Computer Aided Assessment through Hypermedia

Andrew Peel

University of Kent at Canterbury

ABSTRACT

The purpose of this article is to discuss how hypermedia packages can be used as a tool for the creation of Computer Aided Assessments. It describes the successful implementation of a hypermedia tutorial on **CAA** *in Higher Education using Office Workstations Limited's PC Guide.*

Introduction

With the increase in the number of students entering Higher Education Institutions without a similar rise in teaching resources, Computer Aided Assessment (**CAA**) provides not a replacement for personal teaching, but a supporting aid.

If a student does not understand material presented in the traditional lecture format, then perhaps a browse through a computer based tutorial, probably written by the lecturer, from a different perspective, would shed some light on the problem. This would reduce the amount of people who rush straight to the lecturer when there is a point they don't understand.

This is one of the aims of Computer Based Learning (**CBL**), but **CBL** needs to incorporate **CAA** to be of any real use. The student needs to receive feedback on their performance to help them judge how well they are proceeding with the material.

This can be achieved by positioning questions at what may be vital points in the tutorial, which it is only optional to answer. A further, more formal (and compulsory) examination of the main points may be useful at the end of the tutorial.

Hypermedia packages offer a simple tool for the creation of **CBL** materials which are visually attractive, highly flexible, and intuitive for students to use. These packages normally also provide a scripting language that sits behind the information frames and can be used for manipulation of the data needed in **CAA**, as well as manipulation of the hypertext document.

This article discusses the implementation of a package to demonstrate **CAA** through hypermedia, using PC Guide (produced by Office Workstations Ltd) and its scripting language, LOGiiX.

The package

Completed as part of TLTP Project ALTER, the package is called "An Introduction to Computer Aided Assessment in Higher Education", and is a hypermedia tutorial on the use of **CAA** in HE. Although the tutorial material concentrates on multiple-choice items in particular, it gives examples of other types of question, such as multiple-response, and text-entry items. It is implemented in PC Guide.

Hypermedia is suitable for **CBL** because, at its simplest, it just requires the display and navigation of teaching material. However, for student assessment we need some method of automatically marking students' answers, and of keeping track of their total score. In the package, this is done using PC

Guide's scripting language called LOGiiX. This is similar to the Pascal programming language, with added hypermedia functionality through language extensions.

PC Guide has four types of hypermedia "buttons" (where a button is usually represented as a section of text, or part or all of an image): reference, expansion, note and command buttons, although the assessment part of the package mainly uses command buttons. A command button is hyperlinked to a section of a LOGiiX script which is executed when the button is pressed.

All hypermedia buttons have a number of attributes, which include a unique ID, the ID of the object the button is hyperlinked to, and a user defined name field. Each multiple-choice option is implemented as a command button pointing to a single LOGiiX script. The command button representing the correct option in each item has *"correct"* entered in its name field.

When one of the options is chosen (by clicking on it with the mouse), it executes the following script:

```
#LOGiiX
Global score;
function main()
begin
    if (GetObjectName(GetTopDocID(), ButtonID()) = "correct" then
    begin
        score := score + 1;
        Answer(0+64, "Question 1", "Correct!")
    end
    else
        Answer(0+64, "Question 1", "Wrong answer.")
end
```

With each answer option implemented as a command button pointing to the above script, the tutorial responds appropriately to the answer option selected by the student. The answer to each item is stored as part of the hyper-structure of that item (in the command button name field).

The *GetObjectName(GetTopDocID(), ButtonID())* function simply returns the value *"correct"* if the selected button is that of the correct option, or *""* if it is not. This result is then passed to the student using a standard dialogue box (instantiated using the *Answer(0+64, text)* function).

The command button name field is visible to the question's author during editing with PC Guide, but is not visible during student use with PC Guide Viewer (a public domain Guide document browser), which keeps LOGiiX scripts hidden from the user.

The script is independent of the content of the question which means that a series of questions can be set using the same script. Each command button answer option points to this single section of LOGiiX script which handles the dialogue with the student and the scoring mechanism.

This software reuse is analogous to procedure calls in high level languages, with data being passed to the procedure through command button attributes, rather than through parameters. This results in the simplified creation and maintenance of the LOGiiX scripts used in the tutorial.

Types of question

More complex questions may also be implemented using command buttons with variations in this basic script. For example :

- The amount of time the student takes to answer each question can be measured and an average calculated upon finishing the section.
- The answer options for each multiple-choice item can be displayed on the screen individually for five seconds each, and then the student has to select from a list of letters representing the answers. This reduces the probability that the student will guess the correct answer by comparing the possible answers when listed together.

- A list of items can be shown on the screen, and the student has to put them into a priority order, by clicking on the items in that order with the mouse.
- A list of possible answers can be shown on the screen, and the student has to select *all* the options that answer the question correctly.
- Fill-in-the-gap type questions may be set by presenting the student with a sentence, and when they are ready to answer, they may click on the gap and a dialogue box will appear into which they can enter their answer. If they cannot answer, they may be given a list of possible answers as a multiple-choice item. If they then answer correctly, they only score half as many points as they would if they had known the answer immediately.
- Images can be displayed on the screen alongside a list of names to match against the images.
- An image can be displayed on the screen and the student asked to point to a place on the image that represents the correct answer to a question.

Further features

The tutorial discusses ways of discriminating between the students who know the correct answer to a multiple-choice item, and those who manage to guess correctly. It suggests that one way to achieve this would be to use ‘negative marking’. That is, to subtract a fraction of a mark from a student’s score when they choose a wrong answer.

Negative marking uses the following equation:

$$Corrected\_score = A - \frac{B}{C - 1}$$

Where A is the number of items answered correctly, B is the number of items answered incorrectly and C is the number of answer options in each item.

The tutorial allows the user to enter different values into the equation, and then shows the corrected score, allowing the user to see the effect of negative marking.

By subtracting marks for incorrect answers, we can discriminate between students who are good, and students who guess. For example, if a student answers half of the items in a multiple-choice test correctly (ie. A = 50, B = 50), and each item had four answer options (ie. C = 4), the corrected score would be 33%, rather then the normal 50%. If the student’s score is higher than 50% then the corrected score becomes closer to the actual score, and if the score is less than 50% then the corrected score becomes further away.

This demonstrates that the discrimination available through negative marking produces a wider separation between the marks of students with differing abilities.

A later section in the tutorial deals with Computer Assisted Marking. There is a demonstration in which a pro forma is created for each student, detailing the marking criteria and a marking schedule. The tutor can fill in the pro forma with the marks he/she awards to the student, which can then be saved on computer.

This stored score information can then be used to generate statistics showing how the students on the course performed, or can be given directly to the student to help improve their future work.

Problems with PC Guide

In producing this tutorial a number of problems have been encountered which have highlighted shortcomings in PC Guide.

- LOGiiX has no complex data structures, which make LOGiiX scripts more difficult to code, and cumbersome to maintain.
- LOGiiX has poor facilities for the manipulation of text strings.
- It only runs on a PC under MS Windows.

There are also some features that we may like to implement in the future, involving the manipulation of marks produced by a group of students. For example, on completion of a test the user could be told the average score within the group, or the tutors of under-achieving students could be sent email notifying them of this fact. Unfortunately, this is not possible using PC Guide.

Conclusion, and The Future

PC Guide has provided a tool for the successful implementation of a tutorial introduction to **CAA**. Hypermedia provides an extremely intuitive way of navigating around the tutorial material, and LOGiiX provides a powerful tool for the manipulation of hypermedia, although its lack of complex data structures is a hindrance.

In the Computing laboratory at the University of Kent we have now moved onto using NCSA Mosaic and the World Wide Web for another project (the High Performance Computing Consortium TLTP). This provides a multi-platform base (PC, X Terminal, Macintosh) for producing hypermedia **CBL** packages which, although less visually attractive and less flexible in the display of text than PC Guide, the assessment scripts that deal with processing student answers can be written in any programming language. This means that data manipulation can be made more complex, as well as making it possible to provide the multi-user functionality described above.

Also, the project is looking into making it simpler for less technical users to create new assessment materials, by entering information about the assessment (such as the type of question, the correct answer to the question, how many marks for each question part, etc.) into a text file. This information can then be used to automatically generate the hypertext frames of the tutorial, complete with facilities for user dialogue and score keeping.

It is these advantages that have lead us to look at producing a version of the **CAA** tutorial for the World Wide Web (**WWW**), and developing it further to incorporate the multi-user functionality in particular. This would then be accessible from any PC, X Terminal, or Macintosh connected to the **WWW**, anywhere in the world.

Further information

Andrew Peel, Computing Laboratory, The University, Canterbury, Kent CT2 7NF.
Email: A.T.Peel@ukc.ac.uk
Tel: (0227) 764000 Ext: 3979
Fax: (0227) 762811

Information on how to obtain "An Introduction to Computer Aided Assessment in Higher Education" is available from:

Joanna Bull, Project ALTER, UCoSDA, Level Six, University House, Sheffield S10 2TN.
Email: J.Bull@sheffield.ac.uk
Tel: (0742) 750820
Fax: (0742) 728705