KnowledgePro Windows: the order of merit?

Martin Reynolds

Department of Computer Science, The Victoria University of Manchester, UK

Abstract

The producers of KnowledgePro look set with their latest release of KPWIN (KnowledgePro Windows) to fulfil Richard Hale-Shaw's prophecy that it will become 'one of the most powerful visual development environments' (Hale-Shaw 1992). Comparisons are drawn in this paper between the KPWIN family of products and other authoring tools. The conclusion is that KPWIN is worthy of being included in any courseware developer's tool set. Reasons for preferring a tool from the KnowledgePro family of products for courseware development over three main competitors - Authorware, Toolbook and Visual Basic - are explained, and the merits of KPWIN and KPWIN++ (a version that generates C++ code) are examined.

Introduction

Courseware development is invariably a difficult and costly task. The quality of the end-product often depends critically on the types of tool used to develop it. Over the last decade, a wide variety of courseware and authoring tools has been produced (Barker 1989), and the range continues to grow. Methods and reasons for selecting one particular tool in preference to another are therefore of considerable importance.

The introduction of courseware can affect changes which produce organizational, computational, or motivational benefits. The effectiveness of courseware in an educational context is solely dependent on its use (Kulik and Kulik 1991). And the more experimental the educational objectives are, the more important it is for courseware to act as a reflexive tool rather than as a rubric. For courseware to be accepted, its quality must justify its use in a course as a whole, in supporting particular tasks and in helping learners how to learn. When substantial projects or complex products are envisaged, the cost-to-quality ratio is smallest when the author's role is divided between a specialist subject expert, programmer, educationist, and graphic artist/photographer. Storing the work of each professional separately allows the most to be made of each team member.

It is as unfortunate for the programmer as for other team members that the information available about software tools is distorted by the hidden third-party code of surreal demonstra-

tions, the hidden agenda of marketing hype and the hidden naivety of the eager but inexperienced author. However, using the nature of costs, benefits and effects as a guide, the relative merits of each product can be determined, and the effect that visual programming, object orientation, integrated development environments, multimedia and hypermedia can have on the quality of learners' environments can also be revealed.

Programmers have always used tools to help generate program code. This has been particularly true for code that represents the human-computer interface and its screen elements such as windows, buttons and boxes. In order to improve the quality of communication between the programmer and the other team members involved in the development of a product, this visual dimension to programming has been extended in some products to encompass the paradigm of interaction as well as visual appearance. Visual programming tools provide the other team members with a structure with which to order the resources content. The programmer shares this structure, and in building an interface for the user is restricted to the tools paradigm of conversing with the content. When the other team members' intentions for the interaction match those imposed, the programmer will find that realizing the team members' wishes in code terms is straightforward. Conversely, when there is a mismatch in intention, the programmer will find implementation difficult if not impossible. The more visual the tool, the more matched the team members' intentions and the tools' capabilities for interaction will need to be, and the fewer means the programmer has to specify the implementation. The implicit assumptions that visual programming tools contain about team members' intentions tend to reflect the structure of the programming language underlying the tool.

KnowledgePro and other products

KnowledgePro began as a DOS product (KPDOS) in 1986. After establishing itself in an Artificial Intelligence market, a Windows version (KPWIN) was released in 1990 when it won *PC Magazine*'s Best Product of 1990. In 1992, a capacity to generate C code was added (KPWIN++). Its three main competitors are now Authorware Professional, Toolbook and Visual Basic.

Authorware Professional (£800 educational) provides iconic flowcharting as the structure for authoring. It constrains interaction to a single instance, a single window of fixed size, and a linear pathway, and it represents its intentions by just 11 icons. It provides many variables that can be set to alter the way the program operates but, paradoxically, this warns of its limited ability to be programmed - if your intentions do not match one of the 11 icons, beware! Unlike the other products mentioned in this paper, it does not support all the standard Windows screen elements. It requires write access to the directory in which it resides when running, and does not provide file-pointers or database functions, but rather encourages the hard coding of data in an application. Iterative procedures are noticeably slow, and in not being able to act as DDE (Dynamic Data Exchange) server, halt while other applications run or parse more than 13 characters to other applications, Authorware exhibits a limited ability to converse with other programs. A 3% royalty is payable on applications that are re-sold. This reflects the product's suitability for presentations or other passive applications whose production involves no team-work and no programmer, whose delivery involves no multitasking or networking, and whose future involves no maintenance (Daeche and Reynolds 1993).

Toolbook (£250 educational) adopts a form-filling approach to authoring which is especially suited to the production of electronic books. Toolbook thus uses a book metaphor for its

forms, though it would be misleading to suggest that this limits its use to linear presentations of passive text. The links between 'pages' can be displayed in iconic form and manipulated using the mouse - one abstraction removed from Authorware's iconization of screen elements. Each Toolbook page has a foreground and a background. In the foreground, screen elements are placed and sized by selecting them from mouse-driven toolboxes. In the background, Toolbook's event-driven Openscript language is used to define the reaction of screen elements. Although background scripts can be shared between pages or between books, this represents only a limited ability to inherit or export variables and code. Openscript has been optimized for the description of a user-interface rather than for the description of the consequences of a user's action. As if to underline this, Toolbook provides a routine to exchange Windows messages and Openscript messaging so that it can easily front code for processes stored as DLL (Dynamic Link Libraries) or running as hidden or secondary programs, typically written with tools such as Turbo Pascal. Openscript is syntactically simpler than Basic but similar in structure - it is reliant on event-driven equivalents of GOTOs and GOSUBs rather than function descriptions.

Visual Basic (from £90 educational) provides a visual environment not unlike that of Toolbook, but for a version of Basic. For *toolbox* read *custom control*, although unlike Toolbook there is no graphical way of connecting forms. Double-clicking on a screen element placed on a Visual Basic form allows Basic code to be entered which describes the form's reaction to the user-interface. As with Toolbook and Authorware, much of the code that will eventually make up the application - the code that represents the custom-controlled screen elements - remains unseen and uneditable. Since it belongs to the Basic family, Visual Basic is a typed language, and there are limitations on how code or data can be shared between forms. The assumption, as with Toolbook, is that different forms will represent different content or jobs. It becomes awkward if not contrived to code anything other than simple data-driven applications. Although toolboxes invite the amateur programmer, Toolbook and Visual Basic are properly targeted at the professional who wants to get a quick prototype up and running and who wishes to take the advice of non-programming colleagues when it comes to screen design.

KnowledgePro (from £100 educational) uses a topic paradigm which, unlike the form or icon, is a definitive rather than a visual representation of the underlying KnowledgePro language. There are tools provided to generate topics for screen elements, and definitives in the language aimed at fast prototyping. In function these are similar to Authorware's icons or Toolbook's toolboxes, but they have the advantage of generating readable and editable code. This makes KnowledgePro accessible to the amateur programmer, yet by not imposing a structure, allows authors the freedom to define their own, typically by developing a shell for their courseware. The programmer can mix and match techniques to optimize the implementation's performance. It is implicitly assumed that the programmer may wish to use object-orientated programming, and artificial intelligences like backward chaining, neural networks and matched lists, yet is free not to do so. The limits of C on inheritance, persistence and encapsulation do not exist in KPWIN, yet one can write C in-line with KnowledgePro ++ code. However, KnowledgPro is typeless, and syntactically closer to LISP than to C.

OOD (Object Orientated Design) has been found to aid the process of communication between other team members and programmer more than other formal methods (Graham 1991). It is particularly well suited to the evolutionary rather than revolutionary or revelationary development cycles. Work in the educational field is almost exclusively co-operative and

evolutionary in nature, continual user-involvement refinement and maintenance being important factors. KPWIN clearly supports OOD, and in so doing maximizes the re-use of code and simplifies the maintenance of code that represents many similar but different situations. Its strength in evolutionary development is underlined by the KPWIN++ version's ability to generate C code integrally from the KPWIN 4GL, and use standard C compilers to make and compile it. Visual Basic, Toolbook and Authorware are object-based rather than object-orientated and better suited to revolutionary and revelationary developments, in which substantial re-coding is necessary when an author's requirements change or when the speed of compiled rather than interpreted code is required (Winblad *et al* 1990). As an application's complexity grows so does the demand for effective communication between programmer and other team members. Concomitantly, the programmer will demand an increasing ability to optimize the code run.

At one time, programmers would first have to write a program's code using one tool, then make it with another, link and run it with a third, and (if they were fortunate) debug it with a fourth. IDEs (Integrated Development Environments) bring all these tools together in one interface and can vastly improve a programmer's productivity. As a corollary, costs can be reduced while the quality of code and the ease with which code can be maintained or changed can be increased.

KPWIN supplies an IDE familiar to most C-level programmers - one can trace or step through code as it is run, access context-sensitive help, build libraries of functions defined in KnowledgePro to be re-used, and overload functions defined in KPWIN with personalized implementations. Technical support is available in the UK via Knowledge Garden (Europe), and there is a UK user group. There are a number of code libraries available to extend the core KPWIN language including dBase, Clipper, Q&E SQL and Archive. KPWIN source code is provided for all the tools of the IDE as well as the shareware multimedia tool Textpro which exemplifies the replaying of digital video and animation.

The Toolbook and Visual Basic IDEs are extensible, but by the addition of custom controls or toolboxes rather than code libraries per se. Toolboxes for standard screen elements, rudimentary graphics and text come as standard. Ones for audio and video elements are available. Visual Basic (Professional edition) comes with custom controls for deriving charts, graphs and tables from Microsoft Access databases or SQL servers. The breadth of functionality represented in their third-party extensions is perhaps the greatest strength of Visual Basic and Toolbook. Both products sport string-handling functions, online help and debugging tools. But even with Microsoft's support, Visual Basic cannot overcome the constraint of its underlying Basic-like language. Toolbook and Visual Basic seek to reconcile the difficulties in re-using code defined in their own terms by making the inclusion of functions written in more primitive languages relatively straightforward.

Authorware can be extended only with functions built into lower-level languages like C or Pascal. Consequently there are no comparable third-party libraries available. It has no debugger and no online help, nor any UK technical support from its producers Macromedia. Authorware does supply rudimentary tools to create simple graphics or animations, and icons to represent erase effects and video replay, though these encourage an amateurish approach to multimedia. Successful multimedia avoids the pitfalls of inappropriate use by combining the experience of film makers, photographers, graphic artists and animators with that of a behaviourist, and the quality sought by these professionals is more likely to be realized through the custom controls available in Toolbook and Visual Basic or the functions in KPWIN which

create elements and display those created by third-party applications (though it is worth mentioning that many multimedia applications remain undelivered because the cost of delivery is unsustainable).

As for hypermedia, that subclass of multimedia which uses windows, icons and the mouse to link one phrase in a text or region in an art form with another in order to bring out an author's view of their relationship, some form of conversation with the learner needs to be facilitated, and some form of mutual adaptation needs to take place, in order to transform the 'hyperspace' into a useful learning environment (Harri-Augstein and Thomas, 1991). Although Toolbook does have a hypertext toolbox, KPWIN has a unique ability to create hyperlinks on the fly as well as from files and from program code. And it has a specialized set of commands for dealing with the list, tree and network structures. These aid the implementation of artificially intelligent or expert elements as adaptable models of learners' or teachers' meaning. KPWIN's support for OOD can be a great advantage to programmers asked to develop groupware, a means for groups to model meaning (Baecker, 1993). OOD complements the record locking and data compression of KPWIN's dBase and Archive extensions by aiding the implementation of programs which can run concurrently and across a network. In programming terms, KPWIN is an ideal vehicle for the illustration of various techniques without the encumbrance of learning the syntax and paradigms of languages as divergent as CLISP and C.

Conclusion

The only certainty about the use of newer technology is that it is more expensive than older technology. Educational effectiveness is dependent on use rather than its technological implementation. The total total amount of time spent using any tool represents only about 20% of the total time spent on a project, whatever the tool and however easy or complex the project, so that the choice of tool does not have much bearing on cost. But tools differ in what they allow one to do - each has its own specialities and limitations. The choice of tool can therefore determine what one is able to produce, and so what benefit might be gained (Pedlar and Reynolds 1993). It is unlikely that any tool can replace the assistance of a specialist programmer unless the author's intention matches that of the tool exactly. The chance to buy additional libraries or toolkits may make a match more likely than in the past, but if there is a mismatch between the tools and the author's intention, or if multifarious, large, complex or distributed projects are contemplated, it is certain that KnowledgePro's flexibility and maturity as a language put it ahead of Toolbook, Visual Basic and Authorware, in an order of merit which values cost savings and benefits without compromise on performance or maintenance.

References

Baecker, R.M. (1993) (ed), *Readings in Groupware and Computer Supported Co-operative Work*, San Mateo CA, Morgan Kaufmann.

Barker, P.G. (1989), 'KnowledgePro: a review and assessment', *Engineering Applications of Artificial Intelligence*, **2**, 4, 325-338.

Daeche, N. and Reynolds, M. (1993), *Report on Authoring Tools for the Electronic Design Education Consortium*, unpublished report, August 1993.

Graham, I. (1991), *Object-Oriented Methods*, Reading MA, Addison-Wesley.

Hale-Shaw, R. (1992), 'The visual development environment - more than just a pretty face?', *PC Magazine*, **1**, 6, 195-237.

Harri-Augstein, S. and Thomas, L. (1991), *Learning Conversations*, London, Routledge.

Kulik, C. and Kulik, J. (1991), 'Effectiveness of computer-based instruction - an updated analysis', *Computers in Human Behaviour*, **7**, 74-94.

Pedlar, J. and Reynolds, M. (1993), *The National Oral Surgery Computer-Assisted Learning Programme - Final Report*, unpublished report, Nuffield Foundation, February 1993.

Reynolds, M. and Pedlar, J. (1993), 'Choosing an authoring tool for a national CAL programme', in *Proceedings of CAL '93*, London, Pergamon (in press).

Winblad, A.L., Edwards, S.D. and King, D.R. (1990), *Object-orientated Software*, Reading MA, Addison-Wesley.

Addresses

Knowledge Garden Inc., 12-8 Technology Drive, Setauket, NY 11733, USA.

Knowledge Garden (Europe) Ltd., Mount Batton House, Fairacres, Windsor, Berks, SL4 4LE.

Ursula Hayes
KnowledgePro User Group
PEVE IT
Department of Computer Science
The Victoria University of Manchester
Oxford Road
Manchester M13 9PL