NAVIGATION SYSTEMS, ARCHITECTS AND ENGINEERS

Terry Judd
Biomedical Multimedia Unit
The University of Melbourne, Australia
tsj@unimelb.edu.au

Abstract

Interactive multimedia (IMM) navigation systems rely on two key non-graphical components, a schematic or architectural framework and an associated programming solution. Two primary framework types are recognized – branching (structured) and referential (unstructured) – although most navigation systems include elements of both. Programmed solutions to navigational frameworks typically display varying degrees of independence from the frameworks they support, and the role of authoring tools in promoting such independence is discussed. The concept of an authentic navigation system, in which the navigational framework and associated programming are fully integrated, is introduced, and two IMM packages that employ such a system are described. Some advantages of authentic navigation systems are discussed.

Keywords

multimedia design, navigation systems, software architecture, authoring tools

About Navigation Systems

Navigation systems are a pivotal part of IMM projects. Interface elements aside, they comprise two key components; an underlying schematic architecture or framework and the programming or software engineering solution that implements it. Typically, the architecture closely follows the conceptual framework around which the project's content is woven. However, as will be demonstrated, the engineering component of the system can exhibit varying degrees of independence. Oliver and Herrington (1995) identified three types of navigational frameworks within multimedia/hypermedia – linear, hierarchical, and referential – representing the beginning, middle and end points of a continuum based on the degree of linking between individual screens (nodes). The linear end of this continuum is typified by slide show style presentations (as produced by PowerPoint) while purely referential hypermedia are common fare on the world-wide-web. However, comparing these three categories of frameworks in terms of their inherent structure, rather than their degree of linking, reveals a clear distinction between linear and hierarchical type frameworks (i.e. branching frameworks) on the one hand, and those based on referential linking (i.e. networks) on the other. The former exhibit various degrees of structure while the latter are essentially unstructured – most IMM navigation systems typically contain elements of both. As demonstrated by Misanchuk and Schwier (1992), branching frameworks can be characterised using a simple notation describing the degree of branching within the framework and the number of nodes between branches. Figure 1 provides a demonstration of this for a variety of frameworks created from the same ten putative nodes. Frameworks based on referential linking cannot be described in such terms. Each framework is unique and is the product of its constituent links. Thus, all of the branching frameworks in Figure 1 (Figure 1a-d) can be fully characterised by a list of their component nodes (e.g. for Figure 1c the list is 1, 2, 2.1, 2.1.1, 2.2, 3, 3.1, 3.2, 3.2.1, 4), while the referentially linked examples (Figure 1e-f) require a complete list of their component links (e.g. for Figure 1e the list is 1-2,1-3, 2-4, 2-5, 5-6, 5-8, 5-9, 6-10, 8-4, 8-7) – information that is implicit in list of nodes for the branched frameworks.

Earlier, I mentioned that programming solutions to navigational frameworks don't always parallel the structure of the framework in question. This typically occurs when referential linking is used to emulate branched frameworks. It is prevalent in HTML-based hypermedia, where all links are by definition referential, but is widespread across a range of IMM development platforms and authoring tools. The use of referential type linking to create non-referential frameworks is not in itself problematic. However, as will be demonstrated, in certain situations, the use of authentic solutions to essentially branched frameworks can provide real advantages to developers.

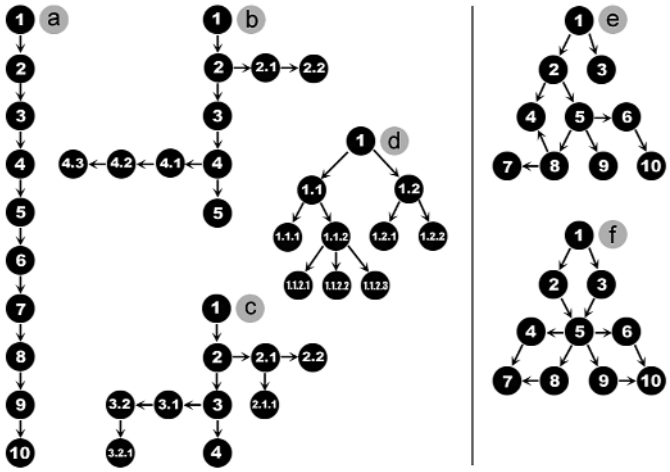

Figure 1: Alternative frameworks based on ten putative nodes; (a) to (d) - branched frameworks, (e) and (f) - referentially linked frameworks

About Authoring Tools

The choice of authoring tool can subtly (or not so subtly) influence both the design and implementation of IMM navigation systems. In the previous section, I described how HTML-based hypermedia makes exclusive use of referential linking. Yet, many WYSISYG web-authoring tools (e.g. Dreamweaver, GoLive) adopt a plainly hierarchical metaphor for displaying/browsing a site's content. This 'mixing of metaphors' can prove invaluable for developers who create web sites that adopt primarily branched navigational frameworks. Once we move away from HTML to the more traditional IMM development tools the situation is very different. With the exception of Authorware, which also employs a 'flowchart' or hierarchical approach for both displaying and building a project's content, the majority of authoring tools adopt a strongly linear development metaphor. So much so, that it can be difficult to implement highly branched navigational frameworks.

In card-based authoring tools (e.g. HyperCard, SuperCard, MetaCard, ToolBook), content is arranged on a 'stack' of virtual cards (nodes) between which the user navigates. Authoring tools based on a timeline metaphor (e.g. Director, Flash) typically divide what are essentially frame-based 'animations' into a series of labeled segments using 'markers' that are navigated in the same way as the individual cards in the card-based tools. PowerPoint represents a special case of card-based tools in that it places content on a sequence of virtual screens, however, unlike most of the card-based and timeline-based tools mentioned, it lacks the scripting/programming capabilities necessary to implement anything other than purely linear navigational frameworks. PowerPoint aside, each of these authoring tools typically provides the developer with two linking methods– linear and referential. In SuperCard for example you can choose to navigate in either direction in a linear sequence using the 'go next' or 'go previous' commands or referentially link to any card in the sequence using the syntax 'go to card' followed by the card's name. These commands are virtually identical across authoring tools, so that in Director, for example, the only difference is that the term 'marker' replaces 'card'. Irrespective of the tool used, however, developers must use referential linking to created branched frameworks and the characteristic solution to most primarily branched frameworks will include both linear (to navigate along branch segments) and referential (to navigate between branch segments) links.

Authentic Navigation Systems

The term authentic is used here to describe navigation systems in which the programming solutions for the system are based directly on the navigational architecture they implement, and this architecture, in turn, directly parallels the conceptual framework of the project it supports. As outlined in the previous section, both steps require that the dominant navigation paradigm of the development environment and the default methods for linking content are circumvented to some degree.

What advantages do authentic systems offer over those developed on an ad-hoc basis? In many cases none (as far, at least, as the end user is concerned). Navigation systems for small to medium sized multimedia or hypermedia projects are relatively easily implemented without either special attention to the design or implementation of the navigational framework, or undue interference by any constraints the authoring tool may place on the developer. However, where scalability and modularity are important considerations, authentic systems can provide distinct advantages to both developers and clients. The following two examples describe the design and implementation of two authentic navigation systems within multimedia projects developed by the Biomedical Multimedia Unit at the University of Melbourne. Both projects were created using Macromedia's Director authoring software.

Example 1 – DNAexplorer is a computer facilitated learning (CFL) package designed to explore key concepts in the field of bioinformatics (Kennedy, Judd, Keppell, Ginns, Crabb & Strugnell 2001). Its gross architecture consists of four tutorials accessed from a central menu screen. Within each of the four tutorials, the primary content is navigated by 'turning' the corners of a sequence of 'pages', presented within a 'manila folder'. On key pages, supplementary pages of content (termed learning loops) are accessed via colour-coded tabs on the side of the page. Navigation within learning loops is again via the page corners and loops are exited either via the appropriate tab or by progressing to the end (or the beginning) of the loop. In either case, when the loop is exited the user is returned to the screen from which the loop was originally accessed. Learning loops are also occasionally embedded within learning loops and are accessed and navigated in the same way.

The navigational framework described above is essentially a branched linear structure in which the central 'axis' can support up to two degrees of branching (i.e. a learning loop within a learning loop), and is homologous to the 'feedback loop' branching structure described by Misanchuk and Schwier (1992). It was implemented by adopting a naming convention for screens similar to that suggested in Figure 1 (see Figure 1c). A series of generic programming routines were then written to track the name of the current screen, provide access to and from the learning loops, and enable navigation to the next or previous screens in the current sequence. These routines call on the authoring tool's referential linking function to navigate between screens, but the screen names passed to this function are dynamically generated based on the name of the current screen and which navigation-related interface component (i.e. page corner or colour tab) is accessed.

Aside from improved reliability, the main advantage this navigation system over an equivalent ad-hoc system, was the additional flexibility and ease of use it provided during development. Throughout the development phase of the project, new content was added and existing content rearranged on a regular basis and generally the only changes necessary to the navigation system were the renaming of affected markers or screens. Because the system is driven by generic programming routines it was generally sufficient to attach the appropriate navigation-related interface components and, provided new or relocated screens were correctly named, navigation to and from them would function correctly.

Example 2 – Communicating With the Tired Patient is a CFL package designed to develop medical students' clinical communication skills (Liaw, Kennedy, Keppell, Marty & McNair, 2000). In this package, the user conducts a virtual interview between a doctor and patient, from the perspective of the doctor, drawing from a database of prerecorded video and audio clips. Briefly, the interview proceeds as follows. The doctor greets the patient asks them about their problem and the patient responds. The doctor (user) is then presented with between two and four questions they can pose, which vary in subject matter and/or style of delivery. A selection is made and the patient responds.

This cycle of doctor question, patient response continues until the interview is completed. Following each patient response, the user is presented with a series of comments relating to the patient's response. The answers to these questions and the records of the various doctor questions and patient responses selected and viewed throughout the interview are also combined to create an up-to-date transcript of the interview, which can be viewed at any time.

The navigational framework that best fits the conceptual framework described above is a regular hierarchy in which each node is defined by a doctor-question patient-response couplet and gives rise to between two and four similar nodes at the next level. However, because some nodes and their subordinate structures are repeated within the hierarchy, a second referential framework is used to re-map duplicated nodes to existing nodes. Thus, no more than two copies of any given node need exist within the hierarchy. Unlike *DNAexplorer*, which uses an embedded navigation framework, The *Communicating with the Tired Patient* package employs a virtual framework that is dynamically created from a series of external files. These files include a settings file containing a description of the interview hierarchy, its component nodes and associated video, audio and text-based resources, in addition to the video, audio and text-based resources. These resources are loaded on request as the user navigates the interview's virtual framework. As for *DNAexplorer*, the programming used to implement the navigational system consists of a number of generic routines based on a system of referential linking. However, because in this case the framework they implement is virtual rather than physical, the links they dynamically generate are paths to external files rather than names of existing screens within the package.

The navigation system outlined above replaces an ad-hoc system developed for the initial version of the package and offers substantial advantages and improvements over it. In the previous version, the navigation framework was created within the authoring environment and implemented using Director's in-built referential linking function. While not especially difficult to achieve, for some of the reasons discussed above in the 'About Authoring Tools' section, Director's timeline (i.e. linear) development metaphor is relatively ill suited to the creation and management of highly branched navigational frameworks. As a consequence, making alterations to the resultant network of nodes was especially problematic. Moreover, the planned addition of new interviews would, due to their unique character, have required the creation of entirely new navigational frameworks and solutions. By implementing an authentic navigation system we were able to improve scalability and simplify maintenance within a truly modular environment. Moreover, because content development was now independent of software development, development costs and times for implementing additional interview or for evaluating alternative interview structures within existing interview were substantially reduced.

References

Kennedy, G., Judd, T.S., Keppell, M., Ginns, C., Strugnell, R.A. & Crabb, B.S. (2001). DNAexplorer: Computer facilitated learning of bioinformatics using a situated model. In *Proceedings of Ed-Media 2001. World Conference on Educational Multimedia, Hypermedia and Telecommunications*. AACE. Tampere. Finland.

Liaw, T., Kennedy, G., Keppell, M., Marty, J. & McNair, R. (2000). Using multimedia to assist students with communication skills and biopsychosocial integration: An evaluation. *Australian Journal of Educational Technology, 16* (2), 104-125.

Misanchuk, E.R. & Schwier, R. (1992). Representing Interactive Multimedia and Hypermedia Audit Trails. *Journal of Educational Multimedia and Hypermedia*, *1* (3), 355-372.

Oliver, R. & Herrington, J. (1995). Developing effective hypermedia instructional materials. *Australian Journal of Educational Technology 11* (2), 8-22.