MTEACH: A Language-Based Framework For Didactic Multimedia Production

P.L. Montessoro[1], D. Pierattoni[1], D. Cortolezzis[2]

[1] Dept. of Electrical, Management and Mechanical Engineering, University of Udine
I-33100 Udine, Italy
{montessoro, pierattoni}@uniud.it
[2] INSIEL – University of Udine, I-33100 Udine, Italy
Daniele.Cortolezzis@insiel.it

Abstract. Multimedia production is generally considered to be an expensive task. This is true when dealing with complex graphics, animation, artistic effects, music and high quality video editing, but didactic applications do not require such advanced features. A conventional lesson can be effectively reproduced in a multimedia environment using a small set of multimedia features and a simple, regular structure. However, didactic application often consists of a huge amount of multimedia material and logical interconnections. The MTEACH approach is based on a methodology supported by a language and a compiler, and allows the author of a didactic multimedia production to work at a higher level than conventional authoring tools. The "running code" of the multimedia application is generated in standard formats starting from a formal description of the lessons, providing a consistent and easy-to-navigate hypermedia structure.

1 Introduction

Didactic represents a natural application field for multimedia systems and authoring techniques. Traditional live lessons are basically "multimedia": the teacher talks, while explaining slides and pictures, showing objects and so on. With some improvements and adaptations, live lessons can be placed on an interactive CD-ROM or a Web site, as a support in media-based training and distance learning courses. Unfortunately, a well-known effect is that even few hours of teaching generate a huge amount of material (texts, audio, video, pictures, etc.). Thus, managing and handling large files and ramified data structures become a difficult task for both users and producers. Common authoring tools, like ToolBook [1] and Macromedia Director [2], offer powerful functions for animation controls, programming and graphics. Authors that are not highly skilled at dealing with multimedia production are able to obtain a simple result with a limited effort. Advanced functions exceed the requirements of basic end users, which are often disoriented by an enormous choice of commands and low-level instructions.

Moreover, a considerable amount of manual work is necessary to collect the various features, especially because these tools are designed for the largest number of authors, including those for which the artistic aspect is the main part of the product. However this fact could not be a constraint in professional multimedia publishing, where authors are experienced programmers and work often in concurrent engineering teams, it often represents an unpleasant overload in case of simpler approaches for less skilled users. Another consequence is that multimedia production costs are often a big problem for high level didactic, such as for university or professional courses that cannot count on the profits of massive sells.

Further, reusability of parts of the final product is limited and additional manual work is necessary to restore all the links, in order to preserve the consistency of the multimedia data structure. This limitation descends from the strict embedding of the original data sources (audio, video and image files) inside the code generated by the authoring tool.

Cross-portability between different production environments, for example merging ToolBook and Director files is nearly impossible, since each tool uses an internal coding system for the project description and generates archives with different formats. Working at a higher level would instead require a tool that describes the structure of the hypermedia, instead of directly implementing it.

Web-oriented authoring tools, like Microsoft FrontPage, provide some functions to preserve link consistence when the Web content is modified, or when a part of it is moved in a different context. However, considerable work is still required to create links, to fill template pages with the available material, and to check the logical correctness and consistence of the links in the final product.

Nevertheless, the total freedom left to the author when creating the outline and assembling the collected material can lead to a hypermedia not homogeneous in its parts. This makes navigation difficult and integration between different products almost impossible. Many authors do not recognize these aspects as disadvantages, being more interested in the artistic aspects of the work. Anyhow, this is not generally true for didactic productions, where lucidity and uniformity of style are key factors to gain the end user's comprehension.

2 A New Approach To Didactic Multimedia Design

When handling multimedia-teaching material, reusability of modules is one of the most important features. For example, advanced courses can recall parts of basic courses, or smaller courses can be rearranged out from more extended ones, regarding the same subjects but modifying their order and/or the details of the treatment.

Most lessons are modularly organized in chapters and arguments, which are accompanied with a store of scheduled pictures, schemes and practical examples that enrich the explanation. We think that the teacher's approach to multimedia production would be easier, if the methodology for the hypermedia description offered a set of rules as coherent as possible with the structural elements of a common lesson.

Graphic environments are the most common user interface for authoring tools. This allows authors to develop their projects in a friendly workspace, and often simplifies the identification of a large variety of toolkit features. Models and templates enhance the visibility of the available instruments; on-line preview areas on the workspace can also present the direct effects of certain modifications on the final product.

A language-based approach could not offer the same helpful features. Although for less experienced authors the difficulty of learning a structured language could seem larger than discovering the instructions of a graphical tool, the value of a language-based hypermedia description is evicted when recursive procedures have to be applied on large hypermedia products. Reproducibility of operations is easier when dealing with a plain text source: text editors have a set of macros and automation features that are more powerful than repeating sequences of visual commands on a GUI.

Further, the regularity of the hypermedia data structure and the reusability of the teaching material are often issues which are difficult to be solved, if artistic features are complex and variously mixed in a high-level graphical authoring tool.

A language that makes the hypermedia project very similar to the index of a book will achieve this target; nevertheless, it allows keeping the project development at a higher level, not depending from toolkit's internal representation schemes. A simplified tool also requires an automated creation of links between data sources and the output, and their dynamic adaptation if some changes are made in the given description.

On the other hand, such requirements imply a set of technical issues regarding both data sources and hypermedia description format. In order to obtain the full modularity at each layer, starting from data up to the project schema, the original multimedia sources should preserve their own identity in the output data structure.

Open data formats should be preferred for both sources and generated output. This assumption achieves the full cross-platform compatibility and allows to collect, modify and reuse the teaching material at any time and with the lowest conversion cost. It gets also through the disadvantages of most proprietary file formats, which require their own run-time libraries and are often platform-dependent.

Nevertheless, the hypermedia project structure should be based on a plain, portable, editable, source file (ASCII): such an open format would represent a common base for the interoperability between different production environments and could also stimulate further implementations of enhanced authoring tools.

3 The MTEACH Methodology

The MTEACH authoring framework is based on automated code generation, starting from a structured language which describes the content of each lesson in terms of "primitive" multimedia elements: audio, video graphics, text, etc. The MTEACH language allows the teacher to describe the hierarchical structure of the lesson in the same way the index of a book is written; the keywords list associated to each element represents the logical interconnections of the arguments that will become glossary entries and hyper textual links.

From the technical point of view, the system can be easily extended to several different platforms and operating system environments, only by adding new interface modules to the code generator of the compiler. In particular, if an authoring tool provides application interfaces to external hi-level programs, it can be controlled by MTEACH to reduce or eliminate the need for manual work.

One of the main purposes of MTEACH is to reduce the hypermedia production time. To achieve this goal, the first step consists in defining a methodology that preserves most of the normal preparation and teaching activity of the author and that leads to a straight sequence of tasks.

Firstly, the teacher prepares slides, pictures and animations to be used during the lesson; most of the didactic material should be ready before the lesson starts. Anyhow, the teacher can hand-draw pictures during the class, if exercises or detailed clarifications are needed.

The structure of the lesson is then described in the MTEACH language. This step provides a guide for the following production phases and guarantees that the audio and video recordings will match the logical flow defined by the lesson's schedule. Late modification of the structure of a didactic multimedia product often requires re-recordings of the teacher's speech to fit the new order of the arguments.

After that, the lesson is audio/video recorded, possibly live with students, in order to make the recording as natural as possible. If studio recording is preferred, the arguments can be individually recorded, possibly following the same order they will appear in the final product.

Audio/video recording is further digitized and edited, so that live recordings are splitted into the elementary modules (clips) defined by the structure of the lesson. Slides, pictures and animations are modified, if necessary, according to the actually recorded talk; for example, new slides can be prepared to reproduce additional drawings, which could have been added on the fly during live lessons.

At this point, all the elements needed for the code generation are ready. The structure described in the MTEACH language contains all the information necessary to build the links of the hypermedia structure. Compiler's template files will be used as definition of the graphical aspect of the output, as explained in the following section.

4 The MTEACH Language

The formal description of the MTEACH language, regarding the detailed explanation of syntax, lexical conventions and data types, would go beyond the purpose of this presentation paper. Nevertheless, current implementation is under development, application fields are getting larger, and new elements are to be added, according to the experimental results. Thus, a formal scheme would not still offer a complete and stable rule set for every kind of hypermedia description.

As said before, the description resembles the hierarchical index of a book. Traditional chapters become "lessons" and the sections within each chapter are called "arguments". Each argument is a module that groups a collection of different types of documents together. "Clips", the most important type, are made of audio/video tracks and several synchronized images, or "slides". Synchronization is not obtained during the audio/video editing sessions; instead, it is featured by the automatic code generation. The way synchronization is achieved depends on the target operating environment; in current implementation, for example, the Real Media [3] synchronization capability is used.

The kernel of the MTEACH language definition.

```
keyword_list   ::=  <identifier>
                    [ , <keyword_list> ]
lesson         ::=  LESSON <identifier>
                    ( [ <keyword_list> ] )
                    { arguments }
arguments      ::=  argument [ arguments ]
argument       ::=  ARGUMENT <identifier>
                    ( [ <keyword_list> ] )
                    { contents }
contents       ::=  clip [ clip ]
clip           ::=  CLIP <identifier>
                         <filename>
                    ( [ <keyword_list> ] )
                    { slides }
slides         ::=  slide [ slide ]
slide          ::=  SLIDE <identifier>
                         <filename>
                    ( [ <keyword_list> ] )
                    AT <time>
```

The MTEACH language requires that timing information is associated to each slide related to the starting point of the audio/video track. A textual description of timing information could appear more complex and less effective than a graphical interface, but plain ASCII format still allow easy editing and manipulation when the description regards a large tree structure. Interactive graphical interfaces do not scale equally well, even though an additional program (with its own graphical interface) could be used to compute and insert timestamps into the source file.

An example of the MTEACH language.

```
lesson "Ulnar nerve exposure at the elbow"
       ("Ulnar","Exposure")
{
   argument "Sites of entrapment"
            ("Entrapment", "Elbow")
   {  clip "Sites of entrapment" "sites.rm" ()
      {  slide "View of the elbow" "elbow1.gif"
```

```
                ("Anatomy", "Elbow") at 01
          slide "Inside the elbow" "elbow2.gif"
                ("Anatomy", "Elbow") at 10
       }
    }
    argument "Surgical steps"
             ("Operating techniques")
    {  clip "Skin incision" "incision.rm" ()
       {  slide "Starting the incision",
                "skin_incision.gif"
               ("Skin incision", "Elbow") at 01 . . .
```

The lesson could be enriched by other documents, such as texts, animations and images not linked to any audio/video track. The language is currently being extended to include more data types. A key feature of MTEACH is the automated generation of the several links, which are needed to logically interconnect topics. A list of keywords (such as "Ulnar", "Exposure", etc. in the example) is associated to each element in the description: this additional information will be used to build a multi-linked glossary during the code generation. Another set of automatically generated links provides a sequential visit of clips of the multimedia lessons.

5 The MTEACH Compiler

Automatic link generation makes MTEACH different from the conventional authoring tools and provides reusability of the multimedia material after it is organized in the lesson structure.

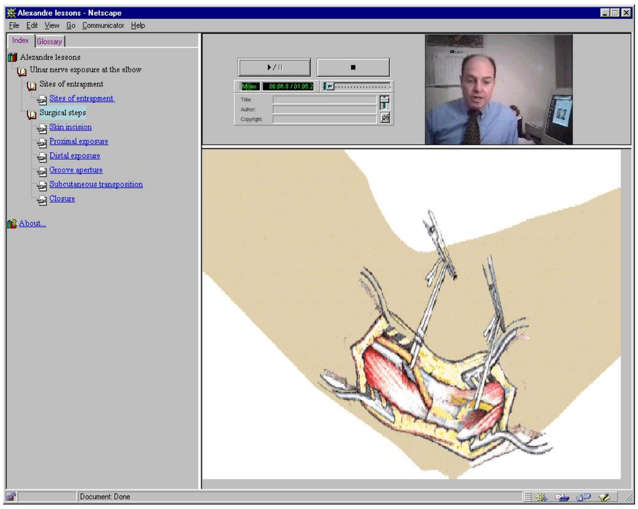

Fig. 1. Shot of a microneurosurgery application.

Moving a part of a lesson description (clips, slides, timing, keywords, etc.) to another project does not lose the structural information. All the links are rebuilt when the new code is generated: this is a nice feature, since even very short lessons may contain hundredths of links.

The MTEACH compiler has been developed using lex and yacc [4]. A data structure is built to internally describe the hypermedia structure and is visited during the code generation phase.

During the code generation phase, the compiler reads a set of template files and inserts into them the actual information described by the internal data structure. This approach allows the use of conventional editing and authoring tools to build and test the templates. Of course, this is possible only if the internal structure of the file of the chosen authoring system is known, or at least some APIs (Application Programming Interfaces) are accessible to control the editing feature of the authoring tool. Fig. 1 shows a shot of the final product in a microneurosurgery teaching application. Code generation in the current prototype is targeted for HTML files with JavaScript procedures and RealMedia [5] audio/video encoding and synchronization. Animations use the animated GIF format [6] with timing information.

6 Summary

A simple and regular structure is suitable for most didactic applications of the multimedia technologies. Their value resides in contents and in richness of links to better describe the relationship between the topics, rather then in artistic and graphical effects.

MTEACH is limited to didactic or didactic-like productions based on a regular structure. The language is still being extended and some new features are being added, but a running prototype is already available and it is being used to understand the needs for future extensions.

To test the easiness of usage by users without specific knowledge in computer science, MTEACH is currently being used to produce some lessons in the field of microneurosurgery, whose examples appear in the pictures of this paper[1].

Another direction of development regards the creation of interfaces to other authoring tools, including ToolBook, Macromedia Director and Shockwave.

References

1. Asymetrix Learning Systems Asymetrix ToolBook web page: http://www.click2learn.com/products/
2. Macromedia Director Technotes web page: http://www.macromedia.com/support/director/
3. Real Networks Real Media File Format web page: http://www.real.com/devzone/documentation
4. Schreiner, A.T., Friedman, H.G. Jr.: Introduction to Compiler Construction with UNIX. Prentice Hall, Upper Saddle River, New Jersey (1985)
5. Real Networks Synchronized Multimedia web page: http://service.real.com/help/library
6. Compuserve Graphics Interchange Format, Version 89a. ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats/

[1] Authors wish to thank dr. Alberto Alexandre from EUNI (Treviso, Italy) for his interest in this project and for his collaboration in the didactic microneurosurgery application.