# A Personal CAL Workbook

*S.C.Maddock*

*s.maddock@dcs.shef.ac.uk*

*Dept. Computer Science*
*University of Sheffield*
*Regent Court, 211 Portobello Street*
*Sheffield S1 4DP*
*U.K.*

Fax: +44 742 780972
Phone: (0742) 768555 x.5592

## Abstract

Computer-Aided Learning (CAL) can act as the catalyst for involving students in their courses more. This paper proposes that a rewarding, active learning experience can be achieved when students develop their own personal CAL workbook for the module they are actually taking. Such students have to (i) assimilate information and then structure it for others, (ii) devise assessments and thus learn how to assess others, and (iii) work in teams and thus foster management skills.

## 1. Introduction

A traditional lecture can be characterised by the lecturer aiming his talk at the target of a hypothetical 'average' student who listens and makes notes - as Virginia Woolf said, "The first duty of a lecturer - to hand you after an hour's discourse a nugget of pure truth to wrap up between the pages of your notebooks and keep on the mantelpiece for ever" [WOOL29]. This is passive learning, often adopted because of the sheer numbers in the lecture theatre. Even in its most developed form, employing sophisticated visual delivery methods, it is still 'couch-potato' learning. Of course, active learning can be encouraged in lectures in a number of ways, eg. pyramids, quiet times and note swapping [SHEF93]. However, none of these address the separation of lectures and project work. Computer-Aided Learning (CAL) can bring the two together and involve the students in their courses more.

CAL systems range from book analogies in which a student 'turns pages' to systems which use multiple media and offer intelligent interactive assessment. [OWEN92] reports on the development of a multimedia system called HyperGraph which is used to teach computer graphics. CAL can replicate existing teaching practices and, more importantly, can stimulate research and re-evaluation of what makes effective teaching. It could also be a solution to the problems caused by increasing the numbers of students on courses. However, one of the main hurdles to producing CAL systems is that they can take an inordinate amount of time and manpower to develop. One possible solution is to use the students as authors.

The supervised development of a CAL tool can be a rich and rewarding learning experience [MADD93] - students learn how to (i) assimilate information and then structure it for others, (ii) devise assessments and thus learn how to assess others, (iii) work in teams and thus foster management skills, and (iv) react to user feedback during product development. In fact, it is not only the students who learn from such an exercise. As [EDWA93] states, "All learning experiences should be interactive, i.e. generation of a dialogue between student and teacher". Maddock et al [MADD93] concentrated on students developing a CAL tool to cover a few lectures worth of material. This paper describes how we are aiming to build on this work by having students use a constrained authoring environment (called HyperBookShell [HOPK93]) to develop a CAL system which covers a complete course (half-) module.

## 2. Computer Graphics Modules at Sheffield

At Sheffield, each Single Honours Computer Science undergraduate student follows a three year programme (four year degrees are also possible with one year being spent in industry) taking six modules in each of the three academic years. An academic year is composed of two semesters[1] each of 15 weeks duration. In the first and second year all modules are taught modules. In the third year, students choose four taught modules and must do an individual project which counts as two modules. The course mark at the end of the second year contributes 25% towards the final degree mark, with the third year counting for 75%.

Students can opt for two Computer Graphics modules in their three years: HCI and Computer Graphics (HCI and CG I, since it is composed of two half modules) in the second year (or third year) and Computer Graphics (CG II) in the third year. HCI and CG I are prerequisites for CG II. Parts of [HILL90] and [FOLE90] are recommended

[1] The University of Sheffield will begin a semester system in September 94. For the purposes of this paper the current system of three terms of 10 weeks has been mapped onto the semester system.

for CG I, [PREE90] is used on the HCI half-module and [WATT93; WATT92] source CG II. In this paper we will concentrate on CG I. This half module consists of 30 hours contact time (lectures and tutorials) and is taught in the second semester (with the HCI half taught in the first semester). Practical work is extra. There are 75 students taking the module this academic year.

For the HCI half of the module, the students undertake a structured exercise to develop a CAL system (using Asymetrix Toolbook) for a topic from one of their other taught modules [MADD93]. They work in self chosen groups, each of which choose a different topic to develop a CAL system for. Examples of topics chosen last year were dataflow diagrams, petri nets and turing machines. The students are told to try to cover about one lecture's worth of course material per group member. The exercise is carefully structured so that the students don't flounder or have a negative learning experience. Each week the next task is set and immediate feedback is given on the previous week's task. Thus the students have goals each week and are continually working on the project. Last year's end of module survey revealed that they welcomed this structure.

Thus, on the HCI half of the module, the students become comfortable with an authoring package for CAL systems, learn how to structure the development of a CAL system and understand the importance of user feedback. This experience can obviously be fruitfully exploited on CG I. In fact, it was under this assumption that a plan for developing CG I was begun at the start of the 1992/93 academic year.

## 3. Bootstrapping Computer Graphics Education Using Student Power

The long-term aim for CG I is to replace formal lectures and tutorials with a mix of CAL and tutorials, thus promoting more student self-learning. In order to realise this, a 4 stage plan is currently being followed. An important part of this plan is that the students are involved as much as possible. In brief the plan is as follows:

*Stage 1. Building a base of enthusiasm (1992/93)*

The goal of this stage was to establish a courseware repository, populated from a range of sources including software created by students as part of their practical work. This was needed for stage 3.

*Stage 2. Development of a navigational framework (May-Sept 93)*

Here, a tool was developed in order to enable CAL systems based on a simple book analogy to be easily created using graphical interaction [HOPK93].

*Stage 3. Development of personal CAL tool (1993/94)*

Students, working as part of 4-person groups, will use the tool from stage 2 along with the courseware repository to construct team CAL workbooks for the module they are taking (ie. CG I).

*Stage 4. Rationalisation (Summer 94)*

The CAL tools of 93/94 will be turned into a single CAL book which will be given to each student on the CG I half-module in 94/95.

### *A Courseware Repository*

The learning value of seeing graphics algorithms in action and interacting with example programs cannot be over stressed. A courseware repository *is* needed for a computer graphics course. Happily, such a repository can be easily sourced. For instance, ftp sites offer a plethora of examples, [SCHW92] describes a set of interactive tools that

can be used in the classroom and students on a module can be asked to write CAL programs as part of their practical work. This last one is perhaps the most important.
Programs developed at the home site show students what is possible using the local facilities and often act as a challenge to do better. Last year, the computer graphics students at Sheffield undertook projects to develop interactive programs for different topics in Computer Graphics. A selection is shown in figure 1. Figure 1a shows an example from a program covering windows and viewports - a user is able to view the results of specifying different windows and viewports for a scene drawn in 2D world space. Figure 1b shows a tesselation design tool. Rubber-banding is used in designing a tile to tesselate the plane. Figure 1c is a CAL system which introduces L-systems and allows a user to specify a set of replacement strings and parameters and then view the resulting 'tree'. Fgure 1d shows a simple paint package incorporating a number of the techniques discussed in lectures. An important part of developing any such programs is for students to review each others work. This has the added advantage that they also increase their practical experience of other topics on the module. The best programs are chosen for the courseware repository. This introduces an element of competition which is heightened by first picking the best six and then choosing the best one after further review.
A courseware repository can thus be easily populated and can be added to continually. It then remains to consider how to make the best use of it. However, first we'll consider CAL authoring.

## *HyperBookShell* **[HOPK93]**

In a typical CAL environment, authors use a full authoring language, such as Openscript in Asymetrix Toolbook, to construct a hypertext or hypermedia CAL system. Readers then use the CAL system. In HyperBookShell, the two are combined - the reader is the author. We shall call them authoreaders. In many CAL systems a reader can annotate pages (see the CLEM system [BOYL93] for an example). In HyperBookShell an authoreader can edit the material on pages of the CAL book, can change the structure of the CAL book by moving pages around and can create new pages.
HyperBookShell is based on a book analogy. It presents authoreaders with a graphical interface with icons for constructing and manipulating chapters, sections and pages. It provides a wireframe construction kit which the authoreaders then render. Figure 2 gives an overview of the architecture of CAL books that can be constructed using HyperBookShell. A CAL book initially consists of a front cover, a contents list (of chapters), a single chapter (starting with a chapter title page (see figure 3) and a contents list of sections) with a single section and a single page in the section, a bibliography and an index. The authoreader will then add new chapters, new sections within chapters and new pages within sections. There is a limit of nine chapters.
A page created using HyperBookShell can take a number of forms. These are chosen from a list of templates when the new page icon is selected. The list includes a text page, a graphics page, a launch page, an algorithms page and an assessment page. The template list also defines mixtures of these. For instance one mixture is a text area and a graphics area. Currently the areas are of a fixed size defined by the template.
Figure 3 shows a chapter cover page. When the new chapter icon is chosen, the authoreader supplies the title of the chapter. This can be subsequently edited by selecting it on the chapter title page. Figure 4 shows an example of a fresh page. A text area and a graphics area are ready for input. The user can type new text into the text

area and can create a graphic for the graphics area by clicking on the buttons which automatically link to a paint or drawing package.

### *CAL Workbooks*

An authoreader creates a CAL workbook. If the authoreader is a team of students then they create a CAL workbook for the module they are taking. If the authoreader is a lecturer, he creates a CAL workbook for the students taking the module. However we suggest that the students should create the CAL workbook and deliver it as (part of) their assessed work for the module. The following year's students could then use the best (perhaps more than one) previous year's CAL workbook as a starting point. This could be used to dispense with formal lectures, replacing them with tutorials.

In using HyperBookShell, an authoreader is constrained to a book analogy. Also all CAL books developed using it will have the same 'look-and-feel'. The creative part is gathering, structuring and presenting the material in the book. Material can be harvested from lectures, textbooks and other sources. This must then be structured. The lecturer helps here by supplying (and discussing in a tutorial) the module map that structures his lectures. An authoreader can then choose from a range of page types in presenting the material. Graphics can be incorporated, text or program code can be transferred from wordprocessors or directly typed in and assessment pages consisting of multiple choice questions can be created.

For an assessment page the page templates list offers the authoreader three formats - one, two or three answers to a single question. This is not particularly flexible but further types of assessment page could be added to the template list. On an assessment page an authoreader types in the question and the answers. He then indicates the correct answer. There is a switch button on the page which then hides the information about which is the correct answer for the next authoreader. Including assessment pages means an authoreader is forced to think about what a reader should have understood from the previous pages of the book. A team of students acting as an authoreader would have to test their workbook on two groups of users - one which has done the module before and one which hasn't. This has the added benefit that those who have not done the module get a chance to feed input into the module they may eventually take and those who have done the module before can supply the experience they learned from doing it (see figure 5). The aim of enforcing such testing is to get students to build software that suits the users. This is a valuable learning experience.

Launch pages offer the ability to link to an executable program. An authoreader would use these to link to programs in the courseware repository. Such programs may offer another reader the chance to practice the techniques described in the previous pages of the book or may just play an animation or show an algorithm in action. As part of the assessed work for the module, the authoreader (a team of students) would be expected to develop an extra example interactive program.

As a final stage of their practical work for the module each student team will review the work of two other teams and write a report on its advantages and disadvantages. Thus the student teams will be assessed on three things: first, their CAL workbooks; second, their assessment of the CAL workbooks of two other teams; and third, the interactive program(s) they developed as part of their CAL workbook.

## 4. Conclusions

CAL systems takes a lot of time and manpower to produce. Using the students on a module as the manpower can solve this problem and at the same time promote a more active learning experience. With HyperBookShell authoreaders (individual teams of students) can create their own personal CAL workbooks for the module they are taking. They become actively involved in understanding the content and structure of the module they are taking, not only for themselves but for those that will use their CAL workbooks in subsequent years - 'Disce Doce'. In effect, we are bootstrapping computer graphics education using student power. As a result, groups of students on the module in 1994/95 will have a personal CAL tool which they will adapt to their own learning style.

As it stands, HyperBookShell has no 'Hyper' part. The current implementation does not allow hyperlinks to be made. Since an authoreader can edit the book, an extra control level needs to be incorporated to update links between parts of the book. Hopkins [HOPK93] explains how this could be done using LIM nodes which act as bridges with multiple typed links between concepts in the book. Further work which could be done on HyperBookShell includes building in facilities for student assessment of the module being taken and the lecturer giving it, incorporating unstructured thoughts and ideas for future authoreaders and allowing multiple authoreaders to work on the same work book (Computer Supported Co-operative Work).

The wordprocessor and the spreadsheet were the applications that caused the widespread use of the computer. It is not yet clear which application will do the same for multimedia. Perhaps it will be the creation of personal CAL workbooks. A student would download a CAL workbook for a particular course module, use it as the core learning material for the module and then add to it and submit it as the practical work (or even all the assessed work) for the module, or critically appraise it, constructing the appraisal as another CAL workbook.

## References

BOYL93 Boyle, T., J. Gray, B. Wendl, M. Davies, "Taking the Plunge: a Case Study in the Intensive Use of Computer Assisted Learning", CAL93, University of York, April 1993.

EDWA93 Edwards, J.D., "Benefits, Uses, Potential and Pitfalls of CBL", Educational Technology Service, University of Bristol, 1993.

FOLE90 Foley, J.D., A. van Dam, S.K. Feiner and J.F. Hughes, "Computer Graphics: Principles and Practice (2nd edition)", Addison-Wesley, 1990.

HILL90 Hill, F.S., "Computer Graphics", MacMillan, 1990.

HOPK93 Hopkins, J., "HyperBookShell - CAL framework", MSc dissertation, Dept. Computer Science, Univ. Sheffield, UK, 1993.

MADD93 Maddock, S.C., R.H. Minors, J.J. McGregor, "Disce Doce: The didactic experience of developing a CAL tool", First United Kingdom Toolbook Users Conference, (Glasgow Caledonian University, 8-9 September 1993), pp 34-35.

OWEN92 Owen, G.S., "Hypergraph - A Hypermedia System for Teaching Computer Graphics", Interactive Learning Through Visualisation, S. Cunningham & R.J. Hubbold, eds., Springer-Verlag, 1992.

PREE90 Preece,J. and L. Keller (eds), "Human-Computer Interaction", Prentice Hall, 1990.

SCHW92 Schweitzer, LtCol. D., "Designing Interactive Visualization Tools for the Graphics Classroom", 23rd SIGCSE Technical Symposium on Computer Science Education, Kansas City, MI, March, 1992 (reprinted in SIGCSE Bulletin, 24(1)).

SHEF93 "The Teaching Times", Enterprise Unit, University of Sheffield, issue no.4, July 1993.

WATT93 "3D Computer Graphics (second edition)", Addison-Wesley, 1993.

WATT92 "Advanced Animation and Rendering Techniques: theory and practice", Addison-Wesley, 1992.

WOOL29 Woolf, V., "A Room of One's Own", 1929, pg 1.

## Profile

Steve Maddock is a lecturer in the Department of Computer Science. His research interests include computer graphics modelling and animation, CAL and virtual reality.