The Web Learning Pages

Burton Rosenberg
Department of Mathematics
and Computer Science
University of Miami

6 April 1995

Abstract

The Web Learning Pages is a project to reinforce and accelerate classroom learning in the Department of Mathematics and Computer Science at the University of Miami. The doctrine followed by these pages is that Web information is of a new and unusual type: partly static, like a book, and partly real-time, like a conversation. We are trying to exploit this transitional nature of the Web for more effective and efficient teaching.

KEYWORDS: education, collaboration environments, documents and authoring tools, hypermedia, tools, usability, user interface, virtual environments.

1 Introduction

This paper describes a script language in which programmed education materials can be easily constructed. The underlying technologies are HTTP Widgets and Forms. Using the standard technology of the WWW gives the materials very high availability. Any one can access the materials at any time. The contribution of this language is to make the materials easy to construct, having been organized into standard widgets which are recalled and customized with very little effort.

The design of widgets is meant to enshrine the tradeoff between interactivity and availability. We are not interested in the very best or the most amusing animation of pages, but on the other hand we remain sensitive to the positive quality an engaging presentation can have on the learning process. We balance this against ease of construction. Writing these pages should be as easy as typing an email response. If this is achieved, there will be a sort of slow yet active dialogue between page writers and page readers.

Although the general aim of electronic teaching tools is pursued by numerous workers in the education industry, the author is unaware of any publicly available tools to aid in the pursuit of this precise goal: to author in the smallest amount of time possible supplementary materials available through the WWW to students which require or desire additional contact with course materials. It is hoped that this project will be a useful tool, that it will accumulate materials over several years and provide a medium for sharing developed materials.

In the next section we analyze how the Web pages might lead to more effective and effi-

cient education. The analysis presented in this section is purposefully abstract to prevent us from committing to too narrow a principle. An implementation is described in the following section. The final section will criticize the implementation according to how it achieves these goals.

2 Goals

Here are three remarkable properties of the Web which we would like to capture in our Web Learning Pages:

1. The inspired manner in which pages are linked into a web-like body of interreferences. What is most stimulating about these links is their capturing the interests and personalities of the people that make these links. Rarely could a set of static links be as interesting.

2. The new compromise between resources and availability. What is required is a less expensive form of conversation. New inexpensive communication and computation technologies make highly available, moderately interactive services a current possibility. We are willing to tradeoff the slowness of the conversation against the expense of full interactivity. In fact, for thoughtful endeavors, this slowness can be turned to an advantage. Meanwhile, by being less expensive, we can afford a larger scope before overwhelming available resources. The Learning Pages can be reached from anywhere at any time.

3. The Web page is a small automaton, using the original sense of that word. For clarity, we will exaggerate by using the word "toy". Toys are more amusing and enjoyable than static pages and I claim that the toy is a more efficient carrier of information than static pages. This active nature of the pages can provide them additional durability and encourage rereading beyond what is necessary for data transfer.

   We hope to create a collection of pages which function as a toy. While playing with the toy the reader learns. Literally, the toy could be a model of what is to be learned. This approach has been widely taken. However, the connection can be more abstract. The toy carries text. The method of playing with the toy engages the reader with the text. Both the physical presence of the text and its meaning are being relied upon to carry the text's message.

Our pages should be moderately captivating, but require very little attention of the page writer to maintain and improve. It can provide a meeting point for discussion, but not require the bandwidth (or dedicate attention on the part of the writer) of a real-time, shared experience. This distinguishes the goals completely from "tele-learning". Conceptual fundamentals for this project are found in the author's early work [1], in Piaget [2] and Papert [3].

3 Implementation

The molding of text into toys is facilitated by the implementation of WLP widgets. A prime directive is that the widgets be easy to use, that they should write easier than even HTML.

The page in WLP is called a *form*, because it corresponds to a HTML form. Each form is generally an HTML page, but supplemented by *directives* which are directed to the WLP software. The directives are meant to make

convenient the construction of Web Learning Pages and to automate much of the HTML authoring. Widgets communicate with each other by sending messages. The messages are caught by the WLP system and trigger the emission of more forms according to a rewriting system described below.

To introduce the type of widgets we have implemented, consider the traditional programmed learning situation: a short text is presented with several options to choose from beneath the text. The "correct" option leads the reader to the next page. The *shuffle widget* is a convenience to the page writer of this type of text. The shuffle will automatically mix the options with each page presentation, hence the writer is free of the burden of keeping track of which among the answers, A, B, C, etc., is correct and the reader must pay attention to all the alternatives even the second time through.

To illustrate the syntax more specifically: the shuffle widget is written into a form and each of the options is given a message name:

```
#Form: EXAMPLE An Example Page
The beautiful city of,
#Shuffle: tal orl mia
Tallahassee
#:
Orlando
#:
Miami
#:
is the capitol of the Sunshine State.
#List: *none all
None of the above.
#:
All of the above
#:
```

See Figure 1 for how this form appears on the Web. The symbols introduced by a start-of-line hash and terminated by a colon are *directives*. Directives introduce widgets, define macros and delineate forms. For widgets which are followed by several text blocks, two forms are allowed. Either the page writer specifies the message texts, then the matching number of text blocks will follow, separated with the null directive `#:`, or the writer simply tags the last text block, and the message texts are generated automatically.

An example of this second form is the *puzzle widget* which randomly shuffles the text blocks and presents each one after a check box. The reader checks two boxes to transpose the blocks, until the puzzle is solved, that is, until the random shuffle is undone. This widget can be used to enliven the discussion of C Language program structure.

```
The following program appears
scrambled.
#Puzzle: named_tag
   void main
#:
   (int argc, char * argv[])
#:
   {
#:
   printf ( "Hello World!\n" ) ;
#:
   }
#:named_tag
#List: *cont exit
Transpose the two checked lines.
#:
Exit from puzzle.
#:
Clicking check boxes, the
reader unscrambles it.
```

See Figure 2 for the Web output of this form. Other widgets implemented include,

- a *list widget*,
- a *text widget*,

- and a *digression widget.*

The list is like the shuffle, but the text items are not shuffled. A default choice on a list is indicated by prefixing the message with an asterisk. The text widget is very much the text HTTP widget, and is of limited use at this point, due to the restricted mechanism that WLP messages are rewritten.

The digression uses two WLP directives, *digress* and *return*, to allow the reader to optionally follow side discussion, and then pop back to where the reader left off. The digression is also used to "digress out of" a session. By digressing out to the *Navigator Page*, the reader can save by name the digression information and leave the WLP system. Later, the reader can resume the session by returning to the Navigator Page and activating the saved digression information.

The message emitted by a widget is rewritten to the name of a form using text replacement strings found in *wire files.* The *static wire file* is written and maintained by the page writer and it serves to connect the pages together. A *dynamic wire file* is maintained automatically for the reader by the WLP system and implements a mechanism to return to visited pages. At present, a stack regime of "digressed from" pages is maintained: a message is tagged as a digression, and enough information is pushed into the dynamic wire file to allow recovery of this page A return directive causes the last pushed digression to be recovered and removed from the wire file. In order to make this system work, a unique *session key* is created for each session, and the reader wears this key throughout the session as a hidden HTTP variable. To conserve bandwidth, all state associated with this key is keep in a working directory on the HTTP server.

The format of the wire file is `message:form-name`. The received message `message` is rewritten to the form name `form-name`. The messages are arbitrary dot-delimited strings. A message match can be against any or all parts of the dot-delimited string, where wild-card matches are given by an asterisk. For example, any page can invoke the rewrite of message "toc" to form "TableOfContents" with the following line in the wire file:

```
*.toc:TableOfContents
```

Once the form name has been found, the page writer's public html directory is searched for the file containing the form. These files are called *digests* and can contain any number of forms. To speed the search, we currently require in this directory the presence of a *contents* file giving the name of all forms and the digest in which it resides.

It is the intent of the project that these many configuration files, the contents file, the wire file, and the initial entry point to the series of forms, along with a title for this series of forms, be easily configured by any writer via a wlp.conf file in the writer's public html directory. This idea has not been further developed. We were working on a single series of pages and therefore had no pressing need to consider issues of maintainability. We chastise this failing here, rather than in the next section, since the failing is entirely caused by lack of effort. The next section is reserved for criticisms aimed squarely at efforts which fell short of their marks.

4 Experiments and criticisms

An implementation with scripts for learning the C programming language is now running at:

`http://www.cs.miami.edu/ burt/wlp.html`

The widget design came directly from our efforts on this specific project. The series consists of seven chapters of which four have been written. These are collected into six digests with a total of about 6,800 words (38,500 characters). The CGI is written in Perl and is about 700 lines long.

The soft-side philosophy of the form design was to avoid saying anything straight-out. We demonstrate by example an idea or construct and ask the student to place it into words, selecting from the multiple choice list, by his or herself. Students of MTH 596: Operating Systems and Networks made valuable suggestions after having used the Learn C material. In particular, the Navigator Page was a direct result of their evaluations.

We have two large criticisms of the project. Both criticism will be related back to the goals stated in the opening section of this paper.

First, it is difficult to install new widgets. This works against the tradeoff of resources and availability. It would be best if the forms could be written as easily as a letter. The writer could respond to email questions from readers with new forms, perhaps incorporating new widgets, by the next few days. At this moment, the software is rewritten to install a widget, due to the restricted nature of message rewriting.

Second, the web-like nature of the pages is not captured. There is not the feeling of a journey, of diving into links, when using the WLP. It is the author's opinion that the participatory nature of the Web is crucial in this regard. This author has not yet found a method to integrate this into the WLP.

5 Conclusion

The Web is an excellent vehicle for computer education, especially for supplemental materials. Motivated students can broaden their classroom experiences, allowing more to be done in the four short undergraduate years. The HTML language provides a reasonable basis for widget building, however the WLP implementation has not be completely successful in tapping its power. New widgets are hard to install: writers are not encouraged to follow their impulses when creating widgets.

The author misinterpreted his own definition of "widget," attaching it wrongly to the HTML form which resulted by the rewriting process, rather than the rewriting process itself. In a next revision, the WLP widget front end will concerning itself only with the extraction, presentation and integration of widget parameters, and the actual rewriting code will be considered external to the WLP software. A separate callback mechanism will be attached to widgets and invoked by widget return messages to handle those form-like aspects required of widgets.

References

[1] Burton Rosenberg, "Reversibility in Computer Architecture," B.Sc. Thesis, MIT 1980.

[2] Jean Piaget, *Play, Dreams and Imitation in Childhood,* translated by C. Gattegno and F. M. Hodgson, Norton, NY, 1962.

[3] Seymour Papert, "Teaching Children Thinking," MIT AI Lab Memo No. 247, Oct. 1971.