the amount of exploration is limited to enable the user to focus on the structure and organisation of the information within the package. The original metaphor was an approach to laying out a number of linked but non-sequential topics. Until it was user tested, the power of the metaphor was not really understood. After use testing, it was decided to build further on the initial "graphical design" so that the metaphor was also a way of modelling the content of the package. The visual design itself moved from a pleasant graphic into a visual model of the professional practice that is the underlying purpose of the whole project. The users are expected to employ certain characteristics in their approaches to the teaching of early mathematics (K-2). Organisation of students in the class and displays of ideas on the furniture and walls all serve to model the techniques for how it should be undertaken. Thus the actual structure of the package, a 360-degree panorama with a series of overlay information screens, serves to situate the task and have the information spatially related to the object within the worldview. The use of the tool with customisable windows has also enabled the media to be displayed within the context of the information display. Figure 4 shows not only a video screen with the controller but also a summary text that can be linked to each explanation. The other aspect of this screen demonstrates

a list structure which allows information to be "chunked" and presented in small screen readable sequences, obviating the need for large amounts of printed materials.

The structure of the tools enables a learning structure to be simply copied and pasted. The graphics, movies, etc, which have to be uniquely displayed in each section can be activated simply by a change of attributes to the window display. For example, changing a movie to display within the same graphical artwork is simply a matter of changing the file name. Thus all media are collected in major components of the project. In the case of this example all the movies are stored in the same folder on the CD-ROM, the data is stored outside the application in a container. Provision is made to have a major project broken into several containers with data for each section in its own container. This aspect has the additional advantage of enabling multiple authors to work on each section and the final project is completed by linking each container together.

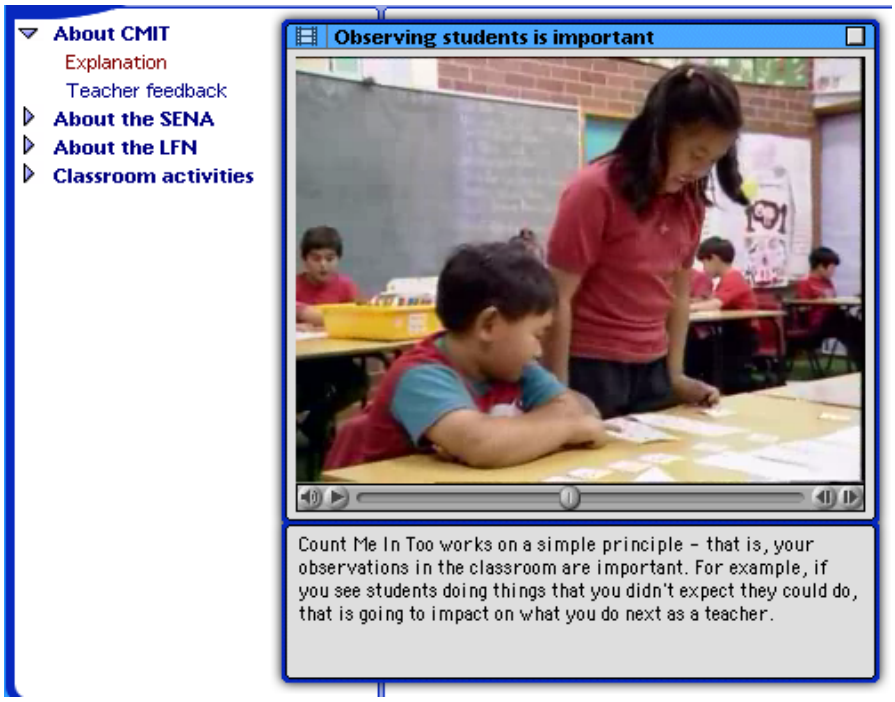

Figure 4: Video screen for 123 Count with me

Example: Creating a palette for data collection in *Exploring the Nardoo*

In many projects there is a need for the student to collect and reflect on the resources within the project. The purpose of the task is to engage students in using the skills of problem solving, measuring, collating, elaborating and communicating. In one package using *MediaPlant*, students aged from 14 to 18 years were given opportunities for to practise investigative, analytical and communicative skills. *Exploring the Nardoo* (1996) provided support for the study of interactions between living organisms and the physical and chemical environment in which they occur. The package employed an information landscape with spatial and geographic metaphors: a Water Research Center and a navigable, fictitious river environment. On entry to the environment, users are challenged to solve problems and carry out investigations on the river. The challenge encourages active learning and requires students to construct their ideas from measurements taken, resources reviewed, maps interpreted and data analysed.

The investigations of problems include issues such as fish dying from pollution, weeds infesting the river, and communities discussing farming practice. To investigate such issues, students access resources and data that are both embedded in the river environments in situ, through hot buttons, and also in the Water Research Center through organising interface metaphors such as reference books and newspaper clippings. Supportive tools include a Personal Digital Assistant (PDA) (Figure 5) designed for editing and elaborating the notes and data collected. The PDA contains a multimedia notebook for collection of any of the resources in the package, including video, audio, graphics and text; a viewer for viewing the video and graphic resources; a set of measuring tools to take measurements on the river, user support through a help file; and navigation tools.

In this second example, the authoring tool provides a re-usable and common element within the project which can be shared with other projects. In its original conception the tool enabled the display of data and visual materials such as video clips, as the user wished to collect them within the learning environment. In current work we are trying to extend the tools to enable the collection of resources in whatever form to become part of the set of resources that the learner can employ and reflect upon when they seek solutions to learning tasks. Through such examples it can be seen that interactivity and collection of ideas is encouraged and

the student will be able to use arguments which are not simply based on textual evidence. Through this type of tool within the authoring system, it should be possible to extend the forms of argument and the way evidence is provided to support each argument.

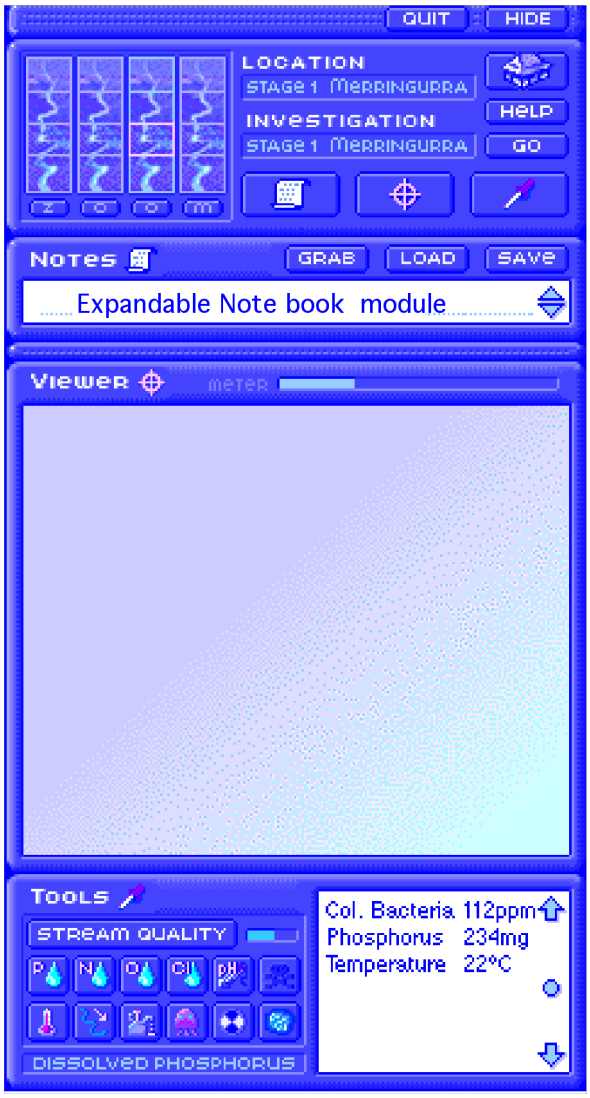

Figure 5: The Personal Digital Assistant (PDA)

Designers constructing with *MediaPlant*

When asked a series of questions about the congruence between the visual representation and how the tools work, two designers made the following comments.

1. Did the visual metaphor i.e. the visual representation of the meta-view (file structure, folders, etc) and the design view (window as it will look in the final product) constrain or support your thinking about how to use *MediaPlant*?

   > Certainly supports my thinking. The structure of containers and folders chunks the functionality of the package and thus has made it easier for me to understand and conceptualise how the functionality for 123 CWM works. For example, the container "MainsScreen" provides the "engine", of the package, which calls upon other folders (that contain the code) to run parts of the package. For example, the panorama functionality resides in the "Mainscreen" container. When a user clicks on a functional part of the panorama the code calls another folder (outside the MainScreen container) to execute. Thus, code is chunked and is easy to find (that is, if the folders have been named appropriately).

2. How well does the structure of *MediaPlant*, i.e. using the design and meta-views and storing all media except QuickTime internally help in the construction process?

   > It helps in the construction process because you can immediately see the consequences of your actions. I.e., you can add something in the file structure and immediately test it to see if it works. It's very logical in its approach. However, access to some help tools such as "how to get started", access to the library of scripts, and possible examples of sample code may assist the user.

3. What built in tools need to be provided within *MediaPlant* to assist with the construction process?

   > An index/library of all MediaPlant scripts, e.g., OnMouseUp, OnToggleDown, etc.
   > A list of short cut keys.
   > Picture icons that are more representative of what they are representing, e.g., currently a picture object is represented with an icon that has a graphic of a pencil.
   > Help on how to insert external media, e.g., movies.
   >
   > *How to insert a movie:*
   > i. Insert Movie into the Movie file (outside MediaPlant)
   > ii. Create a container - (duplicate an existing movie container)
   > iii. Change path name in the container - for SetMovie command. (Copy pathname of movie form movie folder within MediaPlant and paste)

iv. To insert Movie Icon in text pane - click on Container - Copy (apple C), then position cursor in text pane and paste (apple V)

4. How does the design of *MediaPlant* make you think about what interactivity is possible and was the scripting language adequate to implement the design of interactions?

   It seems that MediaPlant offers such flexibility that anything seems possible.

5. How would you describe the scripting required for basic production (adding media, layout and simple navigation) - is it simple enough and fast enough to construct this level of product? What needs to be added or modified?

   A comment I have is the current version is rather complex and does not lend itself easily as a prototyping tool. I also found inserting things like movies to be a very time consuming process. Perhaps if there were some high level scripts/tools like "Insert a movie" and a window appears asking the user to find the movie and enter a title of the movie, etc, that may be helpful. Also, if there were some graphic window templates i.e.: choice of floater windows – that may also be helpful.

Overall, while we have tried to create a very flexible tool, from the above comments it is obvious that the tool has still some distance to go in designing improved levels of support for instructional designers trying for their own construction. Hopefully, the first real test will be of the tool with a front end designed for school students, providing access to setting attributes, but not to the "programmable elements" mentioned in this report.

Conclusion

As new theoretical views about technology supported learning environments develop, we need to be able to explore these ideas using tools that allow us to set up these environments quickly and easily. Tools that assist the task, without complex dialogues for handling variables and options. We also need to review the visual and conceptual functioning of tools that we offer, both to designers to support their efforts in designing high quality software products, and to our users as they express their ideas using the full range of media available to them. In the light of changing approaches, it is an opportune time to review our current authoring tools, to suggest options that derive from educational issues, and to address some of the questions raised in this article.

MediaPlant is a powerful new authoring tool based on a constructivist framework that may be able to help us achieve these goals. It provides an

environment in which ideas can be extended; new options and constructs can be created, augmented and shared by other users/designers. When linked with our understanding about how users work with the more complex environment of computer based knowledge construction, it also provides supportive components. *MediaPlant* allows designers to develop pedagogical tools as objects which can then be used in other projects. We hope that we have begun to break away from the constraints of earlier tools that provided one or two ways of representing the underlying ideas of the designer. In this new environment the user can choose the way they wish to view their project. We believe that the form that makes 'sense' is in terms of the data rather than in terms of what the programmers of the authoring tool chose.

Acknowledgments

We wish to acknowledge the contribution to these ideas by the team who developed *MediaPlant*, our senior programmer Grant Farr, instructional designer Rob Wright and visual designer Karl Mutimer. A number of designers who have used the tool also for construction have provided their comments, in particular Shirley Agostinho and Ian Brown.

References

Duffy, T. M. & Cunningham, D. J. (1996). Constructivism: Implications for the design and delivery of instruction. In D. H. Jonassen (Ed), *Handbook of Research for Educational Communications and Technology*, NY: Macmillan Library Reference USA. pp. 170-198.

Exploring the Nardoo (1996). Canberra: Interactive Multimedia, Project Manager Barry Harper, Instructional Design Barry Harper, John Hedberg, Rob Wright, Grant Farr and Christine Brown (A CD-ROM based interactive multimedia package produced with the NSW Dept of Land and Water Conservation).

Ferry, B., Hedberg, J. G. & Harper, B. M. (1997). Using concept mapping to help pre-service teachers map subject matter knowledge. Paper presented to the Australian Association for Research in Education 1997 Annual Conference, Brisbane, 30 Nov-4 Dec.

Hannafin, M. J. & Land, S. M. (1997). The foundations and assumptions of technology-enhanced student-centered learning environments. *Instructional Science*, 25, 167-202.

Hedberg, J. G., Harper, B. Lockyer, L., Ferry, B., Brown, C. & Wright, R. (1998). Supporting learners to solve ill-structured problems. In R. Corderoy (Ed) *Flexibility: The Next Wave*. Proceedings of the 15th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education, 14-16 December. Wollongong, NSW: University of Wollongong. pp. 317-327. http://www.ascilite.org.au/conferences/wollongong98/asc98-pdf/hedbery.pdf

Hedberg, J. G., Harper, B. M., Brown, C., & Corderoy, R. (1994). Exploring user interfaces to improve learner outcomes. In K. Beatie, C. McNaught & S. Wills

(Eds), *Interactive Multimedia in University Education: Designing for Change in Teaching and Learning*. Amsterdam: North Holland, Elsevier, pp 15-29.

Herrington, A., Herrington, J., Sparrow, L. & Oliver, R. (1999). Investigating mathematics education using multimedia. *Journal of Technology and Teacher Education*, 7(3), 175-186.

iView Multimedia (1999). *User's Manual: Multimedia Asset Management for the MacOS*. (Version 3.6) London: iView Multimedia Limited. [verified 2 May 2002] http://www.iview-multimedia.com/

Jonassen, D. & Tessmer, M. (1996-1997). An outcomes-based taxonomy for instructional systems design, evaluation and research. *Training Research Journal*, 2, 11-46.

Jonassen, D. H. (1997). Instructional design models for well-structured and ill-structured problem-solving learning outcomes. *Educational Technology Research and Development*, 45(1), 65-94.

Jonassen, D. H., & Reeves, T. C. (1996) Learning with technology: Using computers as cognitive tools. In D. H. Jonassen (Ed), *Handbook of Research on Educational Communications and Technology*. New York Scholastic Press in collaboration with the Association for Educational Communications and Technology, Ch. 25.

Lave, J. & Wenger, E. (1991). *Situated learning: Legitimate peripheral practice*. New York: Cambridge University Press.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10, 98-129.

Savery, J. R., & Duffy, T. M. (1996). Problem based learning: An instructional model and its constructivist framework. In B. G. Wilson (Ed), *Constructivist Learning Environments: Case Studies in Instructional Design*. Englewood Cliffs, NJ: Educational Technology Publications. pp. 135-148.

Wright, R., Harper, B. & Hedberg, J. G. (1999). Visual Support for authoring. In J. van den Akker, R. Branch, K. Gustafson, N. Nieveen & T. Plomp (Eds), *Design Approaches and Tools for Education and Training*, London: Kluwer Academic Publications. Chapter 17, pp 205-214.

Wright, R., Hedberg, J. G. & Harper, B. (1998). Learner construction of knowledge: Using *StageStruck* to develop a performance. In R. Corderoy (Ed), *Flexibility: The Next Wave*. Proceedings of the 15th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education, 14-16 December. Wollongong, NSW: University of Wollongong. pp. 673-679. [verified 2 May 2002] http://www.ascilite.org.au/conferences/wollongong98/asc98-pdf/wrighthedbergharper0165.pdf

John G Hedberg, Professor of Education
Faculty of Education, University of Wollongong
Wollongong NSW 2522, Australia Email: John_Hedberg@uow.edu.au

Barry Harper, Professor of Education
Faculty of Education, University of Wollongong
Wollongong NSW 2522, Australia Email: Barry_Harper@uow.edu.au