Centre for Computing in the Social Sciences

A Critical Evaluation of Multimedia Toolbook

A SIMA Report

Document Notes

Authors	Simon Price and Li Lin Cheah
Date	9 May 96
Version	1.0
Document Ref No	SP22-16

Summary

This report gives a detailed evaluation of Asymetrix Multimedia ToolBook 3.0 focusing on the strengths and weaknesses of the tool in different application domains. It enables developers seeking to assess Multimedia ToolBook's suitability for their application to make more informed decisions relating to their choice of authoring tool.

Much of the content of this report is also directly applicable to the recently released Multimedia ToolBook 4.0 whose features are listed in the appendices. However, it is not yet possible to write a similar report for version 4.0 as the requisite body of real world experience and knowledge does not yet exist.

Centre for Computing in the Social Sciences
University of Bristol
8 Woodland Road
Bristol BS8 1TN

Tel. 0117 928 8478
Fax. 0117 928 8473
Email: ccss@bristol.ac.uk
Web: http://sosig.ac.uk/ccss

Contents

About the Authors

Simon Price

Simon Price is Director of Software Development for the Centre for Computing in the Social Sciences at the University of Bristol. The Centre currently hosts a number of EU and UK Internet and multimedia projects including the British Computer Society Medal winning WinEcon ToolBook software for which Simon was the Senior Programmer.

He is the chairman of the Association for Learning Technology (ALT) ToolBook User Group, was an organiser of the 1994 and 1995 ToolBook Users Conferences, and an invited speaker at the 1994 Asymetrix World Developers Conference in Seattle. He has also given invited lecture tours on multimedia software development as far apart as Sweden and South Africa.

In his role as a ToolBook consultant, he has developed software or delivered training for Armadillo Systems, Centre for Computing in Medicine, Department of Health, Economics and Business Education Association, Employment Services Commission, Institute of Dental Surgery, ICS Solutions, ICL, Leicester University, Multicosm, Oxford University, Southampton University, UK Association of Cardiothoracic Anaesthetists, UMIST, Unilever and the University of the West of England.

He holds a B.Sc.(Hons.) in Computing Science from Staffordshire University and is a Member of the British Computer Society, IEEE Computer Society and ACM. For the eight years prior to joining the University of Bristol he worked as an analyst programmer in the computer games industry, designing and programming twelve published titles.

Li Lin Cheah

Li Lin Cheah is Systems Analyst of Software Development for the Centre for Computing in the Social Sciences at the University of Bristol. She was Analyst Programmer on the WinEcon project which won a British Computer Society Medal in October 1995 and an European Academic Software Award in June 1996. Her current work at the Centre includes the WinEcon Authoring Tools, which is the template on which WinEcon was developed, and providing technical support for WinEcon. Her research interests include the assessment and evaluation of interactive learning software.

Li Lin holds a B.Sc. in Mathematical Statistics from Monash University, Australia and a Dip.Comp.Sc. from La Trobe University, Australia. She has worked as an analyst programmer on a number of software projects in Bristol, Kuala Lumpur and Melbourne, specialising in the design and development of database management systems. Prior to joining the Centre in 1993, she was systems development manager on the National Food Survey Statistical Analysis System project which was developed for the Ministry of Agriculture, Fisheries and Food based at the Department of Economics, University of Bristol.

1. Introduction

1.1. Rationale for Report

Asymetrix Multimedia ToolBook 3.0 has been around for more than two years now and has been used as the main authoring tool in numerous multimedia and computer based training projects around the World. It is a mature product, having undergone a pre-release beta phase of approximately a year and having been quietly upgraded to version 3.0a (the so called "service release" or, more simply, bug fix release) shortly after its general release. It is also a well understood product with known strengths and weaknesses which have been discovered by the now substantial user and developer base over the life of the product.

Despite this, the current "best" source of published information for developers considering ToolBook for a multimedia application is Asymetrix's own product information and a few brief reviews in magazines and journals. Faced with this lack of information developers are forced to take a high risk gamble in their choice of authoring tool. The consequences of choosing the wrong tool can range from complicating the development process unnecessarily through to making the project goals unattainable within the given resources. In either of these cases, there is a very real chance that problems resulting from an inappropriate choice of authoring tool will cause the project to fail. The authoring tool is deeply entwined in almost every aspect of a multimedia project and once development is under way it is usually prohibitively expensive to switch tools. This makes it of pivotal importance that a project makes the correct choice of tool before development begins.

Choosing an authoring tool is too important to trust to the manufacturer's product information or to brief reviews and product comparisons; developers require detailed information and evaluations of a tool's suitability for different problem domains. A report, giving a detailed description and evaluation of ToolBook for different types of multimedia development would meet this need. This report sets out to provide that information and to assist developers in making the important decision of whether ToolBook is appropriate for their project. Despite the authors' own enthusiasm for ToolBook it is clear that it has both strengths and weaknesses and we will be equally satisfied if a reader decides not to adopt ToolBook because the weaknesses outweigh the strengths for their project. Indeed, if this report were to stop just one project from choosing the wrong authoring tool then the outlay by the JISC would, in all likelihood, be recouped. Furthermore, by making developers more able to make an informed assessment of a candidate tool the quality of projects, which otherwise might have wasted most of their effort in bending an inappropriate tool to their needs, will be raised.

1.2. Relevance to Multimedia ToolBook 4.0

At the time of writing this report, Multimedia ToolBook 4.0 has just been released and there is already discussion on the Internet of the forthcoming Multimedia ToolBook 5.0. The wealth of knowledge and experience which is currently available for 3.0 does not yet exist for 4.0. However, in many respects, 4.0 is actually closer to a version 3.1; it is not substantially different from 3.0 and consequently, the majority of this report applies equally well to both ToolBook 3.0 and 4.0; it is only in the finer points of tools, reliability and timing that real differences emerge. "Appendix F - ToolBook 3.0 Product Information" of this report lists the key features of the new release and highlights the differences to version 3.0.

Over the coming months, ToolBook 4.0 will start to be used in anger on real world projects and the knowledge base and experience required to produce a "A Critical Evaluation of Multimedia ToolBook

4.0" report will be developed. It is anticipated that such a report will be produced later this year and will compare and contrast the capabilities of the two versions in addition to presenting case studies of its use.

1.3. Scope and Structure of Report

This report, "A Critical Evaluation of Multimedia ToolBook 3.0", as its title suggests, presents an evaluation of Multimedia ToolBook 3.0. However, this implicitly involves evaluating another Asymetrix product - ToolBook 3.0 (non-multimedia, or "vanilla" ToolBook 3.0) which is a subset of the full Multimedia ToolBook 3.0. For this reason, the report begins by looking at the system requirements for each of these two programs and the skills required by prospective developers. Then, the core functionality common to both ToolBook 3.0 and Multimedia ToolBook 3.0 is examined before moving on to look at those features unique to the multimedia version. From there the report goes on to look briefly at other members of the ToolBook 3.0 product family and at how ToolBook performs in real world applications.

Reflecting this approach, the report is divided into the following chapters:

- System Requirements
- Developer Skills Requirements
- Overview of ToolBook 3.0
- Differences between ToolBook 3.0 and Multimedia ToolBook 3.0
- Versions and Upgrades
- Alternatives to ToolBook
- Case Studies
- Conclusions

Supplementary information is included in the appendices at the end of the report.

1.4. Acknowledgement

The authors are grateful to Jan Chipchase of Software Development at the Centre for Computing in the Social Sciences who undertook the task of converting this document into HTML files to enable its viewing on the World Wide Web.

2. System Requirements

This section presents the minimum and recommended system requirements given by Asymetrix in their product documentation and then comments on how realistic they are for various application types and modes of use. It also discusses the issues involved in using ToolBook 3.0 across a local area network and across the Internet. Finally, the prospects for running ToolBook 3.0 on a non-PC Windows platform is examined. Throughout, attention is drawn to known incompatibilities and problems areas.

2.1. Software Requirements

ToolBook was developed as a Windows PC optimised and Intel specific program. Unsurprisingly the software requirements listed by Asymetrix are:

- Microsoft MS-DOS 3.1 or higher
- Microsoft Windows 3.1 or higher

Some time after ToolBook 3.0 was released Microsoft released Windows 95. ToolBook 3.0 works well under Windows 95 and the program windows and menus take on the Windows 95 look and feel. However, there is no built in support for the Windows 95 interface objects like push buttons, check boxes and radio buttons although these can (relatively) easily be mimicked by using ToolBook's ability to customise the look and feel of interface objects.

Although not stated on the packaging, the software requirement for running under Windows 95 is simply:

- Microsoft Windows 95 or higher

Regrettably, Asymetrix omitted to specify one extra software requirement which would have made their own life and that of developers using the ToolBook family of products a lot easier. The missing requirement, in the opinion of the authors of this report, is:

- Windows compatible display driver (small font - **not** large font)

As it is, this is not a stated requirement and few developers are aware of the minefield this issue of supported display drivers is. The heart of the problem lies in the way ToolBook maps its own internal screen co-ordinates (e.g. the position of a button or graphic in the ToolBook window) on to the physical pixel co-ordinates of the underlying graphics hardware. For normal, so called "small font", display drivers this mapping is 15 ToolBook screen units to 1 real pixel on screen. Unfortunately, end-users are able to configure their Windows system to use a "large font" display driver so that on high resolution screens (e.g. 800x600 or 1024x768) the text, menus and window borders are bigger and easier to read on low-cost monitors (i.e. sub 17"). Why is this a problem? Well, in doing this, the mapping of ToolBook co-ordinates to screen co-ordinates ceases to be 15:1 and becomes, say, 18:1 thereby throwing out the layout of the ToolBook screen. In some cases this may not be a problem, but because the ratios are not multiples of one another, the amount by which the scaling occurs will be irregular - so that a regular grid drawn at 15:1 will be unevenly spaced at 18:1. The reverse is also true - a regular grid drawn at 18:1 will appear irregular at 15:1.

The situation worsens if fonts and word wrap are taken into consideration, where text comfortably filling a box in a small font driver will often overflow the space in a large font driver. Asymetrix's policy on

this is, rather unhelpfully, that developers should leave enough space in any text field for this not to matter. However, this would severely limit the type of screens which could be designed using ToolBook and is, for most applications, a ridiculous suggestion.

If developers were aware of this problem from the outset, then there is a simple work around which is to add this missing software requirement to the software requirements of their own product. This does not mean that developers can only develop for small font drivers; rather, that developers must develop using the same driver type as that used by the intended end-users of their product. Typically, 640x480 drivers are all small font so that screen resolution is not a problem. For higher resolutions, small font drivers are the most common, in the authors' experience, in UK higher education student labs. However, many institutions do use large font drivers but can sometimes be persuaded to switch to small font ones. It is therefore important that prospective ToolBook 3.0 developers research their prospective target user base's display driver usage before commencing any development.

Where the users' precise display driver configuration is not known because the software is for general publication rather than for a specific institution, it is possible to check the screen mapping ratio during start up of the application and to advise the user accordingly (see ToolBook's on-line help for `sysPageUnitsPerPixel` and check that this is, say, "15,15" if your software requires a small font driver).

In summary, this arguably missing software requirement is a problem which is not unique to ToolBook but is serious if the ToolBook developer fails to take this into consideration at the outset of a project. (Incidentally, this problem has not been resolved in ToolBook 4.0 either).

2.2. User Hardware Requirements

Asymetrix do not specify separate hardware requirements for end-users and developers; a single minimum machine specification is intended to cover both. For non-Multimedia ToolBook 3.0, the minimum specification stated by Asymetrix is as follows.

- 20MHz 386 SX processor or higher
- 2.5MB of free hard disk space for ToolBook 3 runtimes (plus space for courseware)
- At least 4MB RAM (8MB recommended in User Manual, 6MB recommended on packaging)
- 1.44MB (3.5") floppy disk drive
- Mouse or pointing device
- Windows compatible graphics card (VGA or higher)
- Sound card (optional)
- CD (optional)

For Multimedia ToolBook 3.0, the specification on the box is slightly higher in the areas of CPU speed, RAM and, of course, adds multimedia hardware such as a sound card and CD-ROM drive. Again, a single specification is given to cover both end-user and developer.

- 20MHz 386 or higher
- 8MB hard disk space (minimum to run - actually needs more for developer's installation)
- Sound card (recommended)
- CD-ROM drive

How do these specifications measure up in practice? Taking the non-multimedia specification first, it is claimed that ToolBook 3.0 works on a 20MHz 386SX with only 4MB RAM and, technically speaking, it

does. However, depending upon the rest of the machine configuration, it can take several minutes to simply load and initialise ToolBook on a 386SX even before ToolBook starts loading your application into itself. Then, once loaded, simple operations like selecting an item off a pull down menu can take an unusable length of time - particularly if it is the first time you have performed the operation. In fact, very few types of authoring operation are practical on this speed of machine and it is difficult to see how one could realistically develop interactive graphical software under such an unresponsive environment. Only a limited variety of applications can realistically be developed for this hardware specification. Applications requiring high speed animation or anything other than tiny bitmap graphics require a lot of work on the part of the developer to obtain anything approaching a usable execution speed. In particular, the time taken to navigate from screen to screen (or page to page in ToolBook terminology) can easily exceed a minute if the pages contain large bitmaps and if some initialisation has to take place on entering each page.

Moving to the multimedia minimum specification claimed by Asymetrix, this sets the entry point at 386DX with 8MB of RAM. A 386DX is faster than a 386SX but it is the addition of an extra 4MB RAM that makes a noticeable difference in the performance of the package. Under this specification, the pull down menus operate at a usable speed and it would be possible, but not enjoyable, to develop ToolBook applications on this specification of hardware. As with the non-multimedia specification, this specification is still too low to practically support all the types of graphical and interactive software one might wish to develop in ToolBook. It is, at least, plausible that a subset of application types, which do not make great demands in the areas of graphics and animation, could be developed on this level of hardware. However, if software video clips are used then the load times and frame rates are not, in the opinion of the authors, good enough for serious use.

Clearly, the minimum hardware specifications quoted by Asymetrix are at the very lowest extreme of machines capable of running ToolBook and lie some considerable way off a recommended end-user platform and a long way off a recommended developer platform. In fact, Microsoft Windows itself is not very responsive or usable on such low specification machines and so it comes as no great surprise that ToolBook under this environment is also slow and unresponsive. However, it is not uncommon for developers to develop their application on a high end Pentium PC and just assume that their software will be equally usable on machines down at the bottom of the supported hardware list. Such assumptions can render an application unusable by the intended end-user base and the importance of testing software on the lowest specification target hardware can not be over stressed.

Although there is still a large installed base of 386 machines running Windows, these machines are at, or have already come to the end of their useful life and are being replaced with 486 or Pentium based systems. Indeed, at the time of writing this report it is difficult to buy a PC below a 50MHz 486 and a 75MHz Pentium is becoming the entry level machine even for home users.

However, in a cost conscious world, it is still important to know the minimal practical hardware specification. So, if Asymetrix's stated minimum hardware specification is too low, then what is a practical minimal specification? The answer to this question is different for end-users and developers. The authors' recommended minimum specification for developer hardware is given in the next section; however, for end-users it is as follows (for both ToolBook 3.0 and Multimedia ToolBook 3.0 alike).

- 33MHz 486 DX processor or higher
- 4MB of free hard disk space for ToolBook 3 runtimes (plus space for courseware)
- 8MB RAM
- 1.44MB (3.5") floppy disk drive
- Mouse or pointing device

- Windows compatible graphics card (Super VGA or higher)
- Sound card (optional)
- CD (optional)

If software video is required (e.g. Video for Windows), then a higher CPU clock speed is highly recommended. However, the single most effective add-on to this specification would, for graphically dominated applications, be the addition of a graphics accelerator which brings a significant improvement[1] in screen redraw rates and animation speed. After this, the next best area to optimise would be the speed of the hard disk or CD ROM drive as appropriate.

2.3. Developer Hardware Requirements

Asymetrix do not give a separate list of hardware requirements for developers. However, to develop efficiently and effectively in ToolBook requires a slightly higher specification machine than is required by end-users of applications written in ToolBook. The authors' suggested minimum developer hardware requirement is listed below.

- 66MHz 486 processor or higher
- 24MB of free hard disk space (plus 9MB extra if WIN31WH.HLP Windows API help is loaded)
- 8MB RAM or more
- 1.44MB (3.5") floppy disk drive
- Mouse or pointing device
- Windows compatible Super VGA graphics card (at least 800x600 with the required number of colours)
- Sound card (optional)
- CD (optional for ToolBook, essential for Multimedia ToolBook)

Of these requirements, the increased screen resolution is one of the most important for the simple practical reason that ToolBook litters the screen with a variety of toolbars and floating palettes (tool button windows) and unless the developer has a larger screen area than the target screen area, a large proportion of the ToolBook window is obscured. If developing for a 640x480 resolution screen, developing with ToolBook in a 640x480 window centred in a 800x600 resolution screen leaves plenty of space around the window to place the potential multitude of tool windows.

2.4. Network Support

There is no difference between ToolBook 3.0 and Multimedia ToolBook 3.0 in terms of network support.

In practice, ToolBook works well across a network provided machines have a minimum of 8MB RAM. Also, configuring base memory (memory which lies below 640K) to have as much free space as possible can make quite an improvement in overall performance of both Windows and ToolBook. Sadly, many current network drivers occupy a great deal of this bottleneck space.

[1] Windows graphics acceleration hardware is common in modern display cards. Unfortunately, vendors and computer support departments often install the standard Windows VGA display drivers rather than the custom display drivers supplied by the graphics card manufacturer (usually on separate floppy disks to Windows). Unless the custom display drivers are used, the potential performance improvements are unlikely to be realised.

Multiple users can share ToolBook applications on a network if the ToolBook utility TB30NET.EXE is in each user's path. TB30NET guards against two or more users changing a book at the same time. This utility comes with ToolBook and is installed in the same directory as ToolBook itself and so it will usually be in the path by default.

ToolBook supports two modes of use on any networked system.

Mode of use	File properties[2]	No. simultaneous users
read/write	rw	single
read-only	r	multiple

If a ToolBook file (aka book) is read-only, then any number of users can open and use that book simultaneously. ToolBook behaves as if each user were using their own private, separate installation of that book even though there is only one physical copy on the file server.

By contrast, if a ToolBook book is not set to read-only then only one user will be able to open the book. Until this user has closed the book, subsequent users attempting to open the same book will get a message saying that the book can not be opened because it is already in use.

The obvious design restriction which results from the inability to give multiple users simultaneous write access to the same book (via some sort of record/page locking) is that some external means of recording persistent information must be used in a networking application. For example, a computer based assessment package implemented in ToolBook must find some way of recording the users' scores in tests other than by simply saving the book. Common solutions to this sort of problem include writing to an ANSI text file, a Windows .INI file or a database such as Paradox™ or dBase™.

To be fair to Asymetrix, this restriction is an reasonable trade-off for impressive ease of data access within ToolBook and for the ability to perform a broad range of non-persistent write operations which may differ from instance to instance without any central data locking bottleneck.

In terms of performance, ToolBook itself takes somewhat longer to load across a network than from a local hard drive. The precise figure is highly dependant on the network infrastructure and the current network traffic loading but is typically in the range of 30-90 seconds in student teaching labs. The total load time will also include the time to load the ToolBook book (i.e. the application). It is simply not possible to quote a typical time range for this as it depends upon the degree of initialisation which takes place on entering the application and upon the volume of graphics, video and sound on the first page in the book. A minimum figure would be around 15 seconds for a simple, non-graphical first page and rising to around 90 seconds for a graphical first page with a small video clip. However, the only real way

[2] Novell™ networks usually require the network administrator to explicitly GRANT access to specific files or directories in order to make them writeable by end-users. This is because it is normal for ToolBook applications to be placed in directories which, somewhere in their parentage (i.e. up the file hierarchy), have a read-only directory that will automatically override any setting made to its child directories. The GRANT command by-passes this read-only inheritance rule and permits write access to specific files or directories.

to establish loading time across a network is to try out a representative prototype under realistic loading of the network; near local drive performance can be obtained in a student lab with only one machine accessing the network, but this gives absolutely no indication of performance under load.

Once loaded, performance across the network as the user navigates around a ToolBook application is dominated by the amount of graphics, video and sound data needed for each page visited. ToolBook automatically caches graphics so, provided the graphics are not so large and numerous that the cache is flushed for every page, there will be a marked improvement on subsequent pages re-using the same graphics.

Asymetrix recommend that, for improved performance, ToolBook be installed onto the local hard drive (assuming there is one) and that just the applications (books) be stored on the network. Unfortunately, this is frequently at odds with institutional computer support policies which insist on all executables residing on the server. One work around for this is to configure the machines on the network to, on booting, check their file sizes and dates against those in the same directory on the server and to replace any local files which differ with the master versions on the server. This allows local copies of the ToolBook executables to be maintained while providing some protection against tampering by users.

Asymetrix make no recommendations for the situation where the workstations on the network are entirely diskless; no local hard drive exists on which to install the ToolBook executables locally. In such circumstances, the best course of action is to opt for extra RAM on the machines so that Windows can run more efficiently and, possibly, so that a RAM drive can be established to hold the executables. Applications intended to be run on diskless workstations should be designed with this in mind and options to switch off non-essential graphics can be included to dramatically improve their network performance.

2.5. Portability

ToolBook 3.0 is a PC only product. Asymetrix have repeatedly stated that they monitor the Mac and UNIX markets but that these markets are not large enough to justify the sizeable investment required to develop versions for these platforms. This situation has remained unchanged in the upgrade to ToolBook 4.0 and seems unlikely to change in the near future (although the new Java product may be a step along the road to platform portability).

The Internet ToolBook discussion list from time to time discusses running ToolBook 3.0 under emulators on, say the PowerMac™ or SoftPC™. However, performance is a major problem and there are numerous little incompatibilities which render this approach to portability useless or, at best, extremely restrictive.

3. Developer Skills Requirements

An important consideration, possibly the most important consideration in some cases, when choosing an authoring tool is the training and expertise required to use the tool. No matter how powerful the tool, if the author needs several years of specialist experience to produce a usable application then that tool may not be suitable. Alternatively, the author may decide that a specialist should be employed to undertake or oversee certain aspects of the development. In the case of larger scale projects, it is useful for project managers and programming leads to know a tool's skill requirements ahead of appointing staff.

The skills required to develop in ToolBook can be divided into three broad areas which are common to all applications. These are listed below and discussed in the following sections.

- Programming
- Graphic Design
- HCI (Human Computer Interaction)

This is by no means an exclusive list and other areas could be added including project management, testing, systems analysis, media production and so on. Also, domain specific skills (e.g. economics, physics, biology, etc.) are required at some point to develop computer based learning software. However, these excluded skills requirements are either independent of ToolBook and would be just the same for any other authoring system, and/or too specialised for the intended readership of this report.

3.1. Programming

Despite inferences to the contrary in the Asymetrix publicity and product information, most ToolBook applications development requires programming, or scripting as Asymetrix prefer to call it, to a greater or lesser degree. While it is possible to create pages (i.e. screens) and populate them with objects such as buttons, scrolling lists and graphics, in order to add functionality to these requires the use of the built in "scripting" language called OpenScript. ToolBook's OpenScript, despite its name, is a full blown programming language and to use ToolBook 3.0 effectively OpenScript programming is required.

3.1.1. How much OpenScript coding is required for different types of project?

This is a difficult question to answer as there are numerous ways of implementing exactly the same functionality but with varying levels of "ease of development", robustness, complexity, maintainability and so on. For instance, the script controlling a menu screen consisting of a series of navigation buttons could be implemented as a series of independent self-contained objects. Such a bottom up approach requires a lot of coding, but much of it is very similar. Alternatively, a generalisation could be made and the code could be factored up to a higher level of abstraction so that a single script controlled all the buttons. This abstracted approach requires less scripting but the script is more complex. A further approach still would be develop a reusable component consisting of a set of navigation buttons which could be used in this and any future book. Here, more coding is required to make the buttons independent and generalised. So, clearly, the amount of coding required for different types of projects is entirely dependant on the precise nature of the project.

However, some generalisations can be made. Reference materials and, so called, electronic page turning applications require relatively little coding although, when implemented by inexperienced ToolBook programmers, will consist of numerous cut and pasted copies of

functionally identical buttons. But even at this simple level, some esoteric OpenScript coding will be necessary to create a deliverable quality application. It has been the experience of the authors in their consulting work that most current multimedia applications fall into this category and that developers working on these projects progress well until they hit problems with the esoteric work required to finish off their application. This problem is by no means unique to ToolBook but it is perhaps less obvious that the problem is there in the first place because most of the core programming required to typical multimedia work is very short and fairly simple.

Projects with an element of computer based assessment (e.g. multiple choice questions) or record keeping (e.g. recording visited pages from session to session) require more coding still. Projects with elements of simulation, gaming, modelling and other rich interaction require just as much coding to implement in ToolBook as they would in Visual Basic™ or Visual C++™. The difference between the amount of code required for a rich interaction page and a simple hypertext page can easily be 10:1 and can rise to over 100:1 for complex interactions. If your intended ToolBook application is to contain rich interactions then serious consideration should be given to specialist tools (e.g. Excel™, Microfit™, Mathematica™, etc. for numerically rich interactions, or Access™, Oracle™, etc. for data rich interactions) more closely aligned to the interaction as this may bring more benefit than easily developing the less specialised elements which make up 95% of the pages but only 5% of the effort.

One approach frequently adopted by multimedia projects is to use ToolBook as the framework or container document and to develop custom extensions written in other programming languages. However, this approach requires both more coding and a greater degree of general programming expertise - it is not an approach for novice developers or non-programmers (unless they are prepared to invest a great deal of time acquiring new skills).

3.1.2. How long does it take to learn ToolBook programming?

OpenScript is a full programming language with an exceptionally large number of reserved words and alternative syntax structures. At its heart are the same selection, looping, procedural, functional and object constructs found in numerous other programming languages - although with non-standard syntax and names. Around this core are an extensive set of commands and functions for manipulating objects - allowing manipulation of properties such as size, colour and data. In other, more general purpose programming languages these operations would normally reside in a graphics library rather than in the programming language itself. The net effect of this is that OpenScript is exceptionally well integrated with its environment but at the cost of being daunting to a newcomer. The on-line help for each and every keyword makes life somewhat easier but new users seldom know how to use it efficiently[3].

The documentation accompanying ToolBook 3.0 includes a User Manual which is designed to lead developers through the process of learning all aspects of OpenScript and object manipulation. Opinions as to how good this documentation is vary from person to person. We, as authors' of this report, found this User Manual to be easy to use and helpful. However, we did have prior programming experience and were familiar with ToolBook 1.5 too. Other developers have expressed exactly the opposite view, raising concerns over the way the "how to program" issues are largely ignored. Fortunately, there are a number of third party books on Multimedia

[3] Press F1 key when cursor is on a word in the script editor for a context sensitive look-up of that word in the help file.

ToolBook 3.0 and numerous sources of further information on the Internet (see the appendices of this report for details of both of these). In fact, the Hustedde book is a book teaching computer programming that just happens to use ToolBook OpenScript as the language.

Regardless, ToolBook programming can not be learnt overnight and requires a considerable time investment. A potential ToolBook developer, unassisted by a more experienced ToolBook programmer and without prior extensive programming experience, would be unwise to allow anything less than a 4-12 week (full time) lead in time to climb the steepest part of the learning curve. Becoming fluent in OpenScript and relevant ToolBook-specific design techniques can easily take 6 months to a year of working with the software. Obviously (one would hope), becoming fluent in programming itself can take many years.

Despite the need for programming skills, most ToolBook developers are non-specialist programmers and yet they manage to produce some top quality material. So, this requirement should not necessarily be seen as an insurmountable barrier, rather it should be taken into account in scheduling and resource allocation. Also, it should be noted that a great deal of the effort involved in multimedia software development is not "programming" as such. In the authors' largest ToolBook development, the coding (which is what people normally think of as programming) only accounted for 20% of the "programming" work[4]. The rest consisted of user interface design and educational content (i.e. subject matter content) in roughly equal amounts. Therefore, mastering OpenScript programming is only a part of the overall skills set required to develop in ToolBook.

3.1.3. Does experience in other programming languages help?

Undoubtedly. OpenScript is only a proprietary programming language which supports all the programming constructs normally found in mainstream programming languages. While procedural in some respects, it is also semi object orientated in that it supports message passing and the concept of data and code in the same object.

Experience in programming in Hypercard or Visual Basic is especially useful as there are many similarities. Experience in C/C++ also helps to understand why and how ToolBook works and allows custom extensions to OpenScript to be written to add needed features.

Care is needed though, as a number of familiar constructs have slightly different semantics and it is easy to miss some of the powerful features of OpenScript by assuming the same restrictions on familiar operations apply. A careful read of the manual is always worthwhile even if commands look familiar.

3.1.4. Can the programming be done by someone other than the domain expert?

The design of ToolBook makes it feasible to divide up the development work so that, for instance, a reasonable experienced ToolBook programmer could write a basic application framework or template and an inexperienced programmer (or even a non-programmer) domain expert could then populate the template with content. These and similar approaches are examined in the case studies section of this report.

[4] Based on the recorded distribution of review effort in software reviews.

3.2. Graphic Design

3.2.1. Does ToolBook help non-specialists achieve attractive and professional looking screens?

ToolBook 3.0 provides little software or written guidance in this area and so the answer has to be, "It is very difficult." In fact, the sample applications and documentation accompanying ToolBook are by no means exemplar in this respect.

This is a major weakness of the package in one respect but it could be argued that this is not an area which should be dictated by the software tool. Indeed, it is the lack of layout restriction which has enabled developers to use ToolBook in a wide range of application areas. ToolBook 4.0 introduced a wizard and template based approach to creating basic page layouts but, in the opinion of the authors of this report, the resultant layouts are far from impressive.

3.2.2. Could graphic designers who know little or nothing about ToolBook provide useful input?

Yes. The basic screen design functions of ToolBook 3.0 would be familiar to most graphic artists through their existing knowledge of drawing packages. With only a little knowledge of ToolBook, a graphic designer can produce screen layouts which may be populated by a domain expert and made functional by a programmer.

3.3. Human Computer Interaction (HCI)

3.3.1. Does ToolBook enable effective human computer interaction (HCI) design?

ToolBook, unlike more prescriptive authoring environments, has sufficient flexibility to produce a wide range of interface designs. The down side of this is that it also has the scope to produce highly unusable designs.

3.3.2. How does ToolBook encourage or discourage good practice in this area?

Specialised and highly usable human computer interfaces can be produced provided that the developer has either some training in HCI and/or an ability to think like a user.

ToolBook has been used to mock-up and prototype interfaces so that user input can be obtained while there is still a chance to feed the findings back into development. This makes ToolBook an ideal RAD (Rapid Application Development) tool suitable for an iterative cycle of prototyping and evaluation until the prototype eventually becomes a deliverable product. This is in stark contrast to the traditional waterfall model of software development where the product is specified in detail before implementation.

Other than the RAD nature of the tool, ToolBook 3.0 includes no specific tools or documentation which substantially encourage good practice. The sample applications contain little evidence of having being designed for or ever been near a user. Similarly, the sample widgets (reusable software components) have inconsistent behaviour and look and feel.

3.3.3. Does ToolBook assist in measuring and controlling usability?

There are no direct examples of how to measure and control usability and advice on user trialing is minimal. However, ToolBook does have an outstanding ability to monitor its own environment, including user actions, without having to write complex and long winded code. Coverage, timing and logging are easily implemented. Also, it is a surprisingly simple matter (given a reasonable level of programming experience) to provide integrated user feedback collection or questionnaires.

4. Overview of ToolBook 3.0

This chapter gives an overview of the features of the ToolBook 3.0 system which underlies Multimedia ToolBook 3.0. The subsequent chapter sets out the differences between ToolBook 3.0 and Multimedia ToolBook 3.0.

A complete product feature list is not included in this chapter as such a list is available from the Asymetrix product information and a copy of this is included in the appendices of this report. Instead, this chapter gives a broad overview of ToolBook and expands upon areas not well covered in Asymetrix's own description of the software.

4.1. Authoring Environment

ToolBook uses a book metaphor throughout. A ToolBook file is called a book; screens are called pages; program code is called script (OpenScript); and so on. ToolBook books typically consist of a series of pages through which the user will navigate by clicking on buttons, hyperlinked text and other active objects on the page. Each page is composed of a background and a foreground. The foreground is unique to each page but a background can be shared by any number of pages. As well as displaying the current page in the main window of the book, it is also possible to simultaneously display other pages (or even parts of pages) in other windows, dialog boxes, tool bars or embedded windows in the current page using viewers. Books are normally run in a bordered window but can be set to run at full screen and to not allow task switching out of the application. Multiple books can be run simultaneously although Asymetrix recommend that multiple instances of the same book are not run simultaneously.

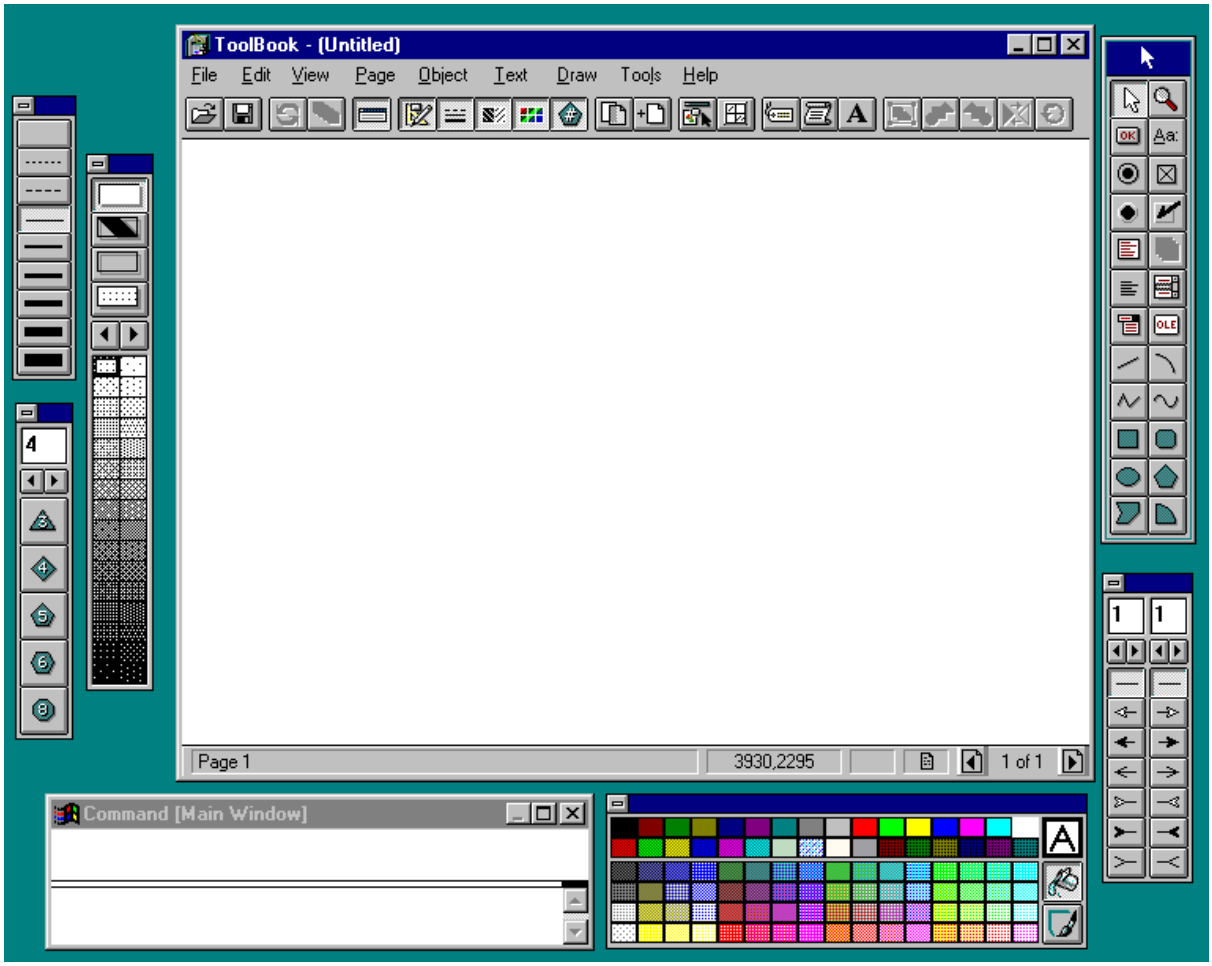
ToolBook books always require either the ToolBook runtime or the full ToolBook authoring tool to be installed in order to run. This is because, like Visual Basic™, ToolBook is an interpreted environment (running p-code or virtual machine code) rather than a fully compiled language running as native machine code. The ToolBook runtime is included free of charge and may be redistributed on a royalty-free basis as part of an application developed in ToolBook. However, this does mean that even the smallest "Hello World" application must be shipped on at least two 1.44MB floppy disks so that the runtime files can be shipped with it.

ToolBook has two modes of operation: author and reader mode. Creation and editing of applications is performed in author mode and then a key press (F3) switches to reader mode so that the material can be tested. There is no delay in switching between these modes and so the test-edit cycle is noticeably faster than in compiled languages like C/C++ or Pascal. The ultimate user of the finished application will only ever see the reader mode view (although, surprisingly, pressing F3 in runtime ToolBook generates an error unless explicitly trapped by the application).

A password can be specified to prevent users entering author mode and if the application is run under the runtime version of ToolBook it is impossible to enter author mode at all. Unfortunately, the password system is known to have at least one bug which allows unauthorised access to author mode. Also, passwords are not application transparent and are of little or no use in multiple book (file) applications which require the transfer of data between books: on the first attempt to access data from another book, the user is prompted to enter the password for author mode and code execution is interrupted if it is not then entered. This whole state of affairs with passwords appeared to be geared towards Asymetrix selling subscriptions to its Developer Support Programme which, amongst other things, gave developers a utility called Remover, to strip source code (the program scripts which enable someone to see how a program works and allow them to copy or modify it). However, with the release of ToolBook 4.0,

Remover became a built in feature of the standard ToolBook package at no extra cost to the developer. At the time of writing this report it seems likely that the ToolBook 3.0 Remover will be available free of charge off their Internet ftp site or web site.

The diagram below shows ToolBook 3.0 in author mode. The main ToolBook window is in the middle and is surrounded by a variety of tool "palettes" to draw objects and manipulate their properties. By default, only the drawing palette is visible and the rest are hidden. All palettes can be switched on or off. They are all shown switched on here to emphasise the need for developers to have a higher resolution display than the size of the ToolBook application they are delivering (e.g. at least 800x600 for a 640x480 sized application window). Without the larger display, much of the working area can be obscured by tool palettes especially as there are other tool windows such as the Command Window (shown below), the property browser, the script editor window and various dialog boxes which occasionally have to remain open for extended periods (e.g. the find dialog box).



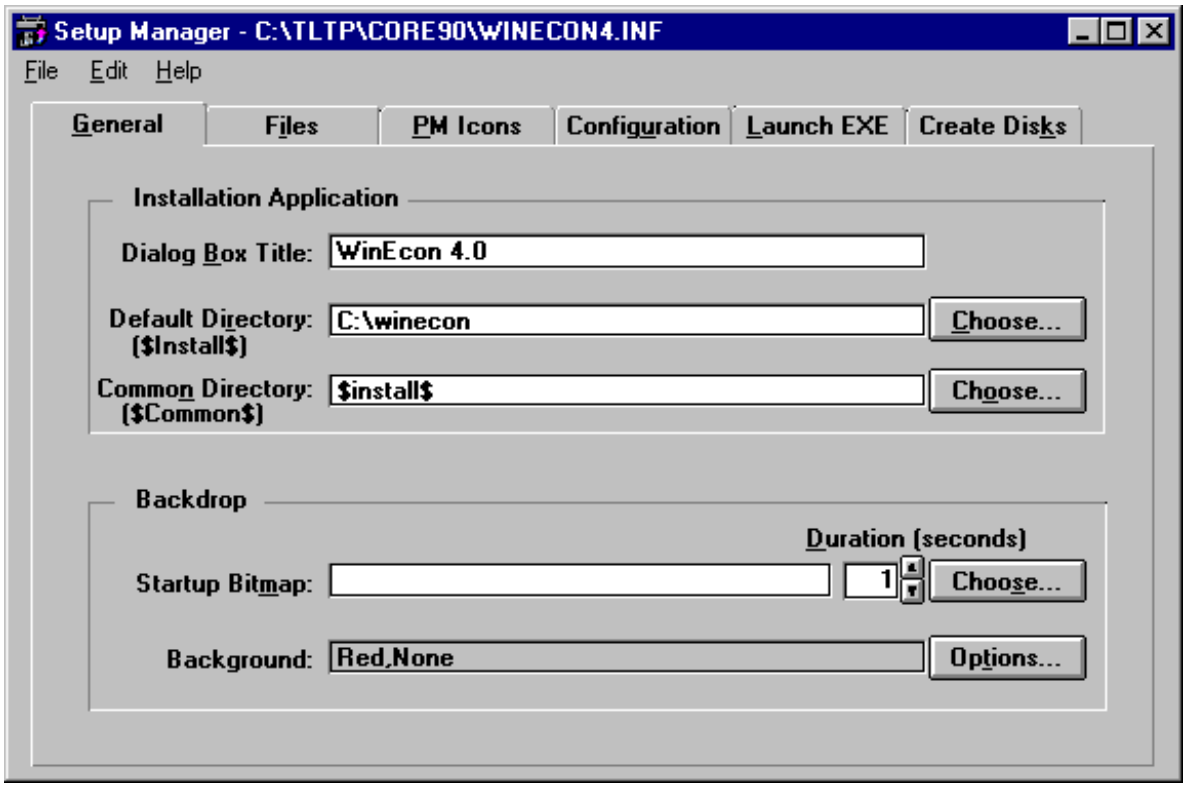
Getting started (which is not the same as completing) applications development in ToolBook is reasonably straight forward. The package includes an on-line tutorial, several sample ToolBook programs, an introductory manual, a reference manual and an on-line help system which is available from within ToolBook itself. Possibly the most useful of these after climbing the initial learning curve is the on-line help which can perform context sensitive searching (e.g. pressing F1 when the cursor is on a word in the script editor will open up the help file at that word or present a menu of nearest matches). The on-line help is a regular Microsoft Help file with numerous cross references and example code fragments which may be cut and pasted into the script editor. The main topic areas are as follows.

- Menu commands
- Step-by-step procedures
- OpenScript[5] reference
- Maps of screen elements
- Keyboard and mouse shortcuts
- Visual Basic and OpenScript terms (a comparison)

Having learnt how to create pages, populate them with objects and create buttons for navigating between pages, a quantum leap is then required in order to flesh out the application to have any significant functionality over and above an electronic text book. In order to make this leap the author must acquire programming skills and become at least partly familiar with the OpenScript programming language. The level of programming skill required can increase exponentially as the scale of the project increases linearly. In ToolBook 3.0, Asymetrix attempt to avoid this steep learning curve by supplying Autoscript re-usable scripts and a clip-book (called a "Widgets Book") of pre-programmed objects which may be cut and pasted into the application as required. This approach can work well in small projects provided that a suitable Autoscript or widget can be found and that the author has a moderate understanding of how the script/widget works. However, the resultant book typically has numerous copies of near identical objects, each with its OpenScript code slightly modified in some way. Subsequent global changes of behaviour or bug fixes require that each and every object be edited individually. Such practice runs contrary to commonly accepted good programming practice and is highly error prone. Perhaps Autoscript and the Widgets Book are best treated as a set of examples around which to base your own general purpose routines rather than as a direct resource to be used as is. Even so, in the hands of someone with even a small amount of programming experience impressive results can be achieved in a remarkably short time.

As a development environment ToolBook is largely self contained and entire applications can be produced using no other tools other than ToolBook itself and the small set of tools which accompany it. Of these tools packaged with ToolBook only Setup Manager (shown below) is absolutely essential. It is an easy to use (compared to Microsoft Setup) utility which allows the developer to specify a list of files in their application and then to build a distributable CD or set of floppy disks which also includes the ToolBook runtime. Compression is used and large files are automatically split over several disks if necessary. The resultant CD or floppy disks can then be shipped to users who simply run a:\setup in order to install the application on their system. A small application in non-multimedia ToolBook 3.0 can just fit on a single floppy along with the runtime. With Multimedia ToolBook 3.0, two disks is the minimum size of a distributable.

[5] OpenScript is ToolBook's built in, proprietary programming language.



It can often make sense to customise the ToolBook authoring environment to automate frequent tasks and add extra features. There are numerous ways in which this can be done and Asymetrix demonstrate one in their default tools system book[6]. The tools system book adds a number of extra menu items to the Tools menu which provide a property browser, layout tools and (multimedia version only) text indexing and searching plus spell checker. The tools system book is entirely implemented in ToolBook so it is possible to study it and see how to customise the environment yourself.

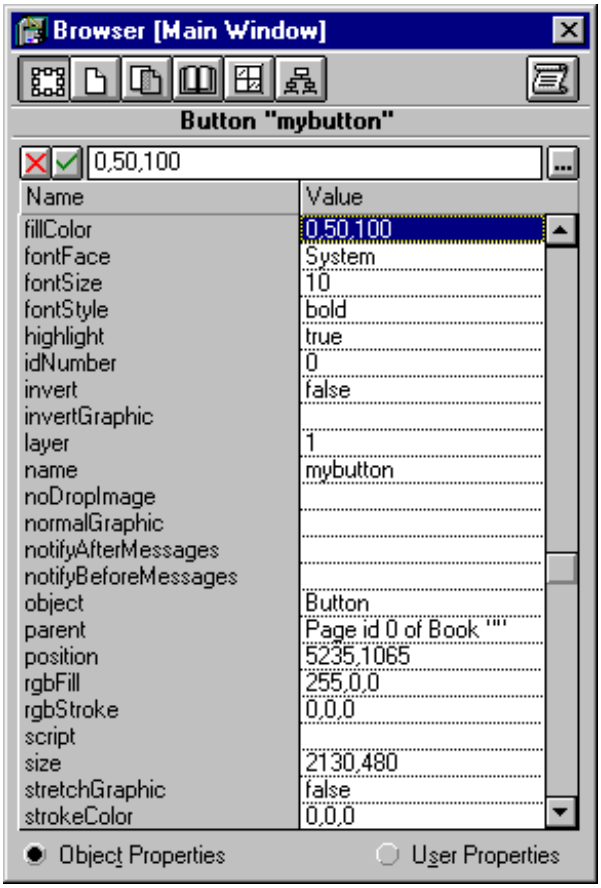
4.2. Objects

ToolBook books are constructed from "objects" - in fact the book itself is considered to be an object. Examples of objects include pages, their backgrounds, viewers (i.e. windows) plus all the graphical objects on pages such as rectangles, circles, pictures, buttons and text fields. Objects are connected together into a massive hierarchy with the book at the top, backgrounds below books, pages below backgrounds and objects on the page below the page. Simply drawing an object or creating a page implicitly links it into the hierarchy. Objects may be copied/cut and pasted through the Windows clipboard and when an object is cut/copied/deleted, all its "children" go along with it. For example, deleting a page will also delete all the objects on that page. This behaviour is as one would expect and it is only when programming scripts to bring these objects to life that the links in this hierarchy become critically important and, sometimes, less than obvious - particularly with objects on a background or with viewers.

[6] A system book is a normal ToolBook book which acts as a library of scripts for one or more other books. Normally, system books are never opened by the end-user and are only ever seen by the developer. Any book can be a system book for another book simply by adding it to the list of system books in that book.

Objects on a page may be grouped together to make composite objects known as groups. The group is an object in its own right and sits between its objects (its "children") and the page. Groups may be grouped inside other groups and so on. This is useful in manipulating objects as a single entity (e.g. simply moving a collection of objects around screen as one) and allows the creation of self contained software components (widgets) with their own in built functionality. The sample widgets book included with ToolBook contains many examples of such widget groups.

All objects have a set of properties which determine their appearance and behaviour. These properties can be manipulated directly while authoring or at reader mode (runtime) using OpenScript scripts. An example of direct manipulation would be setting the colour of a push button by selecting the object (clicking on it) and then clicking on the colour red in the colour palette or assigning the colour red (represented by 0,50,100) to the fillColor property in the property browser as shown below.



The equivalent script to do this would be something like this.

```
fillColor of button "mybutton" = red
```

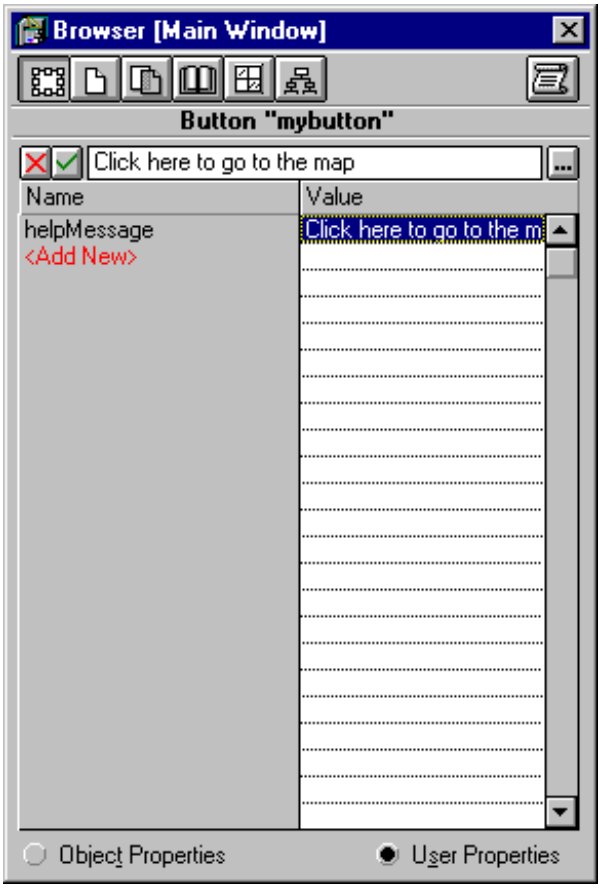
The property "fillColor" is a built in property which every button has. The on-line help lists all the built in properties for each object type and gives explanations of their role. Not all objects have the same range of properties. However, all objects do have a script property which is where the script for that object is stored and, interestingly, the script can be assigned and edited just like any other property (although not in runtime ToolBook).

It is possible to define your own properties for an object (which are effectively variables or constants attached to an object). Explicit declaration of these, so called, user properties is not required; they are

created when they are assigned to the object in a script or using the property browser. An example script assigning a user property to an object is as follows:

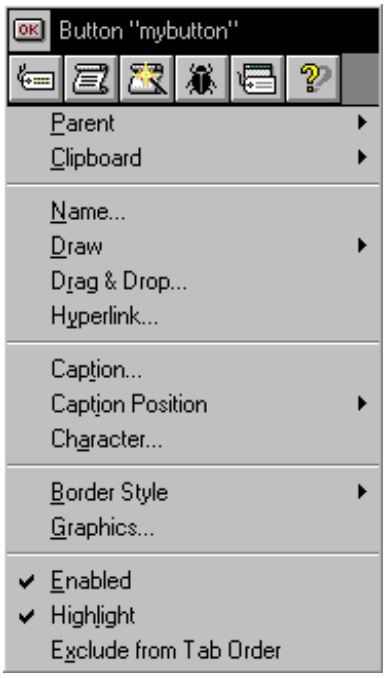
```
helpMessage of button "mybutton" = "Click here to go to the map"
```

This example creates a brand new property, called "helpMessage" for the button and stores the text "Click here to go to the map" in it. The new property would then be visible and editable in the property browser, as shown below, and accessible from any ToolBook script. Also, user properties are persistent so that when the book is saved, so are the properties.



What this means is entirely up to the way it is interpreted in the rest of the scripts. However, it is a poorly covered but extremely powerful feature of the ToolBook object model and developers should make every effort to learn how to use user properties - they can dramatically reduce the amount of scripting required.

There are several other ways of setting properties of objects apart from those shown above. The most useful of these is the right click menu which appears when you right click on any object with the mouse. This menu is context sensitive and is usually the quickest way of setting any property. It also has the advantage of being accessible from reader mode so that properties (including scripts) can be changed while the book is running[7]. An example right click menu, for the same button as in the previous example, is shown below.

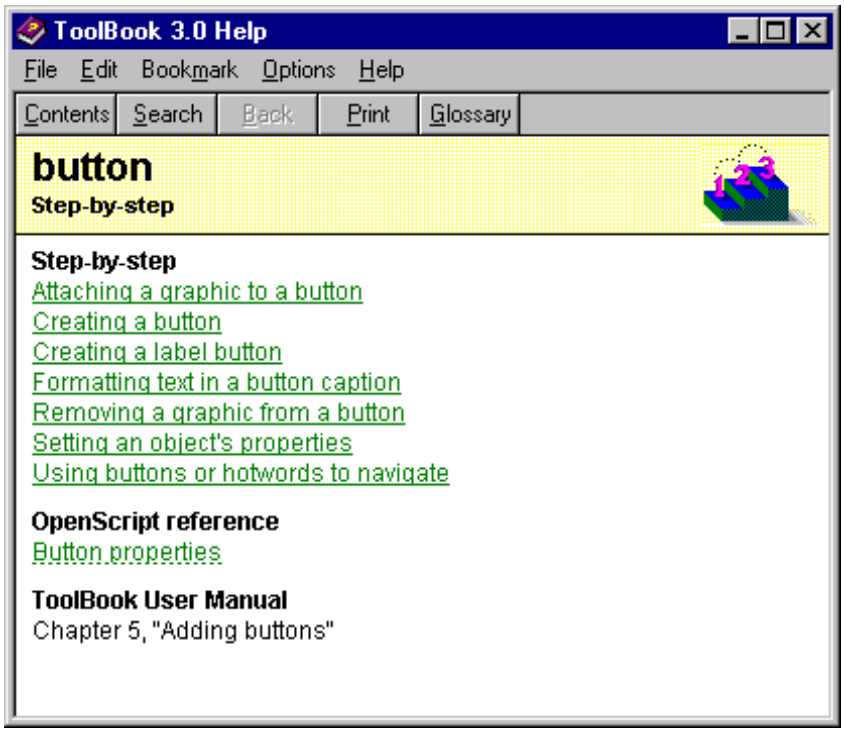


The row of graphics buttons along the top of this menu provide quick context sensitive access to the following six features.

- Property dialog box
- Script editor
- Autoscript selector
- Debugger
- Property editor (or a developer definable action)
- Open on-line help for this object

For beginners (and experienced ToolBook programmers) the last of these options is particularly useful as it opens up the on-line help with a task related list of activities possible with this object type. For example, the entry for a button is as follows.

[7] The line "startupRightReaderClick = true" must be added to the TOOLBOOK.INI file in the Windows directory for this to be enabled.



ToolBook 3.0 (and 4.0) is a 16-bit, rather than a full 32-bit, Windows application and as such suffers from a number of memory related constraints. The following table, reproduced from the Asymetrix product information, lists the limits to which various ToolBook objects are constrained. Further information and comments added by the authors of this report are included in italics.

Object Type	Memory Constraints
Book	Maximum number of pages: 65,536 (64K). This maximum is reduced when the book contains viewers, scripts with more than 1,024 bytes, bitmap resources, menu bars, colour palettes, or embedded OLE objects. In practical terms, a book is limited to about 50,000 pages. *The number of pages in a book has little effect upon the time taken to open it provided that no scripts are added to scan through all pages initialising them as the book is opened. ToolBook only pre-loads the first page off disk; others are loaded on visiting them.* *Page names are limited to 32 characters.* *Page name look up is a straight serial search so name look up time can be large with large numbers of pages.*
Background, Page	Maximum number of objects on background or page: varies. Each background or page can occupy up to 65,536 bytes (64K) of memory. Each object on the background or page, including pages, requires a fixed amount of memory. (For example, each button requires about 90 bytes.) Additional memory is occupied by text (in fields, record fields, and combo boxes), scripts, and user properties. The maximum number of objects therefore depends on which objects are placed on the background and how much extra memory they require. As an example, you could put 1,100 lines on a background if none of them had scripts or user properties. However, you could only place two fields on the background if each contained 32,000 characters of text. *Small graphics (of the size of a typical button or less) can rapidly consume the page memory as, unlike large graphics, they are stored in the 64K page memory area. Keeping small graphics as bitmap resources rather than importing them as pictures/graphics avoids this problem.* *Careful design of screens can avoid the object count becoming a problem. For example, drawing graph curves as a single angledLine object rather than multiple separate line objects avoids memory problems with graphs.* To optimise the way a background or page uses memory, save the book with a different name with the save as command, which organises the memory used by a background or page to its optimal configuration.

	Public domain scripts available from the user group web site automate this so that it occurs automatically on every save.
Field, Record Field	Maximum number of characters: 32,000. If the text is formatted, or if there are other objects on the background or page, you might not be able to put the maximum amount of text into the field or record field. *ToolBook can become unstable and unpredictable when manipulating text in fields if the size of the text string approaches 32,000. It is a good idea to keep well below that limit in practice.*
Variable	Maximum number of characters: 65,473. Maximum data in all variables combined: 16MB. *User properties, although not strictly variables, can hold up 16MB each and are a way of getting past the 64K data per variable barrier. However, user properties are not as fast to access as variables (although this will only show in speed critical loops).*
Handlers	Maximum levels of nesting: 1.024 (1K), but this limit is reduced depending on the complexity of the script. The practical limit is between 50 and 300. *In practice this limit is rarely a problem and is only encountered as a result of accidentally re-entrant handlers.*

4.3. OpenScript

The OpenScript programming language has already been mentioned several times before in this report. It is in fact a proprietary programming language and far from open in the open standards sense. Every aspect of the language is under Asymetrix's direct control and they exercise their reserved right to make changes from version to version of ToolBook. Fortunately, backward compatibility has been mostly maintained and where newer versions do differ a script upgrade program, known as Script Walker, points out incompatibilities and suggests amendments. OpenScript in version 4.0 turned out to be almost entirely backward compatible with 3.0 unlike the upgrade from 1.5 to 3.0 where changes were frequently required.

4.3.1. Programming Paradigm

OpenScript is a semi-object orientated language with message passing and containment inheritance. Scripts may be attached to any object on the object hierarchy. When events occur, such as a mouse button click on a graphic, then ToolBook sends a message (buttonClick in this case) to the target object - the graphic. If there is no script on the target object then the message automatically gets forwarded up to the parent of the object (e.g. the group or the page containing the graphic) and the same occurs for the parent (i.e. if there is no script the message

automatically gets forwarded up to the parent of the object). However, if there is a script to handle the buttonClick message on the object, the script will run and it is then up to the script as to whether the message gets forwarded to the parent - the default is to not forward it and so an explicit "forward" command is required in the script to make the message continue its journey.

This scheme allows generalised behaviour to be factored out of individual object scripts and up into the script of parent objects, or even their parents. For example, placing a buttonClick handler in a book's script will provide a general behaviour when any object is clicked (provided objects do not intercept the message lower down the hierarchy and not forward it up the tree). In this way, generalisations can be scripted onto book, background and page scripts with exceptional cases scripted on individual or groups of objects. This is a remarkably powerful programming paradigm but there are problems. Firstly, it takes quite a while for new ToolBook programmers to really become comfortable with message forwarding and secondly, it is all too easy to forget to forward a message in which case the script handlers higher up the tree never even run.

There is also another down side to this "intuitively best" approach to ToolBook script structuring which should be considered by developers. Functionality distributed across multiple levels of the script hierarchy is extremely difficult to disentangle and isolate when trying to re-use scripts in another application or share scripts with other developers. To make scripts more re-usable they need to be self contained; which means that they are not split over several objects at different levels. Instead, re-usable scripts should be self contained and perhaps attached to a single object or group of objects. Of course, there is a down side to this approach too and that is the introduction of duplicate scripts and increased difficulty in maintaining copies of the same script duplicated on, possibly, hundreds of objects. Clearly there is no right and wrong approach to positioning scripts and developers must make a value judgement in the light of each project.

4.3.2. OpenScript Features

The range of commands, functions and properties supported in OpenScript can be seen in the Asymetrix product information reproduced and annotated in the appendices of this report. However, to give the reader a feel for the language and the way it works a simple example set of scripts are given below. Note that it will not easily be possible to understand these scripts if you have never written computer programs before - OpenScript is just as difficult to understand as Visual Basic or Delphi.

Example 1: Script on a button to show or hide a pop up consisting of a group (e.g. a help pop up).

```
to handle buttonClick
	if visible of group "mypopup" then
		hide group "mypopup"
	else
		show group "mypopup"
	end if
end buttonClick
```

Example 2: Script on a group object which consists of several buttons named "chapter 1" through to "chapter 7" to navigate to pages of the same name as the button when the user clicks on any of the buttons (e.g. clicking on button "chapter 2" will go to page "chapter 2"). Notice the technique of placing a single script on the group rather than a slightly different script on each of the buttons.

```
to handle buttonClick
	local dest
	dest = name of target	-- target will be the button clicked
	go to page dest
end buttonClick
```

Example 3: Script on a button to show or hide a pop up group but to also pass the buttonClick message up to the page script where a generalised action for all buttons is taken.

```
to handle buttonClick
	-- pass the message up to page script by a "forward"...
	forward
	-- ...and return here after page's buttonClick handler has run
	if visible of group "mypopup" then
		hide group "mypopup"
	else
		show group "mypopup"
	end if
end buttonClick
```

The script on the page for this example might then check to see if the object clicked (i.e. the target) was a button and if it was, make a beep sound. This effectively makes all buttons on this page beep when pressed (provided they included a forward in their scripts or if they had no script at all).

```
to handle buttonClick
	if object of target is button then
		beep 1
	end if
end buttonClick
```

If this script were moved up to the book script, rather than being in the page script, then every button in the book would make a beep sound when clicked.

Example 4: A complicated script fragment taken from the book script of a large ToolBook application (Note that OpenScript does not have to get this complicated, but it can). This script does not handle a system message like buttonClick, instead it is a custom handler to make an object move[8] smoothly from where it is currently is to a specified position at a specified speed. If you are familiar with other programming languages you will see several features common to many languages such as parameter passing, global variables (known as system variable), local variables, typed and untyped variables, loop constructs and so on. Interestingly, an assembly language like accumulator called "it" is the default return location of functions of the for "get foo( )" and the pop command (also similar to assembly language pop/push stack instructions). However, the OpenScript versions are far more flexible and "it" is persistent across subroutine calls (i.e. send, get and forward)

```
to handle we_move obj,x,y,spd    -- spd is null,slow,medium,fast
       system xonepixel,yonepixel-- global variables
       local stack lis,olist     -- local variables
       local object o
       local long dx,dy,nsteps,remx,remy,i
       get position of obj
       dx = x - item 1 of it
       dy = y - item 2 of it
       lis = obj
       get p_style of obj
       conditions
              when it = null
                     break conditions
              when it = "s_popup"
                     put p_objlist of obj into olist
                     while olist<>null
                            pop olist into o
                            pop olist                 --style
                            get p_child of o
                            if it<>null
                                   push it onto lis
                            end
                     end
              when it = "s_backdrop"
                     get p_child of obj
                     if it <> null
                            push it onto lis
                     end
              when it = "s_spreadsheet"
                     if p_handle of obj <> null
                            get position of obj
                            get we_spread_position(p_handle of obj, \
```

[8] Note that Multimedia ToolBook includes a path animation tool to do a similar action but that the specific application this script was taken from was written in non-multimedia ToolBook 3.0 and also had a requirement to automatically move associated objects along with the specified object.

```
                                    item 1 of it+dx,item 2 of it+dx)
                        end
                else
                        get p_child of obj
                        if it <> null
                                put ","&it after lis
                        end
        end conditions
        remx = dx
        remy = dy
        if spd <> null
                conditions
                        when spd = "slow"
                                nsteps = 20
                        when spd = "medium"
                                nsteps = 10
                        when spd = "fast"
                                nsteps = 4
                        else
                                nsteps = 1
                end conditions
                dx = dx / nsteps
                dy = dy / nsteps
                step i from 1 to nsteps
                        sysLockScreen = true
                        move lis by dx,dy
                        decrement remx by dx
                        decrement remy by dy
                        sysLockScreen = false
                        if we_sync(1) > 0
                                break step
                        end
                end
        end
        sysLockScreen = true
        move lis by remx,remy
        sysLockScreen = false
end
```

An example of how this script might be used is given below. This sample "call" of we_move is the script on a graphic which, when clicked, zooms to position 300,500.

```
to handle buttonClick
        send we_move self, 300,500, "fast"
end buttonClick
```

As with all interpreted languages, execution speed is often an issue. The degree to which this is a problem depends upon the application type but typical graphically intensive multimedia

applications tend to be dominated by calls to the Windows display routines rather than running numerically complex iterative calculations. Therefore, the speed of the application can be more dependant upon the speed of the graphics card, the hard disk and the amount of free RAM. With this caveat in mind, the remainder of this section examines the execution speed of various activities in OpenScript.

4.3.3. Speed and Efficiency

ToolBook book execution speed tends to be dominated by time taken to fetch graphics from disk and to refresh the screen and this is largely a function of hardware and system software. However, OpenScript execution speed can also be important in some cases. Where this is the case, the way in which the OpenScript is written can have a profound effect on execution time - making orders of magnitude difference. Knowing how to write efficient OpenScript requires general programming experience and a detailed knowledge of how ToolBook works.

The following series of example scripts illustrate the difference between optimised and unoptimised code to do the same task. Times are given for each version (for a 90MHz Pentium although the ratio will be roughly the same for other machines). Each of these examples makes use of the timing handlers called "startTimer" and "stopTimer"; the scripts for these are listed after the last of the examples.

Caching references to objects
Example 1a - unoptimised

```
to handle buttonClick
        local   long    i

        send startTimer
        step i from 1 to 2000
                get text of field "name"
        end
        send stopTimer
end
```

Time: 0.48 seconds

Example 1b - optimised

```
to handle buttonClick
        local   field   tField
        local   long    i

        send startTimer
        tField = field "name"
        step i from 1 to 2000
                get text of tField
        end
        send stopTimer
end
```

Time: 0.16 seconds

Caching references to properties
Example 2a - unoptimised

```
to handle buttonClick
        local   long    i

        send startTimer
        step i from 1 to 500
                increment payAmount of this page
        end
        send stopTimer
end
```

Time: 0.76 seconds

Example 2b - optimised

```
to handle buttonClick
        local   long    i, temp

        send startTimer
        temp = payAmount of this page
        step i from 1 to 500
                increment temp
        end
        payAmount of this page = temp
        send stopTimer
end
```

Time: 0.01 seconds

Setting group properties
Example 3a - unoptimised

```
to handle buttonClick
        local   long    i

        send startTimer
        step i from 1 to 52
                text of field i of group "example" = NULL
        end
        send stopTimer
        text of group "example" = "$0.00"
end
```

Time: 0.27 seconds

Example 3b - optimised

```
to handle buttonClick
        send startTimer
        text of group "example" = NULL
        send stopTimer
        text of group "example" = "$0.00"
end
```

Time: 0.07 seconds

Managing Lists
Example 4a - unoptimised

```
to handle buttonClick
        local   stack   tList
        local   long    i
        local   currentItem

        send startTimer
        tList = \
"1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,
10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,
9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,
8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,
7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,
6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,
5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,
4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,
3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,
2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,
1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,1
0,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9
,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8
,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7
,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6
,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5
,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4
,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3
,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2
,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1
,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10
,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,
10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,
9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,
8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,
7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,
```

```
6,7,8,9,10"

        step i from 1 to itemCount(tList)
                currentItem = item i of tList
        end
        send stopTimer
end
```

Time: 0.56 seconds

Example 4b - optimised

```
to handle buttonClick
        local   stack   tList
        local   long    i

        set tList to \
"1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,
10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,
9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,
8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,
7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,
6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,
5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,
4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,
3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,
2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,
1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,1
0,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9
,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8
,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7
,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6
,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5
,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4
,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3
,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2
,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1
,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10
,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,
10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,
9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,
8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,
7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,
6,7,8,9,10"

        send startTimer
        step i from 1 to itemCount(tList)
                pop tList into currentItem
        end
        send stopTimer
```

```
end
```

Time: 0.02 seconds

Timing Handlers

These handlers were located on the book script and used to time the above examples.

```
to handle enterApplication
        linkDLL "USER"
                DWORD   GetTickCount()
        end
end

to handle startTimer
        system s_testTime
        set s_testTime to GetTickCount()
end

to handle stopTimer
        system s_testTime
        thisTestTime to (getTickCount() - s_testTime)
        set displayTime to thisTestTime/1000
        format number displayTime as "0.00"
        request "This test took" && displayTime && "seconds."
end
```

4.4. Communicating with other Applications

Particularly in Computer Based Training applications of ToolBook it is useful to be able to communicate with other non-ToolBook applications. ToolBook is able to communicate with other applications using the following methods.

- Simple application launching
- Message sending and posting via Windows API
- Windows Dynamic Data Exchange (DDE)
- Object Linking and Embedding (OLE - but not OLE 2 or ActiveX)
- Direct access to other ToolBook books from OpenScript

A brief explanation of each of these is given below along with comments on the practicalities of using each approach.

4.4.1. Simple application launching

Starting up other applications from ToolBook is straight forward assuming that the location of the application on the user's machine is known. A single line of OpenScript will achieve this.

```
run "c:\windows\notepad.exe"
```

Unfortunately, having launched an application no information on that application is passed back to OpenScript and so programmers are forced to write OpenScript code to search through the Windows list of windows in order to get a "window handle" which will allow ToolBook to manipulate that window (e.g. positioning it on screen or closing the application). This is a fairly low level operation which requires exactly the same degree of knowledge of Windows programming as would be required to do the same task in C/C++.

4.4.2. Message sending and posting via Windows API

ToolBook allows Dynamic Link Libraries to be linked into OpenScript to extend its functionality (see 4.5, "Extending ToolBook"). Importantly, this gives access to all the built in Windows DLLs one would normally access in C/C++ to send and post messages to applications.

This allows experienced Windows C programmers to communicate in a number of ways with other applications. The simplest of these would be to, for instance, close another application by sending the appropriate WM_ message. However, unless originally designed to communicate in this way, the level of communication with applications using this technique usually stops short of exchanging data or running tasks; communication is more at the level of the Window rather than the application.

ToolBook comes with redistributed Microsoft SDK API help files which list commands available through the built in Windows DLLs and the numeric constants corresponding to each of the WM_ and other Windows constants.

4.4.3. Windows Dynamic Data Exchange (DDE)

DDE is the Windows protocol for exchanging data between applications. Most mainstream Windows applications respond to DDE to a greater or lesser extent. Microsoft Excel for

instance allows data to be transferred in and out of a spreadsheet using DDE; it is also possible to open, manipulate and close Excel spreadsheets remotely via DDE. Similar actions are possible in Word, Access, Netscape, Explorer and numerous other applications - including ToolBook.

It is therefore possible to control other applications from ToolBook and to transfer data between that application and ToolBook. The nature of the conversation between the programs can be client server or peer to peer (in the control sense rather than the networking sense).

While DDE is impressively easy to use in OpenScript (despite having non-standard names for the messages/commands compared to those used in Windows programming in general). For some applications of ToolBook a simple DDE conversation may be all that is required and that will work perfectly adequately. However, applications such as Access can be unusably slow in transferring data back to ToolBook. It is hard to tell where the blame lies but the net effect is that a DDE link between the two can be unbelievably sluggish. For example, a multiple choice question system which used Access to store the questions and ToolBook to display them on screen took up to 2 minutes to DDE transfer 15 short strings and an integer. The same information retrieved and displayed in Access itself appeared almost instantly.

The use of DDE should not be ruled out on these grounds but a commitment to use this channel of communication should not be made with undertaking a pilot study of the intended link in use.

4.4.4. Object Linking and Embedding (OLE - but not OLE 2 or ActiveX)

It is possible to embed OLE objects, such as an Excel graph, in a ToolBook page. However, this approach to linking applications is seriously flawed if the end-user does not have the same version of the embedded software on their machine or if it is not in the same directory. Certainly this is not viable for ToolBook books produced for general circulation.

An OLE object has a set of properties associated with it just like any other ToolBook object and these can be manipulated in OpenScript.

4.4.5. Direct access to other ToolBook books from OpenScript

Unlike DDE links, directly accessing any objects or data in another ToolBook book does not require that the other book is open and requires no special protocol. For instance, it is not widely known that a second ToolBook book set to read/write access may be used as a simple but effective log file (or to record user preferences or similar persistent data) which may be read and written by multiple users in the same way that an ANSI text file would be. From a book which is read-only and opened simultaneously by multiple users, a separate read/write book can be opened, written to and closed for each and every log operation. ToolBook itself will take care of any back-off and retry upon collision of more than one user trying to access the log book simultaneously. The ease[9] of this operation can be seen from the shortness of the following code fragment which writes a user's name and score to a text box (i.e. field) in a ToolBook book called "scorelog.tbk".

[9] Do not worry about the detailed meaning of the code unless you have looked at OpenScript programming in detail before; it is merely included to show that volumes of programming are not required simply to log information to another ToolBook book.

```
put username & SPACE & userscore & CRLF after text of \
      field "scores" of page 1 of book "scorelog.tbk"
save changes to book "scorelog.tbk"
```

Note that the book is not actually opened in the sense that no one has opened it explicitly anywhere on the network; however, it is opened implicitly when data is written to it as in the above "put" command. To see the logged results, the instructor could later open the log book (scorelog.tbk) and scroll through the text in the field called "scores" on the first page of the book.

4.5. Extending ToolBook

There are two ways of extending the functionality of ToolBook and OpenScript.

- System Books
- Dynamic Link Libraries (DLLs)

In fact Asymetrix have used both these methods to extend the core ToolBook engine and much of what developers perceive as ToolBook is in fact a system book written in ToolBook or a supplementary DLL.

4.5.1. System Books

System books have been briefly mentioned before in this report. They extend the inheritance hierarchy, which normally stops at the book, up another level. So, after a message has risen up to the book script it then passes on up to the first system book (if there is one) and then up to the second system book (if there is one) and so on. By placing the sort of scripts one might normally expect to place in a book script into a system book, these scripts can then be used by several applications. Any changes and bug fixes to scripts in the system book will therefore automatically apply to all applications using that script. Two common uses of system books are to construct libraries of re-usable scripts and to extend the functionality of ToolBook by adding extra menu items or authoring features.

A system book is actually only a normal ToolBook book and any book can be nominated as a system book simply by adding it to the sysBooks list in an application (normally as the application starts up). Adding a book to the sysBooks list places its book script in the inheritance hierarchy but does not open or load the rest of the system book.

By default, ToolBook starts up with either TOOLS30.SBK or MTB30.SBK. These system books add the Tools menu to the author level menu bar and are a good example of what is possible with system books. It is possible to examine and modify the scripts of these books as all the OpenScript code is there.

4.5.2. Dynamic Link Libraries (DLLs)

One of the most flexible ways of extending OpenScript is through the use of Windows DLLs. A number of these are always available on any Windows PC. Several are also supplied with ToolBook and it is possible to develop your own in C/C++, Delphi, Turbo Pascal and any other language capable of generating Windows 16-bit DLLs. However, it should be noted that writing DLLs to extend ToolBook is time consuming and requires a good understanding of the way

both Windows and ToolBook work; it is not an activity which can easily be picked up by a non-programmer without much hard work.

Assuming that a suitable DLL already exists then it may be used in OpenScript by using the linkDLL command to add routines from the DLL as extra commands to ToolBook. The linked command can then be used just like a built in OpenScript command. Before the application closes it must unlink all linked DLLs or Windows will not release them from memory.

An example of linking to one of the DLLs shipped with ToolBook is given below. This example button script opens a standard Windows file open dialog box (with all the directory changing, drive selecting, filtering normally found in Windows open dialogs). It does this by linking in a function from TB30DLG.DLL, using it and then unlinking the DLL again straight away (an alternative would have been to have linked the DLL at the start of the application and to only close it at the end of the application - developers are free to choose). The file name returned is then used to open the file and read it into a text field.

```
to handle buttonClick
	local fname

	linkDLL "TB30DLG.DLL"
		STRING openDlg(STRING,STRING,STRING,STRING)
	end
	fname = openDlg(".","*.txt","Choose a text file...","Open")
	unlinkDLL "TB30DLG.DLL"

	if fname is null
		break  -- was an error so quit handler
	end if
	openFile fname
	readFile fname until EOF
	text of field "myfield" = it
	closeFile fname
end buttonClick
```

4.6. Graphics

This section describes graphics features of ToolBook which are common to both ToolBook 3.0 and Multimedia ToolBook 3.0. Video and other graphics features exclusive to Multimedia ToolBook are described in the next chapter.

4.6.1. Built in Primitives

ToolBook has a number of built in primitive drawing elements which occupy relatively little memory (compared to bitmap graphics), are quick to display and easy to manipulate. The supported types of primitive are listed below.

- line
- angled line
- curve
- rectangle
- rounded rectangle
- ellipse
- regular polygon
- irregular polygon
- pie

4.6.2. Imported Vector Graphics

ToolBook allows vector image file formats such as WMF to be imported into the current page where they become a ToolBook object type known as "picture". By default, these pictures always appear with a border around them but this can be removed by selecting a different line style from the Lines palette. The picture may be made transparent, resized, repositioned, shown/hidden and have a script assigned to it.

On slower machines or with complex pictures picture files pictures take a long while to draw and can be unusably slow. It is important to test the redraw speed on a representative delivery machine.

Large and complex files, particularly EPS files it seems, fail to import correctly.

4.6.3. Imported Bitmap Graphics

Bitmap graphics for all popular file formats can be imported in the same way as vector file formats. ToolBook uses conventional Windows graphic import filters and so as graphics packages installed on the developer's machine add extra formats, these become available to ToolBook. Once a graphic is imported into ToolBook the filter is no longer required so end-users do not need to have software to support these extra formats; the image is stored in ToolBook's own internal format.

An imported bitmap usually becomes a ToolBook paintObject. A paintObject can be manipulated just like a picture object except that it can not be resized - instead it is clipped at the right and bottom edges. However, some formats such as PCX are imported as picture objects and can be resized just like a vector image file format.

4.6.4. Bitmap Resources

A feature new to ToolBook 3.0 was the addition of bitmap resources (not compatible with standard Windows bitmap resources in the way they are stored in the ToolBook file). By

importing a bitmap as a resource it can be used throughout the book. Each instance of the image shares the same bitmap rather than each one duplicating the same bitmap as would occur if they were imported directly into the page.

Bitmap resources can only be used on buttons or in text fields and so it is common practice to use a borderless button to display a bitmap resource as a (seemingly) standalone object.

4.6.5. Storage Options

Small images, about the size of a typical Windows button, if imported as paintObjects consume a relatively large amount of the scarce 64K memory available for a page's objects and are best stored as bitmap resources. Larger paintObjects are stored outside the 64K and so, ironically, are less of a problem.

If bitmaps are to be used more than once in a book they should be stored as bitmap resources rather than paintObjects.

An option exists for complex pages and backgrounds to be converted to a single image to speed up page redraw. However, in practical terms this appears to make little difference and commits to a specific screen format for delivery.

4.6.6. Palettes

256 colour screen displays use palettes, a selection of 256 from the range of colours possible, because only 256 colours can be displayed simultaneously on such hardware display modes. The precise choice of which 256 colours to display is a complex process when different bitmaps may each require a different set of colours. This can result in image degradation and a flashing of incorrect colours when ToolBook pages are displayed. The issue is further complicated by different priorities for bitmaps at different screen depths and between foreground and background images.

There is no simple fix for this problem which will work in all cases. The simplest fix is to remap or originate all the bitmaps so that they share the same palette - even using sub-256 colour images does not entirely avoid the problem. ToolBook allows a single palette to be specified for the entire book and this prevents any palette flash at the cost of restricting the colour range to a fixed set of 256 colours.

5. Differences between ToolBook 3 and Multimedia ToolBook 3

There are two different versions of ToolBook 3.0; these are ToolBook 3.0 and Multimedia ToolBook 3.0. Multimedia ToolBook 3.0 is a superset of ToolBook 3.0. As well as multimedia features it adds a number of extra features to the basic ToolBook engine.

Multimedia ToolBook 3.0 has higher system requirements (see chapter 2, "System Requirements") and a much higher price (see "Appendix A - Purchasing") than the ordinary non-multimedia version.

The additional features of the multimedia version are listed and briefly described below. Then, in chapter 6, "Versions and Upgrades", the restrictions in upgrading from ToolBook to Multimedia ToolBook are discussed along with the problem of being unable to step back in the opposite direction.

5.1. Sound

Unlike graphical images, which are stored within the ToolBook book, sound files are stored in external files and merely referenced from within Multimedia ToolBook 3.0. Small sound files can be played directly using the playSound command. However, the most flexible way to refer to and play sounds is to define them as ToolBook clips. Clips provide an easy way to isolate physical file locations from the OpenScript coding which plays the sound files and at the same time allow developers to easily specify sub-ranges of the media file to play (with one or two options). A clip is essentially an internal reference to a sound file (or a video file). Clips are defined, edited, renamed or deleted using the Clip Manager in conjunction with property dialogs (alternatively, these operations may be performed in OpenScript).

A small set of commands, the "mm" commands allow sound clips to be opened, played, paused, stopped, rewound and closed. The usage of these commands is the same for both sound and video clips.

There is support for all popular MCI sound formats - including:

- WAV
- MIDI

Volume control and sound mixing are easy to implement and samples of these functions are included in the New Features demonstration book in the samples directory. Autoscript code is also provided to play sounds and allow sound synchronisation to Path Animations.

5.2. Video

The same clip mechanism as used in playing sounds can be used to play video clips. Also, the way in which the "mm" commands are used is exactly the same. However, video clips have a much wider choice of options than sound clips. In addition to specifying start and stop positions, effects and scaling can also be specified. This can be done using either OpenScript or the properties dialog box for the clip.

Multimedia ToolBook 3.0 includes support for all popular MCI formats, including:

- AVI
- Director Movies
- MPEG
- QuickTime

The support for Video for Windows includes its runtime on the Multimedia ToolBook 3.0 CD.

The basic method for positioning and playing a video clip is through stage objects. A "stage" rectangle can be drawn on the page and then, still or multimedia clips can be displayed in this stage. A stage can display any MCI and non-MCI multimedia in a single window and automates the process of positioning and sizing visual media without the need for scripting. A few Autoscript scripts are included to support the use of stages but, in practice, a fair degree of OpenScript expertise is required to produce a finished, usable product by this method.

An alternative to manipulating a stage object directly is to use one of five multimedia widgets which allow the inclusion of video and controls to play it without any OpenScript programming by the developer. Each of the widgets consists of a stage and, usually, a number of control buttons to play, stop, pause, etc. Some also have a slider to indicate progress through the video clip as it is playing and to allow the user to use it to move through the clip quickly. These widgets are very easy to use and all parameters may be changed at any time by invoking the property editor dialog for the widget. (Note: there is a bug which makes Path Animations crash when there is a multimedia widget on the same screen).

By and large, video support is excellent under ToolBook. Unfortunately, the one major failing is that the video is played in a borderless window floating on top of the ToolBook window; video is not properly integrated into the ToolBook imaging engine. This means that screen locking such as sysLockScreen does not have any effect on video, making it difficult to cue a video clip before entering a page and have it appear on the first frame without a nasty flicker or flash. Also, it is not straight forward to superimpose ToolBook objects, such as lines or text, on top of a video clip. To do this requires creating a transparent window (a viewer) over the clip and then displaying any superimposed objects in that window rather than the main one; this is not too difficult, but is error prone and lacks the elegance of having video behave just like a bitmap or a rectangle or a text field.

5.3. Digital Video Producer

Digital Video Producer is a program Asymetrix acquired the rights to and then bundled with Multimedia ToolBook 3.0. It allows the recording and editing video and audio files in a drag and drop fashion. This is a powerful program in its own right and a very useful addition to the overall package.

5.4. Other

- Text search and indexing. Allows the semi-automatic creation of a full hyperlinked index to a ToolBook application.
- Chromokey colours. Allow a transparent colour to be specified for any bitmap so that "blue screen" techniques can be used to superimpose or combine graphics without ending up with rectangular edges.
- A few more widgets in the sample widgets book (e.g. simple graphing).
- Kodak Photo CD Support, plus all other popular non-MCI formats (eg. BMP, DIB, PCX, WMF, GIF, PICT) so that still images from external files can be displayed in a stage.
- Embedded TrueType Fonts - so target machines do not need equation fonts etc. installed. However, this does not resolve copyright problems of redistributing third party fonts. Many fonts self protect against inclusion in this way.
- Path Animation Tool - draw lines and curves for animated objects to follow. Requires no scripting but does not allow simultaneous, synchronised definition of multiple paths or multiple objects; they

all must be defined separately and then set off playing one after the other. This severely limits the utility of this otherwise useful tool.

- Media Packager - automatically checks all media links, gathers and assembles all multimedia files for a given application, compresses the files, and packages the application in preparation for distribution through the bundled Set-up Utility (which is supplied with TB3 and MMTB3).
- Spelling Checker (Although why this should be classed as a multimedia extension is beyond understanding!)

6. Versions and Upgrades

There are several versions of ToolBook. Developers need to ensure they have the most suitable version.

6.1. Upgrading from ToolBook 1.5

ToolBook 3 is NOT binary compatible with ToolBook 1.5. To upgrade an existing ToolBook book requires you to run an upgrade utility included with ToolBook 3. The resultant new book can not then be loaded back into ToolBook 1.5 and so if a parallel development approach to upgrading is planned, it will be necessary to maintain both 1.5 and 3.0 versions of your books. Within the Economics Consortium we judged parallel development impractical to configuration management with such a large volume of courseware being developed by a geographically distributed team of programmers. We therefore approached upgrading as a permanent commitment to the new version and as an irreversible step. A high degree of certainty about the feasibility of upgrading existing code was required because of this.

Experimentation with upgrading a variety of non-TLTP ToolBook 1.5 books has shown that in most cases the upgrade is fairly straight forward and often requires no manual editing to make the new version work. However, books with advanced OpenScript techniques - particularly those involving DLLs - often report a successful upgrade only to then cause a Windows General Protection fault when run. In the case of the Economics Consortium, this has made it impossible to upgrade a single module to-date. This is due to the fact that all of our courseware is built around a template which employs several advanced Windows and ToolBook programming techniques. All the problems are associated with making our custom tools DLLs written in C to work under the new system (largely due to a change in the meaning of "" and NULL parameters). Fortunately, none of the problems which have arisen in the template have taken very long to solve and it is reasonable to assume that once the template is working, the courseware modules themselves will take relatively little effort to upgrade as none of them directly employs complex programming techniques - all complex programming is encapsulated in the template.

Once the template is upgraded to run under 3.0, it is then possible to replace a number of our custom tools written in OpenScript with the in-built system ones supported by ToolBook 3 (e.g. graphics buttons, bitmap resources and editing enhancements). The cost of doing this is restricted to the cost of taking advantage of these new features within the template itself; no substantial changes to the existing courseware modules is required to take advantage of these new features. Such an upgrade would be far more extensive in its scope for projects without a core template and would reduce the feasibility of fully embracing the new features during the upgrade process.

6.2. Upgrading from ToolBook 3 to Multimedia ToolBook 3

- It is possible to run ToolBook 3 books under Multimedia ToolBook 3 but the reverse is not true.
- The first time a TB3 book is run under MTB3 there will be a delay while the book is automatically converted. This process is not, officially, reversible.
- Objects can not be cut and pasted between MTB3 and TB3 in either direction.
- An unsupported utility, called MTB2TB.EXE, to convert Multimedia ToolBook 3 books to ToolBook 3 books is available off the Asymetrix FTP site (see "Appendix E - Internet Resources"). Obviously, not every book can be converted and the program comes with a "use at your own risk" warning.

  ftp: //ftp.asymetrix.com/pub/mmtb3x/updates/
  file: MTB2TB.EXE

If you do use MTB2TB it is a good idea to then use MTBXFER (from the same ftp site) to clone a fresh copy of the book after the conversion. This will ensure that the internal data structures are all correct and that no accidentally corrupted objects are introduced.

6.3. Upgrading to Release 3.0a

Older copies of ToolBook will be 3.0 rather than the later and far more stable and slightly enhanced 3.0a.

To upgrade to 3.0a it is necessary to download a patch off the Asymetrix ftp site.

ftp: //ftp.asymetrix.com/pub/tb3x/updates/
file: TB30A.EXE

or, for Multimedia ToolBook

ftp: //ftp.asymetrix.com/pub/mmtb3x/updates/
file: MTB30A.EXE

An extra help file installed when this patch is applied describes differences between 3.0 and 3.0a.

Updates for system books and DLLs are also available in these directories.

6.4. Long-term Upgrade Path

Developers should consider the risks and costs of periodic upgrades before adopting ToolBook. ToolBook, in common with other popular authoring tools, is not an open standard and Asymetrix can make and have made arbitrary changes from version to version. These changes can make it difficult to upgrade legacy code or to switch versions in mid development.

On the plus side, ToolBook is Asymetrix's flagship product and they appear committed to its continued development. Although by no means guaranteed, Asymetrix have stated that they wish to retain high levels of backward compatibility to the previous version in each future release of ToolBook.

6.5. Database Connection

ToolBook Database Connection is an add-on for ToolBook to simplify accessing third party database tables. The feature highlights quoted by Asymetrix are listed below.

Create database structures without any Structured Query Language (SQL) programming.
- Visually link database records with fields on forms.
- Use built-in ODBC drivers for Paradox 3.0/4.0, dBASE.
- III+/IV, Btrieve (5.0 or later), Excel and formatted text.
- Access other databases using database vendor or third-party ODBC-compliant drivers.
- Share data concurrently over a network.

Create sophisticated database forms.

- Display and manipulate data by dragging and dropping built-in data-ready objects, including fields, combo boxes, list boxes, check boxes, grouped buttons, frames, stages and Hotwords™ hypertext links.
- Specify data to retrieve by simply setting properties and picking from lists.
- Navigate, insert, delete, and filter database records without scripting, using the data-ready control panel.

Use rich-text format (RTF) to provide appealing database displays.
- Simultaneously display text and multimedia data.
- Format text for display in data-ready fields using multiple fonts, styles, colours, inline graphics, and superscript and subscript capabilities.
- Create data-ready Hotwords hypertext links to connect data to external applications.

Comprehensive support for multimedia.
- Store references to MCI and non-MCI multimedia data, including wave audio, Musical Instrument Digital Interface (MIDI), .BMP, .DIB, .WMF, .GIF, .PCX, TIFF, Video for Windows, and QuickTime formats.
- Play back multimedia data at the click of a button using the data-ready stage object.
- Display graphics embedded as Binary Large Objects (BLOBs).

New OpenScript functions for faster query and manipulation of data.
- Use over 80 database-specific OpenScript functions to query and manipulate data even faster.
- Execute SQL statements directly from OpenScript.
- Control transaction processing and multi-user access.

In ToolBook 3.0 Database Connection, the product is let down by not supporting grids. This shortcoming has been corrected in the update for 4.0 but the add-on feels like an add-on and lacks the slick database features of a dedicated database product like Access. Certainly, creating a relational database is easier in Access than in this add-on.

The sample applications which accompany the Database Connection are not particularly impressive and the fact that over 80 database specific functions are added to OpenScript somewhat negates the claim on that this tool "requires no programming".

6.6. CBT Edition

Although marketed as a separate product, Multimedia ToolBook 3.0 CBT Edition is actually Multimedia ToolBook 3.0 plus a number of system books and DLLs which extend its capabilities to simplify the development of Computer Based Training (CBT) applications. The major feature of the CBT Edition is its computer based assessment with a light weight course management program to record the student results.

Asymetrix have taken the approach of providing developers with a large catalogue of assessment widgets (e.g. multiple choice, fill-in-the-blank, click on the answer) to be used to hand craft each and every question in a student test. In practice, this is an labour intensive method for developing large scale tests and is, in the opinion of this report's authors, best suited to small scale one-off assessments rather than the high student numbers and large tests which occur in Higher Education (the maximum number of students is 500 and it is not easy to enter lists of students in a single step).

7. Alternatives to ToolBook

The focus of this report is on an in depth evaluation of ToolBook and not on a comparison with alternative systems. However, this section presents a brief, high level comparison of ToolBook against other tools simply to raise the awareness of the reader as to possible alternatives.

7.1. ToolBook's Strengths and Weaknesses

The principle strengths of ToolBook are:

- non-prescriptive application structure;
- powerful and extensible high level programming language;
- fine control of high and low level interface features;
- customisable development environment;
- large user base with active email lists and user groups;
- comparatively low cost of purchase;
- availability of third-party consultancy and training.

The principle weaknesses are:

- when demonstrated or examined briefly authoring seems deceptively easy to use;
- non-trivial development requires programming skills;
- runtime ToolBook required to run application;
- proprietory system subject to changes dictated by manufacturer;
- lack of compatible development support tools (e.g. code analysers, version control, cost estimation);
- Windows PC only.

7.2. Other Authoring Tools

Other currently popular authoring tools are (in alphabetical order):

Authorware	PC, Mac
Director	PC, Mac
Hypercard	Mac
Icon Author	PC, Mac, UNIX

Of these, Hypercard on the Mac bears the most resemblance to ToolBook and it seems likely that, as it pre-dates ToolBook, it was the inspiration for ToolBook's design. Unfortunately direct conversion between the two is not possible despite their similarities; an application must be re-implemented from scratch to convert between the two.

The remaining three systems all have the advantage of being multi-platform whereas ToolBook in its current form is unlikely ever to migrate beyond the PC Windows environment. The precise range of platforms supported and the relative compatibilities between applications authored on one platform and transferred to another varies. Versions and features are changing all the while and so up-to-date information should be sought when making a choice of authoring tool.

Put simply, Director is best suited to applications where timeline based animation, sound and video dominate. Icon Author and Authorware, again simply, are best suited to prescriptive structured applications dominated by flow chartable navigation and requiring limited interaction on each screen.

Of all these authoring tools, ToolBook and Hypercard require the most programming expertise and have the steepest learning curve. In exchange for this, developers using these tools gain flexibility closer to that of general purpose programming tools.

7.3. General Purpose Programming Tools

The authoring tools listed above are not the only possible tools for implementing multimedia and computer based learning software. Another option is to use visual programming tools like Visual C++, Visual Basic and Delphi or even to adopt a hybrid solution like HTML and Java.

In general purpose programming tools everything is possible - eventually. A much greater level of detail has to be specified in these programming systems and this takes time, expertise and a lot of debugging. In return for this developers are able to achieve exactly the result they require. The trade-off in adopting an authoring tools is that the developer has to accept a framework for their application pre-specified by someone else. In some cases, bending an authoring tool to meet requirements can take more effort than implementing the application from scratch in a general purpose programming tool provided the necessary expertise is available.

One advantage general purpose tools tend to have over authoring tools is much better support and integration with configuration management, static and dynamic analysis, cost estimation, measurement and debugging tools. On large scale projects this should be a serious consideration when choosing any authoring tool; the anticipated productivity gains from using an authoring tool should be weighed against potential risks resulting from the lack of these development support tools.

8. Case Studies

This section examines three different types of ToolBook applications through three case studies of completed, real world, projects. These case studies were selected for their different requirements and contrasting implementation techniques as summarised below.

Case Study 1 - ACTA Audit Report
A one off, professionally programmed application specified by a content specialist on paper and through a Multimedia ToolBook prototype.

Case Study 2- Oral Manifestations of HIV
An application with content authoring performed entirely separately from the programming through the construction of a ToolBook template.

Case Study 3 - TLTP Economics' WinEcon
A large scale, multi-programmer, multi-author, multi-site development with collaborative authoring. Also uses ToolBook's expansion capabilities to extend the authoring environment.

For each case study a brief description of the application is followed by an explanation of how the specific requirements led to a particular use of ToolBook. Finally, a hindsight analysis of ToolBook's role in the project is given.

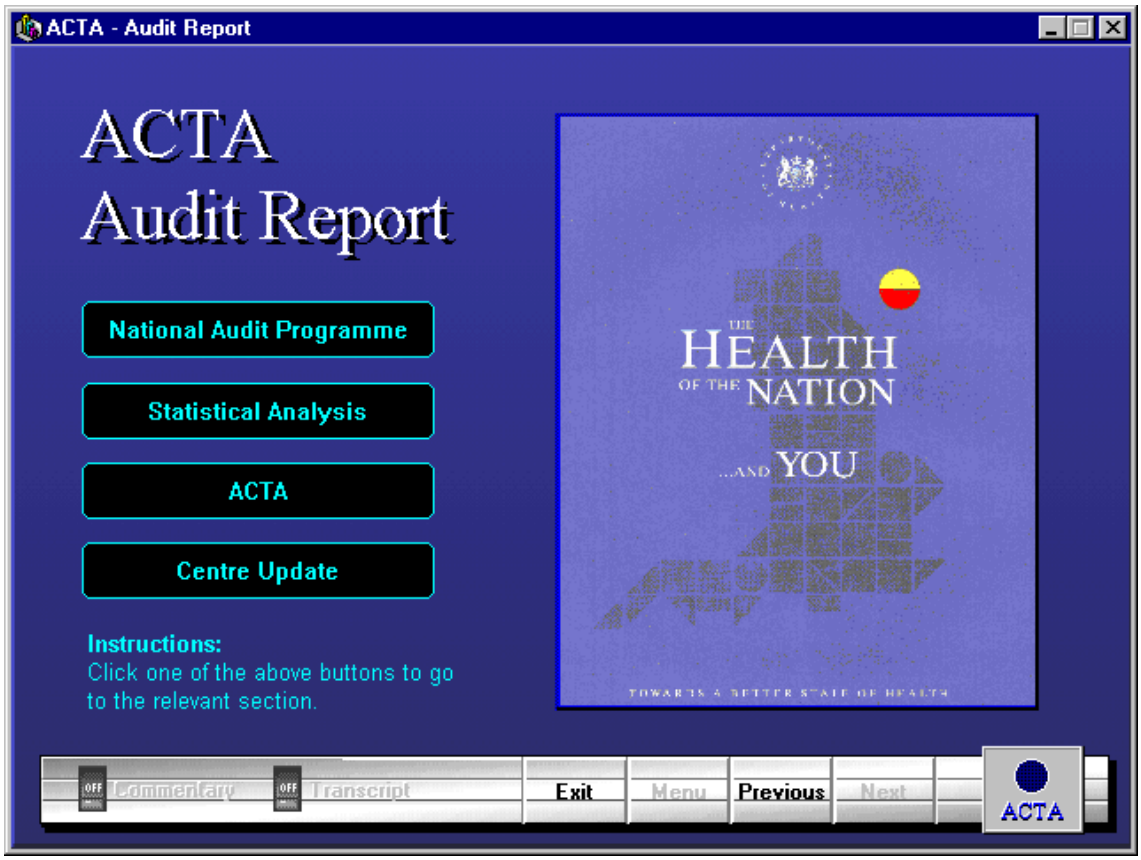
8.1. Case Study 1 - ACTA Audit Report

A one off, professionally programmed application specified by a content specialist on paper and through a Multimedia ToolBook prototype.

In 1992 the Association of Cardiothoracic Anaesthetists of Great Britain & Ireland decided to implement a National Audit Programme. The features of the programme were firstly that information would be collected on computer for electronic retrieval and collation; secondly the audit data would consist of an evolving National Standard Data Set of preoperative, predictive variables and standardised outcome measures. Funding for Department of Health Audit monies allowed all 36 centres in the UK to collect and download the Standard Data Set. The same funding of a statistician and computer hardware and software made possible the detailed analysis of the large numbers that was required. Finally, a link-up with the University of Bristol facilitated the development of a multimedia computer programme for the dissemination of the audit data to the participating centres.

The program application was initially sketched out on paper by the subject specialist Steve Bolsin, a consultant cardiac anaesthetist. As the next step a pilot prototype was produced in ToolBook by a non-programmer, first time user of ToolBook. This pilot had sufficient functionality to demonstrate the feasibility of the concept and to attract funding to complete the development. At this point a ToolBook programmer was contracted to complete the production, taking the prototype as a detailed functional specification and supplementing this with further paper based storyboarding. The consultant's existing PowerPoint slides were used as the style guide for the visual appearance of the software so that the two could be neatly interchanged during presentations.

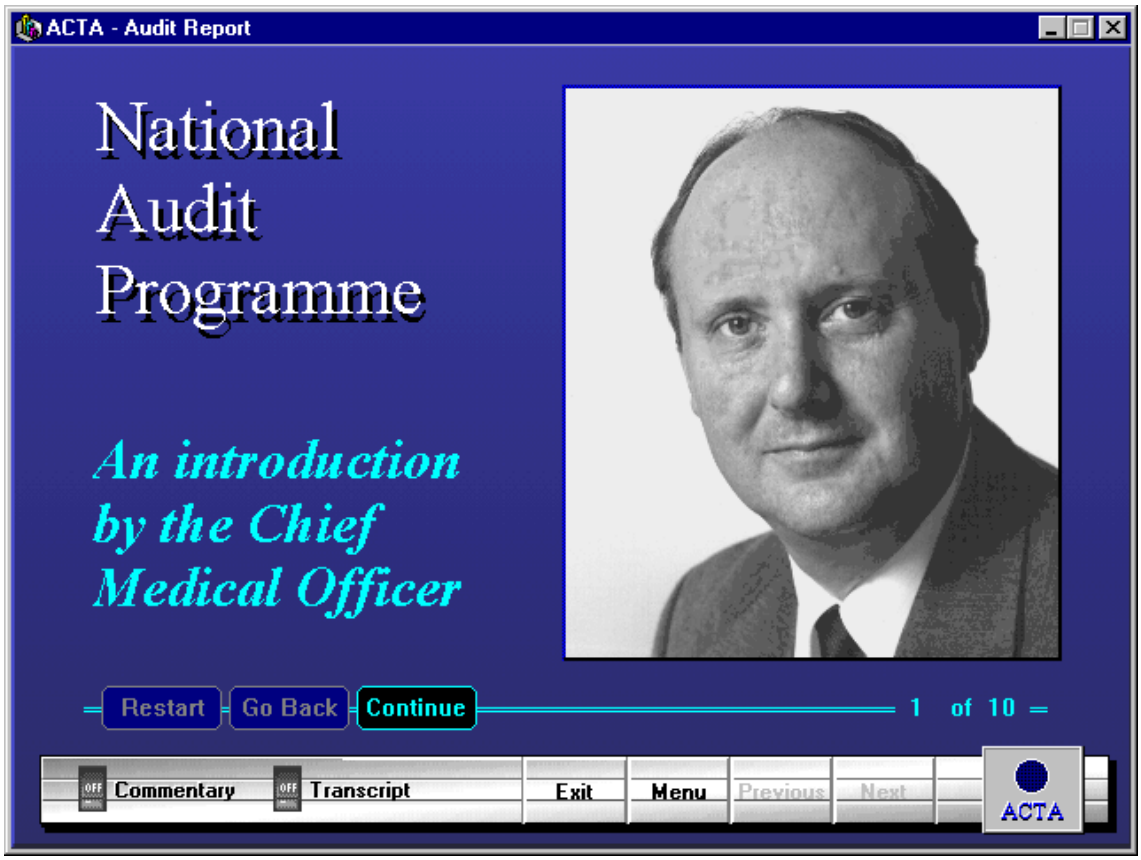
The application consisted of four sections accessible from a menu screen shown below.



The menu screen allowed navigation to the four sections of the application. The first three of these presented a series of slides accompanied by high quality audio commentary from the Chief Medical Officer, a statistician and an anaesthetist. The first two of these slides for the first section are shown on the screen shots on the following two pages.

However, these slide presentations were in fact added value to the main function of the package which was to display Centre performance statistics shipped as encrypted data on a floppy disk to each Centre. While each Centre only received its own data on the disk, this was placed in a national context by also including national statistics on the same disk. To display their data, the Centre ran this ToolBook application, clicked on the Centre Update button, inserted their disk and entered their password. A series of tables, bar charts and pie charts are then displayed comparing the Centre's statistics with the national statistics.

For confidentiality there are no screen shots of the Centre audit report screens.



Users click on the Continue, Go Back and Restart buttons to step through the slide presentation. Clicking on Transcript displays a transcript of the spoken material instead of the bullet point slides. Clicking on Commentary toggles sound on and off. In fact this feature self configures by detecting whether the sound files have been installed so that the application can be used on non-multimedia machines.

The program was implemented over a single ToolBook background with the majority of the scripting on the book script. Because the slide presentation screens all had a similar structure a set of handlers were written to control the step-by-step presentation of this material. The text for each screen of slides was held in two hidden text fields: one for the bullets and one for the transcript. The text in these fields was paginated by "*" characters in the text and these were searched for by the scripts extracting the text.

The graph pages were all individually scripted and no generalised system was developed for these.

The entire application was developed from the prototype in approximately ten days with a couple of extra days some months on to extend the graphing features. In use this ToolBook application has proved robust and reliable. However, the design is essentially hard wired into the OpenScript coding and substantive changes would require a substantive rework of the scripts and page designs.

Further details of this application may be obtained from the following address.

Steve Bolsin
ACTA
Consultant Cardiac Anaesthetist
Bristol Royal Infirmary
Marlborough Street
Bristol
BS2 8HW

Tel: 0117 928 2163
Fax: 0117 928 2098

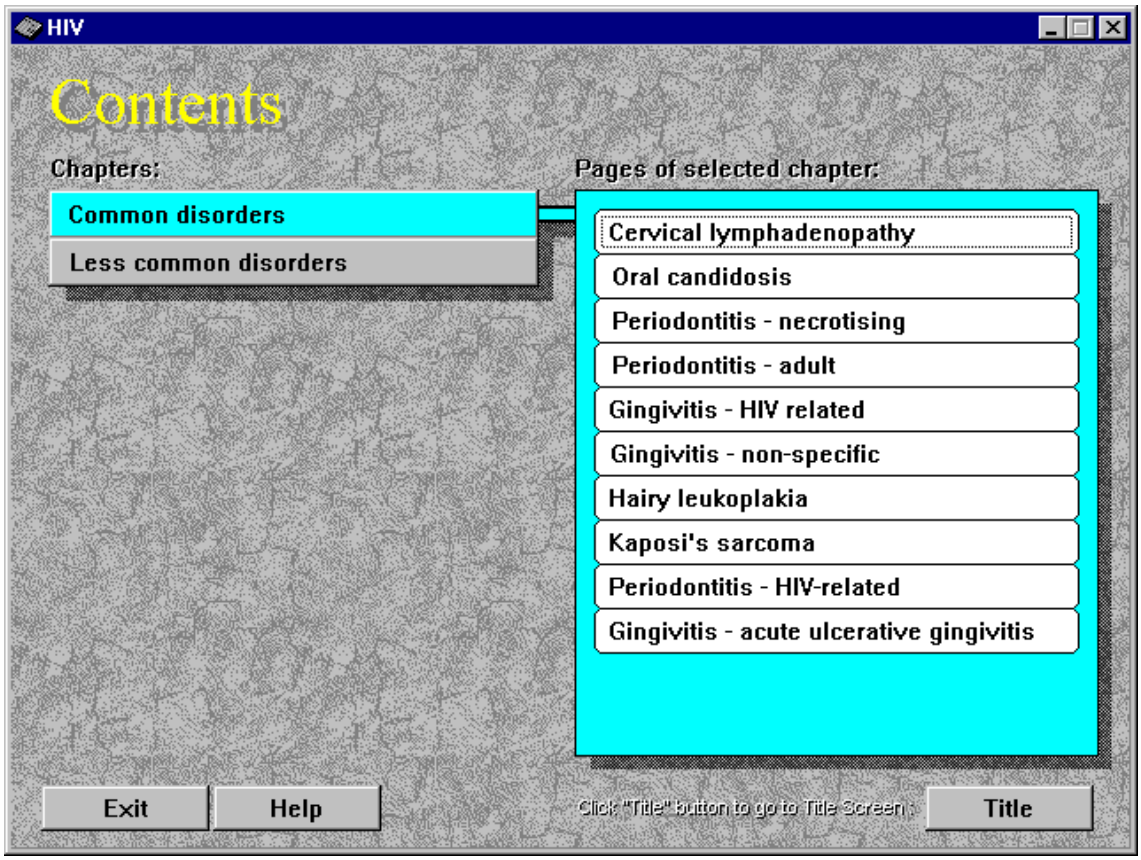
8.2. Case Study 2- Oral Manifestations of HIV

An application with content authoring performed entirely separately from the programming through the construction of a ToolBook template.

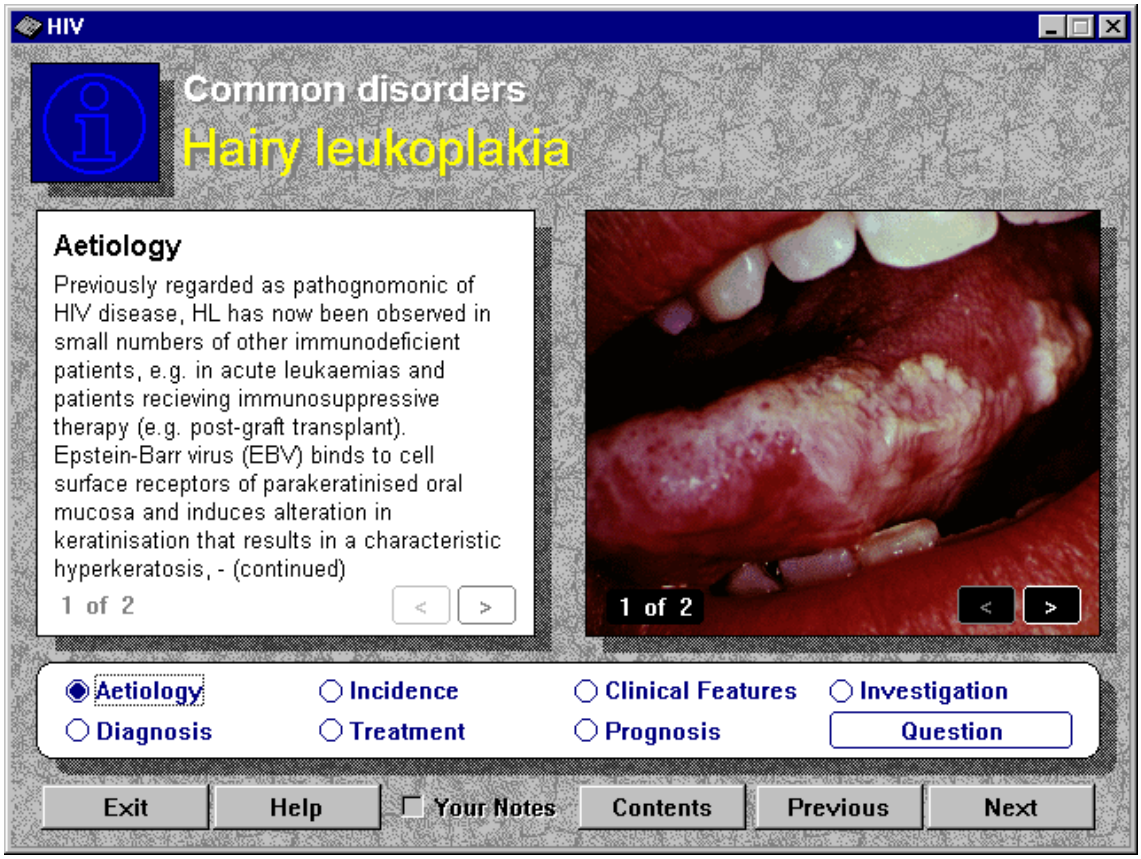
The Oral Manifestations of HIV program was designed for circulation to dental practitioners in the UK but at the same time it was envisaged that the general framework of the application could be re-used. Therefore, the Department of Health funded project, managed by the Computers in Teaching Initiative Centre for Medicine (CTICM), produced a prototype CAL framework for professional updating materials. This was first demonstrated at the Department of Health's CAL Dental Launch in London on 26 February 1993. Since then, further development work has resulted in an 'automated' template so that authors can produce tutorials without the need for programming knowledge.

This CTICM Template is a framework for the creation of Computer Assisted Learning (CAL) materials for practitioner updating. It was designed to meet the needs of non-programmer authors and experts in their field who wished to take advantage of the CAL medium without having to acquire or purchase programming skills. In this way, it becomes possible for experts to disseminate textual and graphical information to practitioners in a rapid and effective manner.

An application created in the template is structured as a series of chapters, each consisting of a number of pages. These are accessible to the user via an automatically generated contents screen as shown below. Clicking on a chapter button on the left displays a list of its pages on the right.



Clicking on a page button takes the user to that page. A typical page is shown below. Once on the page the user can click on buttons to view textual content and step through images. Only the buttons corresponding to the available information are enabled. Clicking on the Question button takes the user to a multiple choice question page. All pages in the book have this simple, standardised layout.

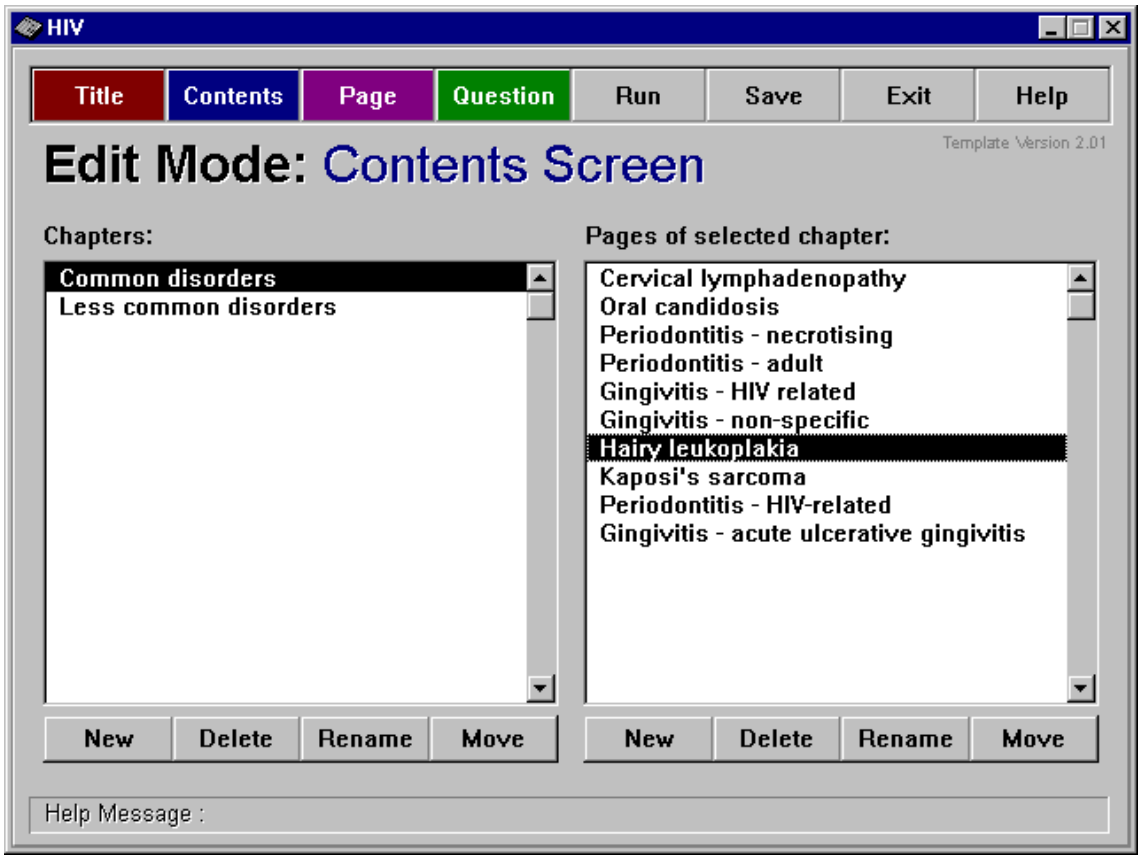


The important point of this design is that the content pages are not authored by hand; no one ever enters ToolBook author mode and positions each of these buttons and codes in the text and images. Instead, this is all done in reader mode by a non-programmer specialist filling in a series of forms. The design and functionality of the pages is set up by the ToolBook programmer who implemented the template; the content comes from the content specialist.

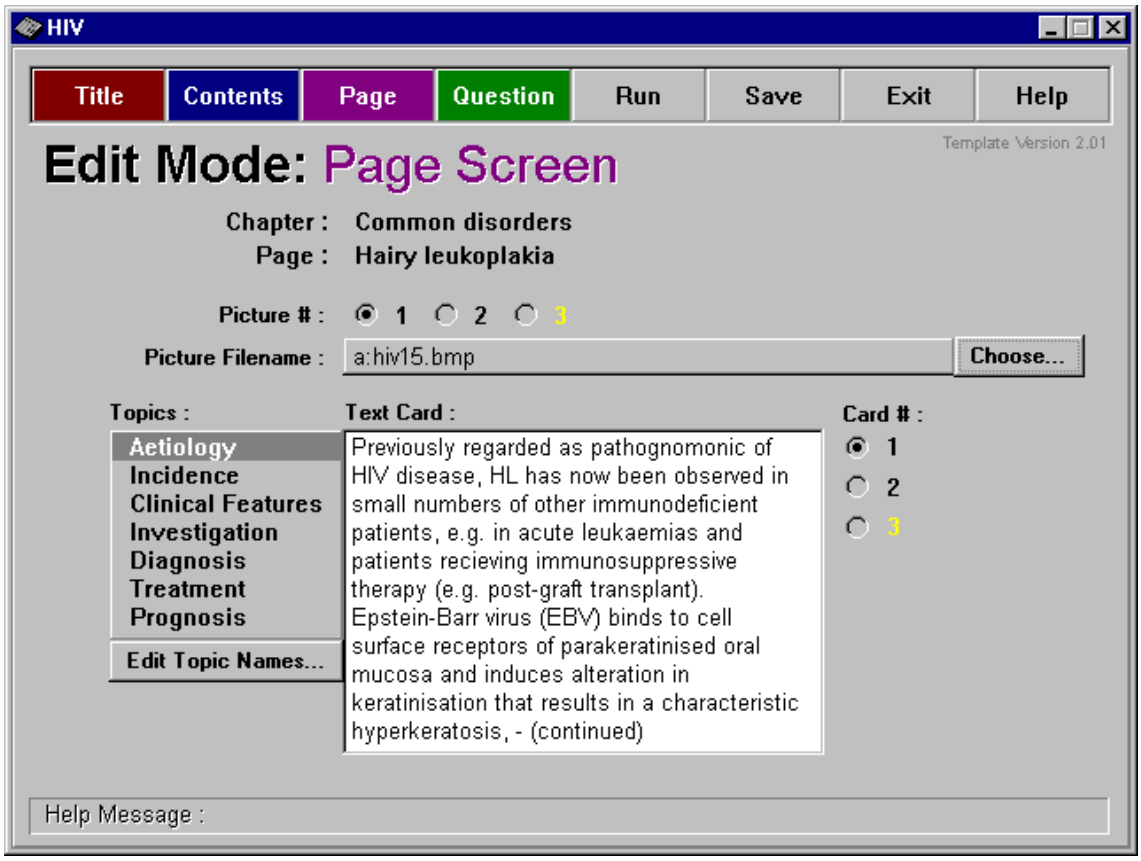
In practice, the form completion may be largely performed by a clerical assistant working from text written by the content author. The author can then perform an editorial and proofing role on the work once it is entered. As the content is keyed in, the template automatically creates a CAL program based on that information. The author may switch back and forwards between the forms (author view) and the CAL (user view) at any time during the authoring process.

Two of the four Edit Mode screens are shown on the following pages.

The Edit Mode Contents Screen allows the author to add, delete, rename and move both pages and chapters.



The Edit Mode Page Screen allows entry of text and specification of bitmap files.



The other Edit Mode screens not shown here allow the title, credits, copyright and objectives to be specified plus questions can be defined to accompany specific pages.

When the authoring is complete, the author simply clicks on the "Release" button and inserts blank floppy disks into the disk drive when prompted in order to create a distributable copy of the CAL program. This disk set (typically 2 or 3 floppy disks) may then be distributed for non-commercial educational purposes.

This will usually be accompanied by a customised version of the two page User Guide included in the paper-based Author's Guide and in this on-line help. The User Guide provides users with instructions on installing and starting up the software. Once in the software, on-line help is available on every screen.

A practitioner receiving a copy of the distributable disk set can install the software on their machine by running the Setup program on the disks. Having done this, they may then double-click on the icon created in their Program Manager to use the CAL software. If the author switched on password protection, the users are prevented from altering or changing the work by keeping them out of the forms screens.

ToolBook proved ideal for this approach to authoring. The non-content aspects of the application took less than twenty days in total, including the development of a help file for authors and paper based documentation. The template has since been used to produce other CAL tutorials for other areas of medicine and entirely unrelated disciplines.

The template was developed by an experienced ToolBook programmer and this approach does require more experience than that in Case Study 1.

Further details of this application may be obtained from the following address.

Sue Furber
CTI Centre for Medicine
University of Bristol
Royal Fort Annexe
Tyndall Avenue
Bristol BS8 1UJ

Tel: 0117 928 7492
Fax: 0117 925 5985
Email: cticm@bristol.ac.uk

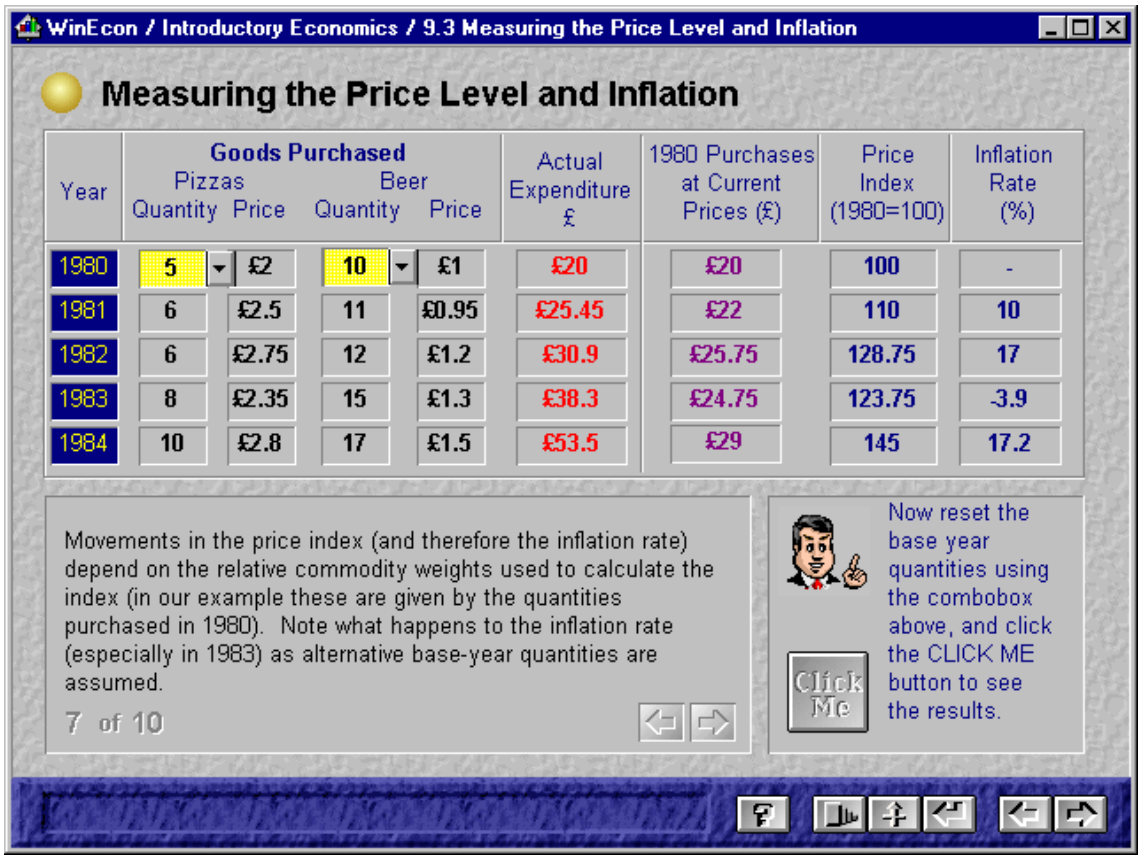
8.3. Case Study 3 - TLTP Economics' WinEcon

Large scale, multi-programmer, multi-author, multi-site development with collaborative authoring. Also uses ToolBook's expansion capabilities to extend the authoring environment.

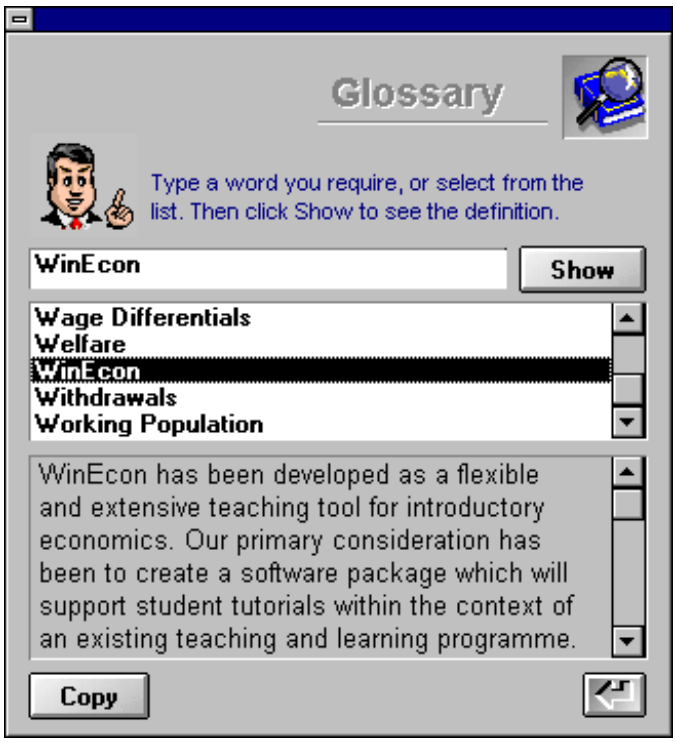
WinEcon is a single integrated teaching package covering the entire first year economics degree syllabus. It was produced over a three year period by the Teaching and Learning Technology Programme (TLTP) Economics Consortium managed by the Centre for Computing in the Social Sciences at the University of Bristol. The Consortium of eight UK universities appointed an economist project director and a senior programmer with some familiarity of ToolBook. A set of authoring tools were developed and 16 economist-programmers (typically research assistants with an economics background and some familiarity with computers) were appointed to work with the 35 content providing economics lecturers.

The student side of the package consists of a thousand teaching pages and a shell which allows the student to navigate around the teaching material. The shell and a typical teaching page are shown in the screen shots below.

The teaching pages were developed by different programmers at different institutions and yet maintain a consistent look and feel without restricting authors to a rigid page layout.



A series of tools such as a glossary, notebook and calculator are implemented as ToolBook viewers and are accessible from any point in the program. Also, each teaching page is indexed into four leading introductory economics textbooks.



From the lecturer's perspective there is a database consisting of a series of forms and data screens collectively known as WinEcon Lecturer. This allows lecturers to edit and extend the glossary, references, tests and exams. They can also rearrange courses, alter text and construct their own courses. To assist administration of courses, the results of tests plus logs of student usage can also be recorded and viewed. This is entirely implemented in ToolBook using direct book access from WinEcon to WinEcon Lecturer. While not the fastest database to work in, it was relatively cheap to develop compared to a similar development in Access or Paradox.

With the benefit of hindsight, more work should have been done in the early stages of the project to pin down the look and feel through the authoring tools. As it was, the tools deliberately constrained authors in terms of fonts, script location and some aspects of layout. However, ToolBook has so many degrees of freedom that it is almost impossible for developers, working apart, to arrive at exactly the same look and feel - much reviewing and reworking was required to achieve the current consistency. Fortunately, a tagging system using user properties was in place from the outset so automatic restyling of many objects was possible by writing OpenScript to scan books and do this. This ability of ToolBook to intraspect and examine the properties of its own objects proved one of the most time saving features of the tool in such a large scale development.

All things considered, ToolBook was a valid choice of tool and it enabled the implementation of features above and beyond the initial design at only marginal extra cost. WinEcon was awarded a medal by the British Computer Society in the 1995 Software Awards and received an award in the 1996 European Academic Software Awards. It is now used in over 90% of UK universities and over 200 universities around the world.

Further details of WinEcon may be obtained from the following address.

Phil Hobbs
TLTP Economics Consortium
Centre for Computing in the Social Sciences
University of Bristol
8 Woodland Road
Bristol BS8 1TN

Tel: 0117 9288478
Fax: 0117 9288473
Email: ccss@bristol.ad.uk

9. Conclusion

Asymetrix ToolBook 3.0 is a mature PC based authoring tool which is unlikely to ever appear on other platforms. Multimedia ToolBook 3.0 is a superset of ToolBook 3.0 which adds sound and video features to the basic ToolBook engine. Applications developed in ToolBook 3.0 will run under Multimedia ToolBook 3.0 but the reverse is not true - even if no multimedia features are used.

Asymetrix are continuously developing their ToolBook product line both in terms of add-ons, such as the Database Connection and the CBT Edition, and in terms of the core system itself - ToolBook 4.0 is the latest version and 5.0 (or Series II as it will be known) is currently in beta testing. Each subsequent version of ToolBook provides a high degree of backward compatibility but this has never been 100% in previous upgrades. However, neither 4.0 or 5.0 differ substantively from 3.0 with regard to the core ToolBook engine; instead, they each provide increased support for novice users through Specialists (wizards) and extra add-on extensions around the core. The critique and description of ToolBook 3.0 in this report applies equally well to 4.0 and 5.0 too.

ToolBook has proven to be a productive and flexible authoring tool suitable for implementing a wide range of applications. Applications may be distributed along with the ToolBook runtime royalty-free and, if correctly used, can be installed on an end-user system without needing to make changes to Windows initialisation files or the system boot configuration files. ToolBook applications may be run across a network although, realistically, graphically intensive programs are best run from a local drive (regardless of authoring system). Also, ToolBook books are able to communicate, exchange data with and manipulate other applications on the system.

All this flexibility and power is not without cost: although marketed as an authoring tool, ToolBook is in fact a full blown programming system. ToolBook is often adopted as an authoring tool by non-programmers on the grounds of its apparent ease of use. Indeed, initial progress on a project can be impressively fast. However this initial ease of use and rapid progress can be deceptive. As with many programming environments, finishing off the last 20% of an application requires a lot of specialist knowledge to provide the finishing touches and solve the inevitable hand full of problems which are unique to this particular application. However, because the first 80% of a ToolBook application can be so (relatively) easily constructed without too much specialist knowledge, the last bump in the learning curve is particularly steep and high. For small projects on a tight schedule, this last step can be prohibitively steep. Unfortunately, the fact that this "wall" is hit near the end of a project rather than being obvious from the outset means that, in the experience of the authors of this report, a lot of ToolBook projects run into problems in this last 20% and have to call in a consultant or revise the schedule and budget. Asymetrix do a very good job of selling ToolBook as an easy to use system but this masks this "completion problem" and can leave developers feeling very frustrated.

Given the numerous positive features of the package this "completion problem" should not be taken as a reason for not adopting ToolBook as a development tool - provided that experienced help is available or that the developer is willing to invest a substantial amount of time learning to program in OpenScript.

Appendix A - Purchasing

Northern Europe

(UK, Ireland, Scandinavia, Holland)
Sales, Educational Discounts and Training
ICS Solutions Ltd., Tempus Business Centre, Kingsclere Road, Basingstoke, U.K.
Tel. + 44 (0)1256 469460, FAX: +44 (0)1256 840494
Support
P.S.C., Willow Grange, Church Road, Watford, Herts., WD1 3QA, U.K.
Tel: + 44 (0)1923.208433, FAX: + 44 (0)1923.208419
Support: +44 (0)1923.208433, Support Fax: +44 (0)1923.208419
Asymetrix Ltd.
The Innovation Centre, 225 Marsh Wall, Docklands, London E14 9FW
Tel: +44 (0)171 454 1061, FAX: +44 (0)171 454 1062

United States *(World Headquarters)*

Asymetrix Corp., 110 - 110th Avenue N.E., Suite 700, Bellevue, WA 98004
Sales: (800) 448-6543, FAX: (206) 637-1504, Support: (206) 637-1600

Southern Europe *(European Headquarters)*

(France, Italy, Spain, Portugal, Belgium, Africa, Greece, Turkey, Middle-East)
Asymetrix S.A.R.L., CNIT-BP 417, 2, Place de la Défense,
92053 Paris La Défense, France
Tel: + (33) 1-46-92-24-34, FAX: + (33) 1-46-92-23-58
BBS: + (33) 1-47-62-96-67, Support: + (44) 1923.208433
Support Fax: + (44) 1923.208419

Central Europe

(Germany, Austria, Switzerland, Eastern Countries)
Asymetrix Info-Service, Postfach 10 01 63, D-80075 Münich, Germany
Tel: + (49) 1 80-5 35 25 25, FAX: + (49) 1 80-5 35 25 75
Support: + (44) 1923.208433, Support Fax: + (44) 1923.208419

Australia/Asia Pacific

236 Balaclava Road, Caulfield North, Victoria 3161, Australia
Tel: 011 613-9500-1333, FAX: 011 613-9500-1344
XL TECH, 3/252 Allambie Rd, Allambie Heights 2100, Sydney, NSW Australia
Tel: + (61) 2 975-2111, Fax: + (61) 2 975-2167

Japan

Something Good, City-Plaza Shinjuku , Bldg. 2-5-20, Okubu Shinjuku-Ku Tokyo 169
Tel: + (81) 3 3232 0803, Fax: + (81) 3 3232 0963

Appendix B - Technical Support

Asymetrix European Technical Support

European support of the ToolBook product family is contracted out to a company called PSC. If you ring with a query, you will need to quote the serial no. of your copy of ToolBook.

P.S.C.
Willow Grange
Church Road
Watford
Herts. WD1 3QA
U.K.

Tel. +44 (0)1923.208433
Fax. +44 (0)1923.208419
Email as per United States (see below)

Asymetrix US Technical Support

Technical support in the US, and worldwide support by email, is supplied by Asymetrix's own support staff based in the same suite of offices as the product developers. The best method for a quick response with a query is to use email.

Asymetrix Corp.
110 - 110th Avenue N.E., Suite 700
Bellevue
WA 98004

Sales (800) 448-6543
Fax. (206) 637-1504
Support (206) 637-1600
Email techsup@asymetrix.com

Appendix C - Bibliography

ToolBook Specific

Hall, T.L., "Utilizing Multimedia ToolBook 3.0", Boyd-Fraser, 1995

Hall, T.L., "Utilizing Multimedia ToolBook 4.0", Boyd-Fraser, 1996

Hobbs, P. (editor), "UK ToolBook User Conference 94 Proceedings", Centre for Computing in Economics, Bristol, March 1995

Hobbs, P. (editor), "UK ToolBook User Conference 95 Proceedings", Centre for Computing in the Social Sciences, Bristol, May 1996

Hustedde, S., "Developing with Asymetrix Toolbook: Applied Programming Theory", Wadsworth, October 1995

Holtz, M., "The Multimedia Workshop: Toolbook 3.0", Wadsworth, 1995

Natal et al, "Special Edition Using Asymetrix Multimedia Toolbook 4", Que, November 1995

Price, S., Asymetrix ToolBook 3 - Implications for Developers, Centre for Computing in Economics, Bristol, June 95

ToolBook Related

Gertler, N., "Multimedia Illustrated", Que, 1995

Tway, L., "Multimedia in Action", Academic Press, 1995

Appendix E - Internet Resources

Email	Asymetrix Technical Support	techsup@asymetrix.com
	ALT-T ToolBook User Group	toolbook@mailbase.ac.uk
ftp	Asymetrix FTP Site	ftp.asymetrix.com (pub)
	mirror	cutl.city.unisa.edu.au (/pub/windows/toolbook/asymetrix)
	TOOLB-L Archive	listserv.arizona.edu (/pub/listserv/toolb-l)
	ToolBook related	winftp.cica.indiana.edu (pub/pc/win3/toolbook)
	mirror	gateway.dec.com (micro/msdos/win3)
	mirror	gandalf.iat.unc.edu (several directories)
	mirror	ftp.cs.umd.edu (pub/hcil)
	mirror	nic.switch.ch
Lists	Discussion List	toolb-l@listserv.arizona.edu
	CIX ToolBook Conference	tool_book
Newsgroups	TOOLB-L (Bi-directional link with TOOLB-L list)	bit.listserv.toolb-l
Web	Asymetrix	http://www.asymetrix.com
	User Group	http://sosig.ac.uk/toolbook
	User Group List Archives	http://www.mailbase.ac.uk/lists-p-t/toolbook
	TB URL Depository	http://weber.u.washington.edu/~brianp/tbusers.html
	TB User's Web	http://www.lib.siu.edu/tbkwww/tbkwww.html
	mirror	http://www.ets.bris.ac.uk/tosolini/tbkwww/
	MM-WWW-PC	http://www.univ.trieste.it/mmwwwpc/mmwwwpc.html
	mirror	http://www.lib.siu.edu/tbkwww/tbkwww.html
	SuperCAL	http://www.staffs.ac.uk/supercal/supercal.htm
	TLTP	http://www.icbl.hw.ac.uk/tltp
	Asym,mirr	http://cutl.city.unisa.edu.au/pub/windows/toolbook/asymetrix
	T.Hall Book	http://www.thomson.com/rcenters/src/srcmmt.html
FAQ*	Asymetrix	http://www.asymetrix.com/techserv/techctr/faq/index.htm
	Search TOOLB-L archives	http://cutl.city.unisa.edu.au/toolbook
	Collected wisdom	http://weber.u.washington.edu/~brianp/tbusers.html

* FAQ = Frequently Asked Questions

Appendix E - UK ToolBook User Group

The UK ToolBook user group exists under the banner of the Association for Learning Technology (ALT). Details of ALT and the user group, known as ALT-T, are given below. Although nominally a UK group, membership is free and unrestricted so there is nothing to stop any Internet user joining and in practice this is very much an international group. At the time of writing this report, ALT-T membership exceeds 500.

What is ALT?

The Association for Learning Technology (ALT) provides a focus for the rapidly emerging community of learning technology practitioners in higher education. It brings together all those concerned with learning technology in higher education including researchers, developers, service providers, IT policy makers, librarians, computer manufacturers, software companies and publishers. At the heart are the academic staff in universities who are seeking to support their students' learning through the use of learning technology.

What does ALT do?

ALT meets the needs of its members, both individual and corporate, with a variety of services and activities. These are regularly evaluated and reviewed. As a membership organisation, ALT members are both recipients and providers of the following services.

- Journal - ALT-J
- Newsletter - ALT-N
- Electronic publishing
- Workshops, forums and conferences
- Award scheme
- Task groups
- Special interest groups and user groups

ALT ToolBook User Group

One of the key functions of a professional association such as ALT is to encourage the transfer of skills and good practice amongst its membership. User groups for specific authoring systems are way of transferring practical information in a way which allows the group's members to increase their efficiency and raise the quality of their work. To this end, a number of ToolBook users founded the ALT ToolBook user group (ALT-T) at the first UK ToolBook Users Conference in Glasgow 1993.

ALT-T aims to address the needs of both non-technical and technical users of ToolBook through the following activities.

- Electronic newsletter - ALT-T News
- World Wide Web site - ALT-T Web
- Annual conference - UK ToolBook Users Conference
- Close contact with Asymetrix Europe

Annual UK ToolBook Users Conference

ALT-T holds an annual UK ToolBook Users Conference, the details of which are announced in the ALT-T News newsletter. Proceedings of the 1994 and 1995 conferences are available from the following address.

Centre for Computing in the Social Sciences
University of Bristol
8 Woodland Road
Bristol BS8 1TN

Tel: 0117 928 8478 Fax: 0117 928 8473
Email: ccss@bristol.ac.uk
Web: http://sosig.ac.uk/ccss

Joining ALT

For further details, contact:

Association for Learning Technology
University of Oxford
13 Banbury Road
Oxford OX2 6NN

Phone: 01865 273273 Fax: 01865 273275
Internet: alt@vax.ox.ac.uk

Joining ALT-T

Membership of ALT-T is available free of charge to all ALT members and users of Asymetrix ToolBook. To join ALT-T, send an email message containing the following line to `mailbase@mailbase.ac.uk`

```
join toolbook <firstname> <lastname>
e.g.    join toolbook Simon Price
```

This message is handled by the mailbase list server without human intervention. In response to your message, an automatic reply will be emailed back confirming that you have been added to the toolbook mailbase list called `toolbook@mailbase.ac.uk`. You will subsequently receive the ALT-T News newsletter which is published electronically around six times a year. In addition, you will also receive occasional news bulletins detailing ToolBook related events, workshops and conferences. Back issues of the newsletter plus pointers to other sources of information are available at the following World Wide Web URL.

```
http://sosig.ac.uk/toolbook
```

Appendix F - ToolBook 3.0 Product Information

The following pages of this appendix are selected extracts from the ToolBook 3.0 Release Notes (RELNOTES.WRI) and give a detailed overview of the new features.

Contents

Note: Asymetrix's original section numbering has been preserved.

1.0. Features new in ToolBook 3.0

ToolBook 3.0 contains hundreds of new features, including a new interface, new and enhanced objects, and many improvements to OpenScript.

Authoring environment

1.1. New interface tools
1.2. Tools menu
1.3. Improved importing
1.4. Printing improvements
1.5. Enhanced color support
1.6. Managing embedded objects as resources
1.7. New tools for distributing applications

Objects

1.8. Drag and drop
1.9. Buttons
1.10. Fields and record fields
1.11. Groups
1.12. Hotwords
1.13. Shapes and lines
1.14. Menus
1.15. Viewers (windows)
1.16. Combo boxes
1.17. OLE objects

OpenScript

1.18. Improved Script editor
1.19. Improved Debugger
1.20. Auto-Script script library utility
1.21. Improved Command window
1.22. Improved OpenScript performance and efficiency
1.23. Improvements to variables
1.24. New operators
1.25. Improvements to messages
1.26. Notify handlers
1.27. New commands and functions (overview)
1.28. New commands
1.29. New functions
1.30. New messages
1.31. New properties
1.32. New DLL features and Windows messaging interface
1.33. New DLL functions

1.1. New interface tools

* **Tool bar**. Execute commands directly by clicking buttons on ToolBook's new tool bar.

* **Ruler shadows**. When you select an object, the object's dimensions display as shadows in the ruler so you can more precisely size and move it.

* **Status bar**. Display your own messages in the status bar. Turn the status bar on and off for any viewer in your application.

* **Right-click menus**. Click the right mouse button to display a context-sensitive menu and tool

bar you can use to manipulate an object as well as view and edit its properties.

* **New Tools menu**. Choose from a wide range of custom tools that make authoring easier and more efficient, including a Properties Browser, a window to define startup preferences, and other timesaving options.

* **Command window history**. Recall previously executed commands in the Command window using the Command window history.

* **Recently used books on File menu**. Open books quickly by choosing from the four most recently used books listed in the File menu.

* **Duplicate menu item**. Choose Duplicate from the Edit menu or press Ctrl+D to duplicate the selected object.

1.2. Tools menu

Adding TOOLS30.SBK to the `sysBooks` property adds a Tools menu at Author level that contains useful authoring tools. The layout of the menu is:

Tools
Property Browser
Startup Preferences
Center
Size
Spread
Set Tab Order
Add 3D Style
Import Text
Export Text
Find/Replace
Applications

Property Browser

Displays a window that allows you to edit and change properties, including user properties, for the selected object or for the page, background, viewer, or book. You can also use the Property Browser to view and set system properties and system variables.

You can also display the Property Browser from the right-click menu by clicking the Property Browser button on the right-click tool bar.

Note Because you cannot select viewers or hotwords, you must use the right-click menu to display the Property Browser for these objects. Or click the Viewers button on the Property Browser tool bar.

Startup Preferences

Displays a window where you can change startup properties such as grid settings, default stroke and fill colors, default page size, and startup system books.

Center

Centers selected objects in the window.

Size

Makes two or more selected objects the same size.

Spread
Arranges two or more selected objects in rows or columns.

Set Tab Order
Arranges the layer order of selected objects so that they follow one another in tab sequence.

Add 3D Style
Gives selected objects a three-dimensional look. This feature applies to fields, record fields, and rectangles. For example, if you choose Add 3D Style for a rectangle, ToolBook adds extra lines to the rectangle to give it a shadowed look.

Import Text
Imports text from a DOS file into the current field or record field.

Export Text
Exports text from the current field or record field to a DOS file.

Find/Replace
Searches for text in scripts throughout the current book and, if you specify, replaces it.

Applications
Launches predefined applications. To launch your own applications, add a line to the [Add On Tools] section of the TOOLBOOK.INI file. The format of the line is:

```
<Name that appears in listbox>=<File to run>,<Help text>
```

For example, this is the line used to launch the ScrapBook:

```
[Add On Tools]
ScrapBook=C:\TB30\SAMPLES\SCRAPBK.TBK,Launches the ToolBook ScrapBook
```

1.3. Improved importing

* **Importing pages**. Import pages from other books in one step. ToolBook automatically imports the associated resources and remaps the record field text.

* **Importing graphics**. Import a wider variety of graphics.

1.4. Printing improvements

* **Updated interface**. Use the updated printing user interface for easier access to printing.

* **WYSIWYG word wrapping**. Design your printed output more accurately with WYSIWYG (what you see is what you get) word wrapping, which makes text look the same on the screen as it does on the printed page.

* **Printing colors**. Control how well colors are rendered on black-and-white printers.

* **Page scaling**. Create more flexible printer output with improved page scaling.

* **Robust printing**. Enjoy consistent and reliable output with more robust printing support.

1.5. Enhanced color support

* **Solid colors**. Display custom solid object colors by setting a book's `solidColorsEnabled` property to `true`.

* **RGB values**. Set colors using RGB values instead of, or in addition to, HLS values.

* **New constants**. Use the new color constants `gray` and `lightGray`.

* **Palette shift**. Eliminate palette shift by assigning a shared color palette to a book.

* **16- and 24-bit color support**. Take advantage of 16- and 24-bit display devices in your applications.

* **Graphics of all color depths**. Add 8-bit, 16-bit, or 24-bit graphics to your application. ToolBook 3.0 automatically dithers the images to the color depth of the user's machine.

* **Windows colors**. Set objects to use the colors set in the Windows Control Panel with the new `useWindowsColors` property.

1.6. Managing embedded objects as resources

* **Centralized resources**. Maintain a centralized database of resources, including menu bars, icons, cursors, bitmaps, and color palettes, that you can use throughout your book.

* **Menu bars**. Create and modify menu bars without scripting using the new Menu Bar editor.

* **Icons and cursors**. Create and edit icons and cursors using the new Icon/Cursor editor.

* **Bitmaps and color palettes**. Create and edit bitmaps and color palettes, including those created with other Windows programs, using BitEdit or PalEdit.

* **Single-source changes**. Update every occurrence in a book of a bitmap, icon, color palette, or cursor with a single change. For example, if several graphic buttons share the same bitmap resource, ToolBook automatically updates each button when you edit or replace the bitmap.

* **Clip art**. Get a head start on user interface design by importing predefined icons, cursors, and bitmaps from the library of clip art included with ToolBook. 3.0.

1.7. New tools for distributing applications

* **.EXE files**. Save books as .EXE files that you can launch directly from the Program Manager without starting ToolBook or Runtime ToolBook first. (ToolBook must still be available in your path, however.)

* **Installation utility**. Package your applications with the new Asymetrix Setup Utility, which allows you to create installable components, assign icons to each file, and compress files for distribution on floppy disks or CD.

1.8. Drag and drop

* **Drag objects**. Define drag-and-drop behavior for any object to create highly interactive and intuitive applications.

* **Drag-and-drop properties**. Specify drag-and-drop properties using dialog boxes or create conditional drag-and-drop behavior in OpenScript.

* **Drag and no-drop cursors**. Use any graphic resource as a drag cursor, including full-color bitmaps (a capability unique to ToolBook).

1.9. Buttons

* **3D buttons**. Create three-dimensional radio buttons and check boxes for your application interface.

* **Graphical buttons**. Create custom checkboxes, radio buttons, and pushbuttons by adding graphics to your buttons.

* **Radio buttons**. Create radio button groups more easily with the new `autoRadioButtons` group property.

* **Disabled buttons**. Prevent users from using buttons by disabling the buttons with the new `enabled` property.

* **Mnemonic access characters**. Define custom keyboard interfaces in your application by assigning mnemonic access characters to your buttons, so that users can use Alt plus a keystroke to click the button.

1.10. Fields and record fields

* **Inline graphics**. Paste bitmaps and icon graphics into a field, or insert a graphic that you've imported into ToolBook's resource system.

* **Borderless fields**. Create transparent, borderless fields in one step using the borderless field tool.

* **Label buttons**. Create labels next to fields or in group boxes to define mnemonic access characters for objects that don't have captions.

* **Superscripts and subscripts**. Add superscripts and subscripts to your field text (invaluable for displaying mathematical and scientific data).

* **Color text**. Add visual emphasis to your fields with color. Each character in a field can be a different color.

* **Inset and raised styles.** Create three-dimensional fields and record fields using new styles.

* **Rich-text format (RTF) support**. Paste formatted text from word processing programs into fields or import it directly from a DOS file. Reduce development of text-rich applications by eliminating time spent reformatting imported text. Move formatted text, including hotwords, from one field to another using the `richText` field property.

1.11. Groups

* **Group editor**. Select objects in a group individually so you can edit them without destroying the group. You can also add objects to groups by selecting an object in the group and then drawing the new object, which automatically becomes part of the group.

* **Add or delete objects**. Add or delete objects in existing groups.

* **Auto-radio buttons**. Define radio button behavior more easily with the new `autoRadioButtons` property.

1.12. Hotwords

* **Hotword styles**. Define hotwords in color. Define a single hotword style for an entire book for convenience, but override it for individual hotwords for flexibility.

* **Importing hotwords**. Define hotwords in your word processing program, then import them into your ToolBook fields. This feature is very useful to anyone who is creating hypertext applications in ToolBook.

* **Finer control**. Control hotwords more precisely with the new `text`, `textOffset`, and `bounds` hotword properties.

* **Hotword graphics**. Create hotword graphics by including graphics in your hotwords.

1.13. Shapes and lines

* **Reshape objects**. Reshape polygons and arcs more easily with an enhanced Reshape command that allows you to add and remove vertices.

* **Line ends**. Add arrowheads and other effects to lines with the new line ends palette.

1.14. Menus

* **Menu Bar editor**. Use the new Menu Bar editor to create and edit menu resources without programming. You can define multiple levels of cascading menus. ToolBook 3.0 comes with a predefined Reader-level menu resource.

Use menu resources instead of modifying menus in the `enterBook` handler, as you did in ToolBook 1.5.

* **`menuItem` selected message**. Write a handler for the new `menuItemSelected` message instead of writing individual handlers for each menu item. (ToolBook still sends the menu command and alias messages if there is no `menuItemSelected` handler.)

* **New OpenScript commands**. Control menus more precisely with new OpenScript commands and functions that support menus, such as `menuEnabled`, `setMenuName`, `setMenuHelpText`, `removeSeparator`, and others.

* **Popup menus**. Display popup menus with the new `popupMenu()` function.

Tip You can use the tool bar to send many of the same commands that are defined in menus.

1.15. Viewers (multiple windows)

* **Multi-window applications**. Display pages and create multi-window applications, dialog boxes, palettes, tool and status bars, popup windows, nested child windows, and much more to display your information in a variety of ways.

* **Multiple concurrent windows**. Display multiple pages simultaneously.

* **Dialog boxes and palettes**. Create deluxe dialog boxes and palettes that make use of all of ToolBook's color and graphics capabilities.

* **Popup windows**. Create popup viewers that disappear when you click the mouse.

* **Captions**. Add thin captions to create low-profile windows such as tool palettes.

* **Replace multiple instances**. Combine multiple ToolBook instances in a single, powerful, easily-managed application.

1.16. Combo boxes

* **Drop-down list box**. Provide the choices of a list box in the space of a single-line field by creating combo box objects with drop-down lists.

* **Sort items**. Sort items in the drop-down list automatically.

1.17. OLE objects

* **Embed OLE 1 objects**. Embed or link any OLE 1.0 object into your ToolBook application.

* **Embed data from other Windows programs**. Add graphs, drawings, and other objects to your ToolBook pages by pasting them from OLE-compatible applications.

1.18. Improved Script editor

* **Multiple windows**. Edit multiple scripts simultaneously in multiple modeless Script editors.
* **Tool bar**. Use the Script editor tool bar as a shortcut for most commands.
* **Exchange text**. Import and export text files directly into and from the Script editor.
* **Block commenting**. Comment and uncomment blocks of code in one step.
* **Block indenting**. Promote and demote blocks in one step.
* **Parent script editing**. Edit the scripts of parent objects by choosing the name of the parent object from a menu.

1.19. Improved Debugger

* **Edit variables**. View and edit variable values of any length.
* **Command window**. Inspect and change any property using the Command window while in the Debugger.

1.20. Auto-Script script library utility

* **Learning tool**. Learn ToolBook quickly by browsing through predefined handlers, inserting a handler into a script, and testing the results.
* **Copy scripts**. Insert predefined handlers from a script library using the Auto-Script feature. You can customize options in each handler you insert.
* **Edit existing scripts**. Add your favorite handlers to an Auto-Script library file.
* **Programmer productivity**. Share Auto-Script libraries among multiple programmers.

1.21. Improved Command window

* **Command window history**. The Command window maintains a history of the most recent commands you executed. Use the PageUp and PageDown keys to review and recall the commands or execute any command by double-clicking it.
* **Support for Ctrl+arrow keys**. Jump from word to word using Ctrl+Left Arrow or Ctrl+Right Arrow.
* **Properties stored between sessions**. Command window position, size, and split bar height are written to TOOLBOOK.INI when you close the program. They're restored when you start ToolBook again and display the Command window.

Note You can no longer evaluate expressions directly in the Command window.

1.22. Improved OpenScript performance and efficiency

* **Faster compiler**. Write scripts that are up to 10 times faster than scripts in ToolBook 1.5 with the new optimizing OpenScript compiler. Looping and access to program variables are key areas where performance has been improved.

* **Larger scripts**. Write more complex programs that use larger variables. Each variable in OpenScript can contain 64K of data, and you can have up to 16MB of variable data (an increase from 32K in ToolBook 1.5).

* **Optional typed variables**. Improve efficiency of your programs by using optional typed variables, including types such as `int`, `word`, `string`, `object`, and many others.

1.23. Improvements to variables

* **Optional typed variables.** Improve efficiency of your programs by using optional typed variables, including types such as `int`, `word`, `string`, `object`, and many others.

* **Larger variables.** Write more complex programs that use larger variables; each variable in OpenScript can contain 64K of data, and you can have up to 16MB of variable data -- an increase from 32K in Multimedia ToolBook 1.5.

* **New assignment operator.** Assign values to variables and properties more conveniently using the new equal assignment operator (=), which also makes your OpenScript handlers faster to type and easier to read.

* **Arrays.** Access structured data quickly and easily using arrays of up to 16 dimensions. Use fixed arrays for fast, efficient management of data and dynamic arrays to store variable data whose size changes dynamically. Each array element can contain up to 65,536 characters.

1.24. New operators

* **Assignment operator**. Assign values to variables and properties more easily using the new equal assignment operator (=), which also makes your OpenScript handlers simpler to read.

* **Embedded graphics**. Manipulate graphics embedded in text with the new `graphic` text operator.

* **Bitwise operators**. Manipulate individual bits in numbers using the new bitwise operators. This capability is important when you're working with DLLs and certain algorithms.

* **Variable names**. Use the at symbol (@) to distinguish variable names from OpenScript keywords or property names.

* **Variable as message**. Use parentheses with the `send` command to evaluate an expression as a message.

1.25. Improvements to messages

* **Clicking mouse buttons**. Make buttons and other objects respond more like standard Windows controls with the new `buttonClick` message -- the message is only sent if the user releases the mouse button while the cursor is still over the object.

* **Linking and unlinking system books**. Write more reliable system books with the new `linkSysBook` and `unlinkSysBook` messages. These messages notify your system book when it is time to initialize or clean itself up, so you no longer need to rely on forwarded `enterSystem`, `enterBook`, or `enterPage` messages.

* **Entering and leaving applications**. Track when a book is opened and closed in the Main window with the `enterApplication` and `leaveApplication` messages. Because you can now display multiple books concurrently using viewers, you can use these messages in place of `enterBook` and `leaveBook` to perform application initialization and cleanup.

* **Pressing Alt+key combinations**. Write handlers for the new `keyMnemonic` message to define your own Alt+key combinations.

* **Sending variables as messages**. Send the contents of variables as messages using an enhanced `send` command. For example:

```
ask "Send what message?"
send (It)
```

* **Request notification of messages**. Write handlers that are notified when messages reach the current page or background. This allows you to create self-contained objects that maintain their customized behavior when they are copied.

1.26. Notify handlers

* **Self-contained objects**. Build intelligent, self-contained objects using the new notify handlers, which execute in response to messages that are sent to the object's page. For example, you can write a clock object that automatically updates the current time whenever the page receives an idle message, or you can write an animation object that starts playing when the page is entered. You can copy and paste these objects onto any page and they will work with no further scripting.

* **Libraries**. Use notify handlers to create libraries of intelligent objects that you and other ToolBook authors can reuse.

1.27. New commands and functions (overview)

* **Page transitions**. Add a variety of special effects to page navigation using the new `transition` command. Choose from `blinds`, `drip`, `push`, `slide`, `spiral`, and many others.

* **Number formats**. Format numbers as binary or hexadecimal with new options for the `format` command.

* **Importing books**. Import other books with the `import book` command.

* **Importing pages**. Import any single page or range of pages from any other book with the `import pages` command.

* **Copying objects**. Copy objects directly, without the Clipboard, using the `copyObject()` function.

* **Verify objects**. Test for the existence of an object with the `isObject()` function.

* **Verify formats**. Test whether a value matches a format using the `isType()` function.

* **Sounds**. Play .WAV files through your sound board using the `playSound()` function.

* **Financial functions**. Create "money-smart" applications using a wide range of financial functions.

* **Converting coordinates**. Convert easily between page units and pixels using new coordinate conversion functions.

* **Popup menus**. Display popup menus using the `popupMenu()` function.

1.28. New commands

ToolBook 3.0 includes the following new OpenScript commands.

Menu commands

Command	Description
disable menu	Deactivates an entire menu.
disable menuItem	Deactivates a single menu item.
enable menu	Activates a menu previously disabled with `disable menu`.
enable menuItem	Activates a menu item previously disabled with `disable menuItem`.
remove separator	Removes the line between items in a menu.

Objects

Command	Description
align <type>	Aligns selected objects.
copy resource	Makes another copy of an embedded bitmap, icon, cursor, palette, or menu bar.
drag	Starts a drag-and-drop operation.
sendNotifyAfter	Sends a message that triggers an object's `notifyAfter` handler.
sendNotifyBefore	Sends a message that triggers an object's `notifyBefore` handler.

Resource commands

Command	Description
export resource	Copies an embedded bitmap, icon, cursor, palette, or menu bar to a DOS file.
import resource file	Embeds a bitmap, icon, cursor, palette, or menu bar from a DOS
insert graphic	Embeds a bitmap or icon resource into a field.
replace resource	Overwrites an embedded bitmap, icon, cursor, palette, or menu bar with another.

remove resource	Discards an embedded bitmap, icon, cursor, palette, or menu bar imported previously with `import resource`.

Book, page, and background commands

Command	Description
import book	Copies all pages from another book to the current book.
import pages	Copies selected pages from another book to the current book.
save as EXE	Saves the current book with an. EXE shell that starts ToolBook automatically.
transition	Displays a special effect such as fade or wipe between pages.

Viewer (window) commands

Command	Description
activate <viewer>	Makes a particular viewer the current one.
close <viewer>	Closes a viewer.
hide <viewer>	Makes a viewer invisible but still available.
in <viewer>	Changes the active window temporarily in a script.
new viewer	Creates a new viewer.
open <viewer>	Initializes a viewer and makes its nonpersistent properties available.
show <viewer>	Displays a viewer opened previously with `openViewer`.

Other new commands

Command	Description
fill <array>	Fills all values in an array with a single value.
seekFile	Moves to a specified position in an ASCII file after opening the file with `openFile`.

1.29. New functions

ToolBook 3.0 includes the following new OpenScript functions. For more details about each function, see the OpenScript Reference Manual or the online Help.

Menu-related functions

Function	Description
menuEnabled()	Verifies whether a menu is active.
menuItemChecked()	Verifies whether a menu item has a checkmark next to it.
menuItemEnabled()	Verifies whether a menu item is active.
popupMenu()	Displays a popup (tear-off) menu.
setMenuHelpText()	Sets the status bar text that appears when this menu is highlighted.
setMenuItemHelpText()	Sets the status bar text that appears when this menu item is highlighted.
setMenuItemName()	Changes the name of a menu item.
setMenuName()	Changes the name of a menu.

Coordinate conversion functions

Function	Description
clientToPageUnits()	Converts a location in a client window in pixels to page units.
clientToScreen()	Converts a location in a client window in pixels into an absolute location in the screen.
frameToPageUnits()	Converts a location within a window's frame in pixels into page units.
frameToScreen()	Converts a location within a window's frame in pixels into an

absolute screen location.	
pageUnitsToClient()	Converts a location in page units into a location in a client window in pixels.
pageUnitsToFrame()	Converts a location in page units into a location in a frame window in pixels.
pageUnitsToScreen()	Converts a location in page units into an absolute screen location in pixels.
screenToClient()	Converts an absolute location on the screen into a location in a client window in pixels.
screenToFrame()	Converts an absolute location on the screen into a location in a frame window in pixels.
screenToPageUnits()	Converts an absolute location on the screen into page units.

Financial functions

Function	Description
annuityFactor()	Returns a factor of the present value of an annuity.
compoundFactor()	Returns the future value of an interest-bearing account.
ddb()	Returns depreciation of an asset for a specific period.
fv()	Returns the future value of an annuity.
ipmt()	Returns the amount of interest to be paid on an investment.
irr()	Returns the interest rate for a series of cash flow amounts.
nper()	Returns the number of periods required for an investment.
npv()	Returns the present value of an investment based on cash flow amounts.
pmt()	Returns the periodic payment for an annuity.
ppmt()	Returns the payment on principal for an investment.
pv()	Returns the present value of an investment.
rate()	Returns the interest rate per period for an investment.

Resource functions

Function	Description
chooseResource()	Displays the Choose Resource dialog box from which you can select a bitmap, icon, cursor, color palette, or menu bar resource.
GDIHandle()	Returns the handle of a resource in a format that can be passed to Windows.
resourceCount()	Returns the number of times a resource is referenced in the book.
resourceHandle()	Returns the handle of a resource in a format that can be passed tO ToolBook.
resourceList()	Returns a list of the resource references for a specified resource type.

Other functions

Function	Description
clipboardFormats()	Returns a list of formats currently on the Clipboard.
dimensions()	Returns the dimensions of the specified array.
flushMessageQueue()	Clears pending keystrokes, mouse clicks, or messages.
isObject()	Determines if the specified object exists.
isType()	Determines if the specified variable matches a particular data type.
playSound()	Plays a .WAV file.
windowFromPoint()	Returns a reference to the topmost viewer (window) displayed at the specified location.
windowRefFromHandle()	Returns a reference to the viewer (window) identified by the

specified handle.

1.30. New messages

ToolBook 3.0 includes the following new OpenScript messages.

Menu event messages

Message	Description
align<type>	Aligns selected objects.
color	Toggles display of the Color Tray.
find	Displays the Find dialog box.
graphic	If the focus is in a field and a character is selected, displays the Graphic dialog box, from which the user can choose a graphic to insert into text.
insertGraphic	Inserts a graphic resource (bitmap, icon) into a field.
line	Displays the line palette.
lineEnds	Displays the line ends palette.
normalScript	Removes superscripting or subscripting from the selected text.
pasteSpecial	Displays the Paste Special dialog box, from which the user can choose the format for pasting an object from the Clipboard (including creating an OLE object).
pattern	Displays the pattern palette.
polygon	Displays the polygon palette.
printSetup	Displays the Print Setup dialog box, from which the user can choose the current printer.
readerRightClick	Toggles the `sysReaderRightClick` property, which determines whether the right-click menus appear at Reader level.
regular	Changes a font style to `regular` (no italics or bold).
replace	Displays the Replace dialog box to allow the user to specify replacement text when searching for text.
resources	Displays the Resource Manager dialog box, from which the user can import, edit, or remove resources.
saveAsExe	Displays the Save As .EXE dialog box, from which the user can specify the name of the .EXE file to create.
sendMail	Displays the Send Note dialog box, from which the user can send an electronic mail message.
statusBar	Toggles the display of the status bar.
subscript	Formats the selected text as subscripted.
superscript	Formats the selected text as superscripted.
tool	Toggles the display of the tool palette.
toolBar	Toggles the display of the tool bar.

Enter event and leave event messages

Message	Description
enterApplication	Sent when a book opens in ToolBook's Main window.
enterComboBox	Sent when a combo box receives the focus.
enterDrop	Sent when the cursor enters the bounds of an object during a drag-and-drop operation.
enterDropDown	Sent when a user clicks the combo box pushbutton to display the drop-down list.
enterMenu	Sent just before a menu is shown.

enterWindow	Sent when a viewer gets the focus.
leaveApplication	Sent when the book in ToolBook's Main window is closed.
leaveComboBox	Sent when a combo box loses the focus.
leaveDrop	Sent when the cursor leaves the bounds of an object during a drag-and-drop operation.
leaveDropDown	Sent when a combo box's drop-down list box closes.
leaveWindow	Sent when the target window changes.

Mouse and keyboard event messages

Message	Description
buttonClick	Sent when the user presses and releases the mouse button while remaining over one object.
keyMnemonic	Sent when the user presses an Alt+key combination not already defined as a button or menu mnemonic access key.

Standard messages

Message	Description
autoScript	Displays the Auto-Script dialog box, from which the user can insert a prewritten script.
convertPicture	Converts the currently selected picture object to a paint object.
customEdit	Sent when the user clicks the Custom Edit button in an object's right-click menu. If the current system books include TOOLS30.SBK, this message displays the Property Browser.
debugScript	Displays the Debugger.
editScript	Displays the Script editor.
pageSize	Displays the Page Size dialog box, from which the user can choose a new size for the current book or background.
showGrid	Toggles the visibility of the grid.
sizeToViewer	Contracts or expands the current background to the size of the viewer in which it is displayed.
snapGrid	Toggles whether objects snap to the grid.

Notification messages

Message	Description
linkSysBook	Sent when a book is added to the `sysBooks` property.
menuItemSelected	Sent when a menu item is selected.
pageScrolled	Sent when the page scroll of a viewer changes to indicate how far a page has been scrolled.
selectChange	Sent when the user selects an item from a combo box's drop-down list box.
selectionChanged	Sent when the object selection changes at Author level.
shown	Sent when a viewer is shown.
sized	Sent when a viewer is sized.
unlinkSysBook	Sent when a book is removed from the `sysBooks` property.

Drag-and-drop messages

Message	Description
allowDrag	Sent to query whether the object can be dragged.
allowDrop	Sent to query whether the object will allow another object to be dropped on it.
beginDrag	Sent to the dragged object when a drag-and-drop operation begins.
endDrag	Sent to the dragged object when it is dropped.

enterDrop	Sent to an object when the drag cursor enters its bounds.
leaveDrop	Sent to an object when the drag cursor leaves its bounds.
objectDropped	Sent to the destination object when a dragged object is dropped on it.
stillOverDrop	Sent continuously to an object while the drag cursor is above it.

OLE messages

Message	Description
insertOLEObject	Displays the Insert OLE Object dialog box, which lists the types of OLE servers available.
oleAction	Executes the specified action for the selected OLE object.
oleLinks	Displays the OLE Links dialog box, which allows users to maintain links for OLE objects.
pasteSpecial create	Displays the Paste Special dialog box, from which the user can create an OLE object by pasting it from the Clipboard.

Viewer messages

Message	Description
closeWindow	Sent to a viewer when it is closed.
enterMenu	Sent to a viewer when it is activated.
hidden	Sent to a viewer when it is hidden.
leaveWindow	Sent to a viewer when it loses the focus.
menuItemSelected	Sent to a viewer when the user chooses a menu item.
moved	Sent to a viewer when it is repositioned.
newViewer	Displays the New Viewer dialog box, from which the user can create a new viewer.
openWindow	Sent to a viewer when it is opened.
shown	Sent to a viewer when it is shown.
sized	Sent to a viewer when it is resized.
viewers	Displays the Viewers dialog box, from which users can choose a viewer to edit or create a new one.

1.31. New properties

ToolBook 3.0 includes the following new properties.

Books, pages, and backgrounds

Property	Description
hotwordColor	Specifies the default color for hotwords.
hotwordStyle	Specifies the default style (`frame`, `color`, or `none`) in which ToolBook displays hotwords.
keepMenubar	Specifies whether the current menu bar should be retained when opening a new book.
notifyObjects messages	Lists objects on a page that have requested notification of reaching the page.
percentFreeSpace	Specifies the approximate percentage of free space available on a page or background.
saveOnClose	Specifies how changes are saved when the book is closed.
shownBy	Lists the viewers currently showing a specified page.
skipNavigation the	Prevents ToolBook from displaying the page at Reader level when

	user navigates to it using the Page menu or using a script with implicit navigation commands (`send next`, `send previous`, and so on).
solidColorsEnabled	Specifies whether a book uses solid colors or dithered colors.
windows	Lists the viewers in a book.

Buttons

Property	Description
checkedGraphic	Specifies the graphic resource (bitmap, cursor, or icon) that is displayed when a checkbox-style button is checked.
disabledGraphic	Specifies the graphic resource that is displayed when a button is disabled.
invertGraphic	Specifies the graphic resource that is displayed when an enabled button is clicked and held down.
normalGraphic	Specifies the graphic resource that is displayed when a button is enabled and not being held down.

Lines

Property	Description
lineEndSize	Specifies the size of a line's arrowhead and other ends.
lineEndStyle	Specifies the style of a line's ends (filled or open arrowheads and tails).

Fields and record fields

Property	Description
richText	Specifies the RTF version of text, which includes information about character and paragraph formatting.

Hotwords

Property	Description
hotwordStyle	Specifies how ToolBook should display hotwords (`frame`, `color`, or `none`).

Groups

Property	Description
autoRadioButtons	Specifies whether the radio buttons in a group operate as mutually exclusive buttons.

Viewers

Property	Description
authorStatusBar	Specifies whether a viewer will display a status bar by default at Author level.
imageBuffers	Specifies the number of buffers (0, 1, or 2) available to store pages and backgrounds for the current viewer.
readerStatusBar	Specifies whether a viewer will display a status bar by default at Author level.

Multiple objects

Property	Description
defaultAllowDrag	Specifies whether an object is dragged by default when a user clicks it.
defaultAllowDrop	Specifies whether an object accepts a drop by default.
dragImage	Specifies the graphic resource (bitmap, cursor, or icon) that is displayed when the object is dragged.

enabled	Specifies whether the object can respond to mouse clicks and receive the focus.
noDropImage	Specifies the graphic resource (bitmap, cursor, or icon) that is displayed when the object is dragged over an object that does not accept a drop.
notifyAfterMessages	Lists the messages for which the object has requested notification with `notifyAfter` handlers.
notifyBeforeMessages	Lists the messages for which the object has requested notification with `notifyBefore` handlers.
rgbFill	Specifies the object's fill color using RGB values (instead of HLS values).
rgbStroke	Specifies the object's stroke color using RGB values (instead of HLS values).

System

Property	Description
caretLocation	Specifies the location of the insertion point within a field or record field.
startup3DInterface	Specifies whether ToolBook displays dialog boxes using a three-dimensional effect by default.
sys3DInterface	Specifies whether ToolBook currently displays dialog boxes using a three-dimensional effect.
sysToolBookDirectory	Returns the path from which ToolBook was started.

1.32. New DLL features and Windows messaging interface

* **New functions**. Extend your application with new functions in the DOS (formerly File), Windows, and Dialog DLLs.

* **Database functions**. Call the Paradox database engine to create and access Paradox database tables and indexes using the new Paradox DLL.

* **Resource handles**. Manipulate resources with Windows API functions by getting a handle for the resource with the `GDIHandle()` function.

* **Windows messages**. Extend your ability to intercept Windows messages with an improved `translateWindowMessage` control structure, which gives you complete control over how and when a message is sent to a window. You can intercept any Windows message and prevent it from reaching its window, and you can add your own custom processing for any Windows message. To make it easier for you to find the right Windows message, ToolBook includes an online Help application called WINCONST.HLP that lists Windows constants.

1.33. New DLL functions

ToolBook 3.0 includes the following new DLL functions.

TB30DLG.DLL

```
addComboBoxItem()
addListBoxItem()
chooseColorDlg()
chooseDirectoryDlg()
```

```
chooseFontDlg()
colorPaletteDlg()
controlIDToName()
controlNameTohWnd()
controlNameToID()
deleteComboBoxItem()
deleteListBoxItem()
deletenComboBoxItem()
deletenListBoxItem()
dialogCallback()
enableControl()
endTBKDialog()
enterWaitState()
getControlText()
getCustomColors()
getDialogFocus()
getListBoxItems()
getListBoxSelection()
getnListBoxSelection()
getOpenFileDlgFilterIndex()
getSaveAsDlgFilterIndex()
isButtonChecked()
isControlEnabled()
leaveWaitState()
openFileDlg()
saveAsDlg()
setBitmapData()
setButtonCheck()
setComboBoxItems()
setControlText()
setCustomColors()
setDialogFocus()
setGroupedButtonCheck()
setIconData()
setListBoxItems()
setListBoxSelection()
setnListBoxSelection()
```

TB30DOS.DLL

```
getCDDriveList()
getDirectoryOnlyList()
getDriveKind()
getFileOnlyList()
getVolumeName()
isCDDRive()
setFileDate()
setSystemDate()
setSystemTime()
```

TB30WIN.DLL

```
getIniVar()
getModuleList()
getModulePath()
popText()
popTextGetBounds()
sendKeys()
setIniVar()
```

2.0. Converting books from earlier versions

2.1. How to convert books from earlier versions

The ToolBook 3.0 file format has changed from ToolBook 1.5. To use your existing 1.5x books in ToolBook 3.0, you must convert them.

Important Once a book is converted to the new file format, you will not be able to open or otherwise access it in ToolBook 1.5.

To convert your 1.5x books:

1. Back up all your books before opening them in ToolBook 3.0.

   ToolBook automatically backs up a version of each book it converts and gives it the extension .OLD. However, for extra safety you should save your own backup copy of each file.

   Be especially careful to back up all system books and other books that are accessed from within your application.

2. Confirm that you have enough free space on your hard disk to save a duplicate copy of your books.

3. Open each book individually.

   When you open your ToolBook 1.5x books in ToolBook 3.0, your books are automatically converted to ToolBook's new file format. This process may take some time if your book is large.

Recompiling scripts

ToolBook 3.0 features an updated compiler. To bring scripts up to date, ToolBook recompiles them automatically.

If errors are reported during the conversion, check the conversion log file to determine the source of the error. We recommend that you then run the Script Walker utility (SCRWALK.TBK) and open each book to update all your scripts in one pass. Remember to update your system books as well. (You do not need to update system books shipped with ToolBook 3.0, such as TOOLS30.SBK).

Note If your ToolBook 1.5 scripts use syntax no longer supported by ToolBook 3.0, they cannot be recompiled until you fix the syntax errors.

Keyword conflicts

While converting books from ToolBook 1.5x, ToolBook 3.0 automatically adds an at symbol (@) to the beginning of any user-defined property, function, or variable in your scripts that conflicts with a new keyword. This ensures that your scripts will continue to execute correctly.

Note If scripts cannot be converted (for example, if they use syntax no longer allowed in ToolBook), ToolBook marks the script with a special code to indicate that it is an old script that was not converted. Edit the script to remove syntax errors. ToolBook will then automatically detect that it

has been marked and add the @ operator to words that conflict with new keywords.

ToolBook cannot automatically add an at symbol to the following keywords because they are used with DLL declarations:

BYTE
DWORD
DOUBLE
FLOAT
INT
LONG
POINTER
STRING

If these words appear in your script as the names of variables or user properties, you must manually add the at symbol as a prefix or change the name.

Books with stripped scripts

ToolBook cannot convert books whose script text has been removed with the Script Remover (REMOVER.EXE) utility provided in the ToolBook 1.5 Developer Utilities. ToolBook recompiles all scripts when it converts books to the new file format, so the original script text must be present. If you open a book whose scripts have been removed, ToolBook converts the objects.

2.2. Maintaining old and new versions of ToolBook

When you install ToolBook 3.0, you can continue working with earlier versions of ToolBook. Keep these points in mind when using more than one version of ToolBook:

* If you convert a book to ToolBook 3.0 format, you can no longer use it with earlier versions of ToolBook. If you want to use a book with both ToolBook 1.5 and ToolBook 3.0, you must make a copy for each version.

* ToolBook executables (.EXE files) and DLLs have new names in ToolBook 3.0 so that they don't conflict with older versions of these files. You can continue to use older file names in the scripts of your ToolBook 1.5 applications; when you convert applications to ToolBook 3.0, update them by substituting new file names.

* You can launch the appropriate version of ToolBook automatically when you click on a .TBK file in the Windows File Manager or use the Program Manager Run command to run a .TBK file. If you associate the new utility TBLOAD.EXE with your .TBK files, it can detect which version of ToolBook created the file and launch it.

2.3. Recompiling scripts created with Early Access versions of ToolBook

ToolBook 3.0 features an updated compiler. To bring scripts up to date, ToolBook recompiles them automatically if it detects that the script was compiled using the older compiler. Books created with Early Access will initially experience a slowdown as ToolBook encounters and recompiles scripts not yet updated. We recommend that you run the Script Walker utility (SCRWALK.TBK) and open each book to update all your scripts in one pass. Remember to update your system books as well (you do not need to update system books shipped with ToolBook 3.0 such as TOOLS30.SBK).

3.0 Features that work differently in ToolBook 3.0

3.1. Changed keyboard accelerators

A number of ToolBook's keyboard accelerators have changed. To view the new keyboard accelerator assignments, browse through the menus.

You can now move the edit cursor one word at a time in a field by pressing the Ctrl+Left Arrow or Ctrl+Right arrow keys. To accommodate this change, ToolBook's new keyboard accelerators for page navigation are as follows:

Navigation	Old accelerator	New accelerator
Previous Page	Ctrl+Left	Alt+Left
Next Page	Ctrl+Right	Alt+Right
First Page	Ctrl+Up	Alt+Up
Last Page	Ctrl+Down	Alt+Down

Note To move from word to word in the Command window, use Alt+arrow keys.

3.2. Evaluating expressions in the Command window

Because of the new assignment operator (=), you can no longer evaluate expressions by typing them

in the Command window by themselves. Instead, enter a full command such as the following:

```
put sqrt(540)
request uniqueName of this book
```

3.3. Starting Runtime ToolBook without a file name

You can start the Runtime version of ToolBook 3.0 without specifying a file name in the command line. In ToolBook 1.5, executing TBOOK.EXE without a file name would result in an error message and ToolBook would not start. If you start Runtime ToolBook 3.0 (TB30RUN.EXE) without a file name, ToolBook displays an Open dialog box from which you can choose a book.

3.4. ID numbers of objects during recording

As in ToolBook 1.5, ToolBook 3.0 inserts the keyword `selection` into a script as you are recording it. However, if you use the right-click menu to change an object's properties while recording, ToolBook inserts the object's actual ID number. You should examine scripts you record with the script recorder in ToolBook 3.0 to be sure that this change does not affect how the recorded script runs.

3.5. Change in file format

ToolBook 3.0 features a new file format for books (.TBK and .SBK files). Files from older versions of ToolBook are converted automatically when you open them using ToolBook 3.0, or when you refer to an object in them.

> **Important** After you convert a book using ToolBook 3.0, you cannot open it using an earlier version. Be sure to make backup copies of books that you intend to use with both versions.

3.6. Using the TOOLBOOK.INI and ASYM.INI files

ToolBook 3.0 keeps startup information in the TOOLBOOK.INI file (in contrast to ToolBook 1.5, which kept this information in a [TOOLBOOK] section of the WIN.INI file).

Information about graphic filters is now stored in the ASYM.INI file, which can be shared with other Asymetrix applications such as MediaBlitz! and Compel. ToolBook 1.5 stored this information in the [ToolBook Filters] section of the WIN.INI file.

3.7. New executable and DLL file names

The names of the ToolBook executable (.EXE) files and DLLs have been changed in ToolBook 3.0, so you can continue using your ToolBook 1.5 applications without changing names or scripts that link DLLs.

The old and new names appear in the tables below.

Development version files

Old name	New name
toolbook.exe	tb30.exe
tbknet.exe	tb30net.exe
tbkbase.dll	tb30bas.dll
tbkutil.dll	tb30utl.dll

tbkcomp.dll	tb30cmp.dll
tbkedit.dll	tb30edt.dll

Optional runtime files

Old name	New name
tbkdb3.dll	tb30db3.dll
tbkdlg.dll	tb30dlg.dll
tbkwin.dll	tb30win.dll
tbkfile.dll	tb30dos.dll

Runtime ToolBook files

Old name	New name
tbkrun.exe	tb30run.exe
tbknet.exe	tb30net.exe
tbkbase.dll	tb30bas.dll
tbkutil.dll	tb30utl.dll
tbkcomp.dll	tb30cmp.dll

3.8. Importing picture graphics

When you imported any graphic that created a picture in ToolBook 1.5, the graphic was imported at a fixed size, regardless of its natural size. In ToolBook 3.0, graphic files are imported at their true size, which can be as large or larger than the Main window. If you want to import graphic files and create fixed-size picture objects out of them, use a script such as the following:

```
importGraphic "C:\GRAPHICS\SAMPLE.TIF"
get size of selection
size of selection = 1920,1920*(item 2 of It/item 1 of It)
```

3.9. `visible` property of groups

In ToolBook 1.5, if you set a group's `visible` property to `true`, the `visible` property of all the objects in the group was set to `true` as well, and they became visible. If you set the group's `visible` property to `false`, all objects in the group disappeared as well.

ToolBook no longer passes the group's `visible` property to individual objects in the group. If you hide a group, all objects in it disappear. However, if you show a group, only the objects whose `visible` property is `true` appear. If an object's `visible` property is `false`, it remains hidden.

This change may affect ToolBook 1.5 applications that show or hide all objects in a group by showing or hiding the group. To work around this problem, you can show or hide the objects of the group. For example, these ToolBook 1.5 statements show or hide all objects in a group:

```
show group id 23
hide group id 23
```

To do the same in ToolBook 3.0, use statements such as these:

```
show objects of group id 23
hide objects of group id 23
```

If you have nested groups, you must execute these commands on all the nested groups in turn.

3.10. Changes in wordwrap behavior

ToolBook 3.0 wraps text slightly differently than ToolBook 1.5. All text now wraps the same way on the printed page as it does on screen.

If you notice that text in fields or button captions is wrapping and clipping differently, resize your fields and buttons to regain the same wrapping and clipping of text as in ToolBook 1.5.

3.11. New Author menu

The ToolBook 3.0 Author-level menu bar is changed from ToolBook 1.5. Scripts that modify the Author-level menu bar may not work correctly if they refer to menus or items that no longer exist.

In addition, menus are now resources that can differ from book to book. For backward compatibility, ToolBook 3.0 provides the `keepMenuBar` property that specifies whether the menu bar should be retained when you switch between books.

3.12. Author menu item in Runtime ToolBook

Runtime ToolBook no longer automatically removes the Author menu item from the default menu bar resource assigned to the Main window. To remove it, use the OpenScript `remove menuItem` command.

You may want to remove the Author menu item from all menus, even in the full version of ToolBook. Because the F3 accelerator key is always available, you do not require the Author menu item in order to switch to Author level in the full version of ToolBook.

3.13. Extended user property and variable references

ToolBook 3.0 allows you to add the at symbol (@) prefix to the names of user properties, user-defined functions (`to get` or `to set` handlers), or variables to distinguish them from OpenScript keywords.

The following statements illustrate using @ to distinguish a user property name from a keyword:

```
text of field id 0 = "this goes into the text of the field"
@text of field id 0 = "this goes into a user property"
get @text of field id 0
```

The following example illustrates using @ for variable references. Without the prefix, ToolBook would generate a syntax error because `style` is a property name:

```
local @style
@style = 1
```

There are over 180 new keywords that may conflict with user properties or variable names you used in ToolBook 1.5. For example, one of ToolBook's new keywords is `enabled`; if you had a user property named `enabled` in ToolBook 1.5, you must now use the @ prefix to get or set its value:

```
get @enabled of button "Quit"
```

```
set @enabled of button "Quit" to false
```

3.14. Passing `null` to a DLL

ToolBook 3.0 treats `""` and `null` differently when you use them as string parameters to a DLL function. If you specify `""`, ToolBook passes a pointer to an empty string (`"\0"`). If you specify `null`, it passes a null pointer (`0:0`). In contrast, ToolBook 1.5 passes a pointer to an empty string in both cases.

This change makes it easier to pass a null pointer to DLLs that distinguish between a pointer to an empty string and a null pointer (for example, the Windows `FindWindow( )` function).

To ensure complete compatibility with ToolBook 1.5, change your scripts to use `""` instead of `null` when passing empty strings to DLL functions. If you do not make this change, you may encounter a general protection fault when a script a passes `null` to a DLL.

If you write your own DLLs, be sure a string parameter is a null pointer before dereferencing it. Otherwise, your DLL may cause a general protection fault when ToolBook passes `null` to it.

3.15. Stricter OpenScript syntax

ToolBook 3.0 enforces OpenScript syntax more rigorously than ToolBook 1.5 did. When you save your scripts, the OpenScript compiler does not allow some syntax that was allowed in ToolBook 1.5. You may not be able to save some of your scripts without making minor modifications to eliminate syntax errors.

To check the syntax of an individual script, display the script in the Script editor, then choose Check Syntax from the File menu.

To review all scripts in the book in one pass, run the Script Walker utility included with ToolBook 3.0 by double-clicking its icon from the ToolBook 3.0 Program Manager group. Script Walker examines all of the scripts in a converted book and finds those that contain syntax errors.

3.16. Changes in `to get` handlers

Unlike ToolBook 1.5x, ToolBook 3.0 requires every code path in a `to get` handler to contain a return statement. If a `to get` handler does not contain the necessary `return` statements, its script will not compile when the book is converted. You can fix this by editing the script and adding `return` statements at appropriate locations.

For example, you might be missing a `return` statement in a `conditions` control structure that doesn't have a default branch and has no `return` statement after it. Even if you provide a `return` statement for each branch, ToolBook requires that you provide a default `return` statement of some kind. The following handler is missing a `return` statement:

```
to get calcVal x
   conditions
      when x is 1
         return 2
      when x is 2
         return 3
   end conditions
end calcVal
```

The following two scripts illustrate corrected versions of this handler.

```
to get calcVal x
   conditions
      when x is 1
         return 2
      when x is 2
         return 3
      else
         return 0 --return added to default branch
   end conditions
end calcVal
```

```
to get calcVal x
   conditions
      when x is 1
         return 2
      when x is 2
         return 3
   end conditions
   return 0 -- return added after conditions
end calcVal
```

You might also be missing `return` statements in `if/then` control structures. If the only `return` statement is inside an `if/then` control structure that doesn't have an `else` branch, you must add at least one additional `return` statement following the `if/then` control structure.

3.17. Redeclaring variables in handlers

ToolBook 3.0 does not allow you to declare the same variable twice in the same handler. You should delete duplicate variable declarations. If you're using the `local` statement to clear a local variable, use the `clear` command instead.

3.18. `Null` versus none

ToolBook 3.0 does not allow you to use `null` and `none` interchangeably, as you could in some circumstances in ToolBook 1.5. For example, these two statements are equivalent in ToolBook 1.5:

```
set borderStyle to null
set borderStyle to none
```

The actual value is `none`, but ToolBook 1.5 automatically converts `null` to `none`. In ToolBook 3.0 you must replace such uses of `null` with `none`.

3.19. Changed error messages

A number of error messages in ToolBook 3.0 have changed, so an error in ToolBook 3.0 may return a different error message than the same error in ToolBook 1.5. If your application checks for particular error messages, test carefully to ensure that your error checking still operates correctly.

3.20. Obsolete OpenScript properties, messages, commands, and functions

The following OpenScript terms are obsolete in ToolBook 3.0. Some of the terms still work for backward compatibility with ToolBook 1.5, but you should update your scripts to use the new terms. For more information about the new terms, see the OpenScript Reference Manual and online Help.

Properties

Term	Replaced with	Still works?
`caption` book property	`caption of mainWindow`	yes
`captionShown` book property	set caption of `mainWindow` to `space`	no
`icon` book property	`icon of mainWindow`	yes
`sysMagnification`	`magnification` viewer property	yes
`sysMousePosition`	`mousePosition` viewer property	yes
`sysRuler`	`rulers` viewer property	yes

Messages

Term	Replaced with	Still works?
`align` menu event message	`align<Type>` messages	no
`palettes` menu event message	(use new messages that toggle each individual palette)	no
`printerSetup` menu event message	`printSetup`	no
`toggleStatus` keyboard event message	`statusBar` message	no
`windowMoved` notification message	`moved`	no
`windowShown` notification message	`shown`	no
`windowSized` notification message	`sized`	no

Commands and functions

Term	Replaced with	Still works?
`activate menuItem`	`enable menuItem`	yes
`deactivate menuItem`	`disable menuItem`	yes
`menuState()`	`menuEnabled, menuItemChecked, menuItemEnabled`	yes
`before|after` with `translateWindowMessage`	(forward from called handler)	yes

3.21. Changed OpenScript commands, functions, and messages

Commands

Command	Change
`local <array>`	Declares the dimensions of an array.
`system <array>`	Declares the dimensions of an array that is used as a system variable.

Functions

Function	Change
`round()`	Accepts a parameter to specify the number of decimal places at which to round.
`objectFromPoint()`	Accepts a viewer reference.

Messages

Message	Change
`moved`	Sends a message when a viewer is moved.

Appendix G - Annotated File List

This section is an annotated version of the file list (FILELIST.WRI) included with ToolBook 3.0. Annotations added to Asymetrix's are in italics. In addition, some reformatting has been performed and the full list of clipart files is not included.

Contents

1.0 Distributing Runtime ToolBook

Asymetrix allows you, a licensed owner of ToolBook 3.0, to distribute Runtime ToolBook 3.0 free of charge with your ToolBook application. You are required to ship the minimum set of files needed to run Runtime ToolBook, and you may, at your option, distribute certain additional ToolBook files. The required, linked, and optional files are listed below.

The disk space occupied by these runtime files is between 2 MB and 2.5 MB depending on whether your application uses Paradox Engine and other optional DLLs. This compares with approximately 1MB required under ToolBook 1.5.

	File Name	*Size (bytes)*	*Sub-total*	***Total***
Required:	TB30RUN.EXE	*665920*		
	TB30NET.EXE	*7296*		
	TB30BAS.DLL	*745408*		
	TB30CMP.DLL	*324912*		
	TB30FLT.DLL	*27232*		
	TB30UTL.DLL	*227520*		
	TBLOAD.EXE	*7040*	*2005328*	
			(1958K)	
Linked (optional):	PXENGWIN.DLL	*241853*		
	TB30PDX.DLL	*41824*	*283677*	
			(277K)	
Optional:	TB30DB3.DLL	*92032*		
	TB30DOS.DLL	*18752*		
	TB30DLG.DLL	*81408*		
	TB30WIN.DLL	*39392*	*231584*	***2520589***
			(226K)	*(2462K)*

TBLOAD.EXE is given as a "required" file but this is not strictly true. It is only required if you adopt the Asymetrix start up mechanism by associating .TBK file endings with TBLOAD.EXE. An alternative is to omit this file and just explicitly run TB30RUN.EXE with your .TBK file as the command line parameter. e.g. "`tb30run mybook.tbk`*"*

While PXENGWIN.DLL and TB30PDX.DLL are optional, you must ship both if you ship either one. This requirement is simply a practical matter as these files work together. You may only distribute these files as part of an application developed with and running in ToolBook.

Your users (i.e. those to whom you distribute Runtime ToolBook) are not allowed to re-distribute any of these files, unless they re-distribute them as part of your whole and unmodified ToolBook application.

2.0 Distributing Clip Art Files

You may use and distribute any of the ToolBook clip art files that are installed in the CLIPART sub-directory located in the ToolBook directory. You may, however, only distribute these files as integral part of a ToolBook application. You cannot distribute any clip art files in the form of a clip art library, and your users may not redistribute these files except as part of your whole application (see above).

3.0 File List

3.1 Release Notes

RELNOTES.HLP
RELNOTES.WRI*

* This is a Windows Write file that contains all the information in RELNOTES.HLP. You may print this file if you prefer reviewing the Release Notes on paper.

3.2 ToolBook System Files

TB30.EXE
TB30BAS.DLL
TB30CMP.DLL
TB30CVT.DLL
TB30EDT.DLL
TB30FLT.DLL
TB30NET.EXE
TB30RED.DLL
TB30RUN.EXE
TB30UTL.DLL
TB30XTR.DLL

3.3 ToolBook 3.0 Utility Files

PRINTWND.SBK
SCRWALK.ICO
SCRWALK.TBK
SYSINFO.EXE
TB30.ATS
TBLOAD.EXE
TOOLS30.SBK

3.4 ToolBook Add-On DLLs

PXENGWIN.DLL
TB30DB3.DLL
TB30DLG.DLL
TB30DOS.DLL

TB30PDX.DLL
TB30WIN.DLL

3.5 Add-On DLLs with ToolBook 1.5 names (included for compatibility)

TBKDB3.DLL*
TBKDLG.DLL*
TBKFILE.DLL*
TBKWIN.DLL*

* These files are simply the ToolBook 3.0 DLLs renamed to the ToolBook 1.5 file names. They are included so that your ToolBook 1.5 scripts will execute properly when they link to these DLLs. You should update your scripts to link to the new file names.

3.6 ToolBook 3.0 Resource Editor Files

ASYMCMN.DLL
BITED30.EXE
MENUED30.EXE
PALED30.EXE
ICONED30.EXE
MEDBITST.MMH (WINDOWS\SYTEM directory)
MEDIMPT.MMH (WINDOWS\SYTEM directory)
WINCOMT.DLL
WRKBNCHT.DLL
MEDMANT.DLL

3.7 INI Files

ASYM.INI (WINDOWS directory, do not modify)
TOOLBOOK.INI (WINDOWS directory, do not modify)

3.8 New Features Overview

CHIRP.WAV
FEATURES.ICO
FEATURES.TBK

3.9 Graphic Import Filters

IMCDR9.FLT
IMCDR9.INI
IMCGM9.FLT
IMCGM9.INI
IMDRW9.FLT
IMDRW9.INI
IMDXF9.FLT
IMDXF9.INI
IRGIF9.FLT

IMHG29.FLT
IMHG29.INI
IMHG39.FLT
IMHG39.INI
IMPCT9.FLT
IMPCT9.INI
IMPIC9.FLT
IMPIC9.INI
IMPS_9.FLT
IMPS_9.INI
IRPCX9.FLT
IRTIF9.FLT
ISGDI9.DLL (WINDOWS directory)
ISGDI9.INI (WINDOWS directory)

3.10 Online Help Files

BITED30.HLP
DATABASE.HLP
ICONED30.HLP
MENUED30.HLP
PALED30.HLP
TB30.HLP
WINCONST.HLP

3.11 Sample Applications

ANIMATE.ICO
ANIMATE.TBK
AUTOEDIT.ICO
AUTOEDIT.EXE
COMPANY.NDX
CONTACT.DBF
CONTACT.ICO•
CONTACT.TBK
DBEXCHNG.ICO
DBEXCHNG.TBK
DIALOG.HLP
DIALOG.ICO
DIALOG.TBK
HANDBOOK.ICO
HANDBOOK.TBK
HANDBOOK.TXT
LIBRARY.ICO
LIBRARY.TBK
MAPI.ICO
MAPI.TBK
NAME.NDX
SCRAPBK.ICO
SCRAPBK.TBK

WIDGETS.ICO
WIDGETS.TBK
TAQUIN.BMP
TAQUIN.TBK
TAQUIN.ICO
WHOWHERE.DB
WHOWHERE.MB
WHOWHERE.PX
WHOWHERE.TBK

3.12 Asymetrix Setup Utility

ASYMARCH.DLL
DLGS.DLL
SETUP.EXE
SETUPMGR TBK
SETUPMGR.HLP

3.13 ToolBook 3.0 Tutorial

LEARN30.EXE
LOGO.BMP
TRY.SBK
XSAT.TBK

3.14 Windows 3.1 API Reference (CD-ROM only)

WIN31WH.HLP

4.0 Clip Art

4.1 Clip Art Bitmaps (.BMP)

- *18 256-colour BMP files*
- *11 16-colour optimised BMP files. These 16 color bitmaps use optimized palettes. They will not display correctly under 16 color VGA. See 16OPTMIZ\README.TXT for additional information.*
- *5 NASA 256-colour BMP files.*

4.2 Clip Art Cursors (.CUR)

- *An assortment of 64 cursors.*

4.3 Clip Art Icons (.ICO)

- *arrows and signs* 76
- *computer* 29
- *controller* 7

- *electronics* 50
- *elements* 28
- *industry* 9
- *miscellaneous* 23
- *numbers* 10
- *office* 70
- *traffic* 7

4.4 Clip Art Colour Palettes (.PAL)

FLESH.PAL
METALS.PAL
NATURE.PAL
STANDARD.PAL

Appendix H - Implications for Developers of Upgrading

This appendix reproduces a document which was originally an internal report for the TLTP Economics Consortium. It gives an overview of ToolBook 3.0 and examines strategic, managerial and technical issues involved in upgrading to the (then) new release. In particular, it focuses on the costs and benefits of upgrading given an existing investment in ToolBook 1.5 and highlights areas of potential risk.

For readers of this report currently considering upgrading existing 3.0 software to ToolBook 4.0, the analysis presented here may act as a framework for analysing the unique costs and benefits of doing this for your own project. In fact, many of the issues are entirely independent of any specific release - they are generic issues which arise from the ceaseless superseding of one software tool version by another.

Contents

Note: The original section numbering has been preserved and is independent of the section numbering in the main report.

1. Executive Summary

ToolBook 3 is a substantial upgrade from ToolBook 1.5 in terms of both features and authoring interface.

On purely technical grounds, ToolBook 3 is a highly desirable upgrade for the Economics Consortium and could potentially increase the viable scope of their WinEcon courseware within the available development time.

Upgrading existing courseware to ToolBook 3 from ToolBook 1.5 is technically feasible in terms of system and software upgradability. The continuing royalty-free runtime policy make the upgrade feasible in terms of ToolBook Runtime requirements. The evolution rather than revolution of the OpenScript programming language and the authoring tools make it feasible in terms of compatible skills requirements for authors.

ToolBook 3 has a longer-term development path than ToolBook 1.5 provided that a continual cycle of upgrading to the latest version is acceptable to the project. In doing so, this iterative development path can provide a degree of future-proofing for the courseware against changes in operating systems and hardware platforms over time.

Before taking a decision to upgrade to ToolBook 3 developers should carefully consider the expected payback over the remaining life of the project. Where a project is short-term, or has only a short period remaining, upgrading is unlikely to be cost effective. Projects with a longer time horizon, or with substantial development time remaining are likely to find that the strategic payback outweighs the initial costs. Where projects are adverse to risk, or have very tight development schedules, it is likely that they will choose to audit the performance of ToolBook 3 over the first few months following its release.

2. Managerial and Strategic Issues

2.1. Managerial Issues

From a management perspective the decision to upgrade to ToolBook 3 is a simple matter of weighing the costs against the prospective benefits. Our approach has been to consider the following three issues:

- Cost
- Time
- Quality

These are individually discussed below and the overall managerial impact is expressed by summing the changes in these three variables.

2.1.1. Cost

The direct cost of upgrading is modest (£99 per copy for each of 17 programmers), however the indirect costs are far more substantial. Not least of these costs is the additional staff time that needs to be allocated to upgrading custom features, training and reauthoring. In addition, projects need to assess the opportunity cost of the authoring work which will not be undertaken during this transitional phase. We estimate that the time involved in upgrading and reauthoring an average module is 5 days (assuming each module is approximately 80 pages). In respect of training, experienced authors and programmers we estimate the time required to be 3 days per person.

A further indirect cost is the time required for technical upgrades to system-wide custom tools and features. In the case of the Economics Consortium, significant investment has been made in developing custom system-wide tools and features, and as a result the associated cost of upgrading to ToolBook 3 is relatively high. We estimate the time required for technical upgrading to be 22 days.

Calculating the financial cost of the time required to move the whole project to ToolBook 3 is based on the assumption that programming staff are charged at £70 per day on average. In the case of a project of the size of the Economics Consortium (forty-two staff, of which 17 are directly responsible for programming 26 modules) we calculate the direct staff cost will be £10,640. This figure can be doubled to take account of the opportunity cost of the material that will not be produced during the transitional period of upgrading from ToolBook 1.53 to 3 i.e. £22,880.

Therefore, the total real cost of upgrading to ToolBook 3 is £22,980.

2.1.2. Time

There is obviously a time penalty involved in upgrading to ToolBook 3. This includes both the time required to make technical modifications (e.g. employing resources, viewers and faster execution speed features in our custom system-wide tools) and the time required to modify the authored teaching material. As is detailed above, within the Economics Consortium we estimate that a total of 152 days will be required across the project. This raises the issue of the effect of upgrading to ToolBook 3 on the overall project schedule and the milestones therein. On projects where there is an agreed delivery date this can be a major factor in decision making.

Upgrading to ToolBook 3 may also have an effect on the overall management of the project. Experience suggests that upgrading is used as an explanation for slippage, when in fact the real cause lies elsewhere. These unobserved blockages can mask more serious underlying problems that development teams are encountering.

The effect of these two factors on the project schedule are difficult to calculate with any degree of accuracy. We assume that additional slippage accounted for by the upgrade will be 10% of the total time required to train staff and reauthor material. In the case of the Economics Consortium, we calculate this to be 15 days. At £70 per day, this suggests a cost of £1050.

Set against this are the efficiency gains that upgrading to ToolBook 3 can contribute to the projects development of material. Our initial assessment of ToolBook 3 is that the time required to author a page of our learning material is not that significantly different to that required under ToolBook 1.5. If extensive use is already being made of system wide templates, development time will be decreased by only 10%. This implies an efficiency gain of 13 days. That is, costing at an average of £70, this implies an effective contribution to the overall budget of the project of £910.

Overall the time penalty for upgrading to ToolBook 3 is equivalent to £23,120.

2.1.3. Quality

The issue of quality is more difficult to deal with from a managerial perspective. If we consider only the question of reliability and accuracy of coding, then it is possible to derive some estimate of the impact of upgrading to ToolBook 3. The negative impact on quality will be the additional errors in coding and design that will result from programmers using an authoring system and features that they are less familiar with. In addition, it is inevitable that more errors remain in the code for ToolBook 3 than in release 1.53.

Conversely, ToolBook 3 offers new features (e.g. improved syntax checking, self contained re-usable objects and less need to code tools in C/C++) that add both to the functionality of authored material and its reliability.

Our estimate is that upgrading to ToolBook 3 can provide a net contribution to quality of £2520. This is calculated on the basis that, overall, 180 days are allocated to quality control and testing across the Consortium. We then assume that the new features of ToolBook 3 outweigh the negative effects of lower reliability and additional coding errors, and that these features provide a net reduction in the time required for testing and evaluation of 20%.

2.1.4. Overall Impact

From the above calculations of the effect on cost, time and quality of upgrading to ToolBook 3 we estimate that the overall impact on the budget of the Economics Consortium of upgrading to ToolBook 3 is likely to be of the order of £20,600.

2.2. Strategic Issues

Our assessment of ToolBook 3 has highlighted three key strategic issues:

- Achievement of Mission
- Product Shelf-Life
- Skills Base

These are described below with respect to the Economics Consortium.

2.2.1. Achievement of Mission

The Economics Consortium's mission is to produce an integrated and interactive teaching tool for introductory economics. The improvements and new features available in ToolBook 3 (e.g. viewers) will enable us to more easily achieve this goal.

2.2.2. Product Shelf-Life

By upgrading to ToolBook 3 the Consortium can, to some extent, assist in lengthening the shelf-life of the material being produced by maintaining its link to the latest version of the associated authoring system. Furthermore, upgrading will enhance the maintainability of the material being produced.

2.2.3. Skills Base

In relation to staff development, we believe that it is important that the Consortium upgrades to ToolBook 3, since this will equip our programming staff with more relevant skills. If we accept that not all staff will be retained by the project as we move towards a steady state equilibrium then it will increase their employment opportunities. It is, of course, possible that this might be construed as being a negative point if one assumes that staff with dated skills are less likely to leave the project.

2.3. Assessing the Options

From a purely managerial perspective, the Economics Consortium should not upgrade to ToolBook 3, since the costs outweigh the benefits, at least in the short-term. However, from a strategic perspective there are clear benefits to doing so. Whilst the assumptions made in the calculations may be open to debate, the underlying balance appears robust.

How will projects decide whether to upgrade? From the above discussion it is reasonable to assume that projects which decide to upgrade will be more closely focused on the strategic and long term benefits. Those that remain with ToolBook 1.53 are more likely to be short term projects, and therefore, more closely focused on the managerial costs and benefits.

2.4. The Impact of Risk

The assumptions we have made in assessing the impact of upgrading to ToolBook 3 are clearly subject to modification. For each project the figures calculated in respect of cost, time and quality will vary, as will the relative importance of strategic issues. However, for all projects there needs to be an explicit assessment of risk. This assessment needs to consider to the following questions.

- How certain is it that we can successfully move to ToolBook 3? For example, can we acquire the expertise to train staff?

- How robust do we expect ToolBook 3 to be, and what are the implications? For example, will there be an early release of version 3.1, and what will the costs be?

- How likely are the benefits and the costs to be realised? For example, we may find that the costs we have calculated rise substantially in practice and that the benefits are over stated.

- What are the strategic implications of locking in to the development path that Asymetrix choose? For example, do Asymetrix have a strong commitment to maintaining and developing ToolBook? Does their development model see an important role for the educational market?

An assessment of the risk and associated impact of these questions needs to be taken in to account when deciding whether to upgrade to ToolBook 3. This sensitivity analysis will establish an envelope within which it is beneficial to upgrade, outside of this envelope the benefits are negative. The size of this risk envelope is determined not only by the projects own assessment of the risk involved, but also the level of risk that the project is prepared to accept.

2.5. The Decision to Upgrade

Before taking a decision to upgrade to ToolBook 3 developers should carefully consider the expected payback over the remaining life of the project. Where a project is short-term, or has only a short period remaining, upgrading is unlikely to be cost effective. Projects with a longer time horizon, or with substantial development time remaining are likely to find that the strategic payback outweighs the initial

costs. Where projects are adverse to risk, or have very tight development schedules, it is likely that they will choose to audit the performance of ToolBook 3 over the first few months following its release.

3. Technical Issues

To a large extent, the technical issues in deciding to upgrade to ToolBook 3 underpin the managerial and strategic issues. These technical issues can be broadly divided into the following areas.

- Desirability
- Systems Feasibility
- Software Feasibility
- Long-term Development Path

These areas are discussed below followed by a summary.

3.1. Desirability

ToolBook 3 is a substantial upgrade from 1.5 and offers a host of new features. The definition of a desirable feature will differ for each project and it is necessary to examine the extensive list of new features in the in the context of your project.

For the Economics Consortium, the following features are rated as highly desirable in that they will reduce the number of hours required to implement outstanding features of WinEcon or they will increase the quality of existing courseware.

- Bitmap resources (to replace our own OpenScript implementation of this and attain speed improvements)
- Viewers (to simplify implementing the yet to be written Lecturer's Interface and global Student Tools Popups)
- Paradox Engine DLL (to replace a slow DDE link to Microsoft Access for our tests module)
- Graphics Buttons (to replace our own DLL/OpenScript implementation and attain increased page to page speed)
- copyObject() command (to bypass the Windows Clipboard and increase our runtime resource book speed)
- Right Mouse Button Editing (to replace and enhance our own implementation of this)
- ToolBook 3 is, on purely technical grounds, a highly desirable upgrade for the Economics Consortium and could potentially increase the viable scope of the WinEcon courseware within the available development time.

3.2. Systems Feasibility

ToolBook 3 is a substantial upgrade from ToolBook 1.5 in terms of features and, indirectly, in terms of system requirements. For an upgrade to be feasible ToolBook 3 must run adequately under both student and develop systems. The system requirements of version 3 can be divided into three sections.

- developer's hardware requirements
- student's hardware requirements
- software requirements

These are presented below followed by an analysis of changes from 1.5 and their impact on the project and an assessment of the feasibility of upgrading in terms of systems requirements.

3.2.1. Developer's Hardware Requirements

Minimum Development Machine

- 20MHz 30386 SX processor or higher (30386 or higher stated on the box)
- 8-24MB of free hard disk space (depending on installation options)
- At least 4MB RAM (8MB recommended in User Manual, 6MB recommended on box)
- 1.44MB (3.5") disk drive
- Mouse or pointing device
- Graphics adapter card (VGA, SuperVGA, or other Windows compatible card)
- Sound card (optional)
- CD (optional)

ToolBook 3 will run on the hardware described above. However, we consider the editing speed of the base level machine to be too low for time efficient editing of large scripts and complex screens. This was true for ToolBook 1.5 but the extra functionality of 3.0 coupled with the more graphically demanding author's palettes, toolbars and popups further lengthen the edit/run cycle.

The increased number of tool palettes, toolbars and a status bar make the requirement for a screen with a resolution higher than the target screen's resolution more important if these new editing tools are not to obscure the main window. It is not particularly easy to work with all these objects on a laptop where the resolution is typically 640x480 - the same as the student machine for the Economics Consortium; in practice on a 640x480, it is easier to switch off most of the palettes and use the menu equivalents instead.

Under ToolBook 1.5, the Economics Consortium recommended a system based around at least a 25MHz 486 SX as the minimum development machine. However, as a consequence of the changes in hardware requirements, we recommend the following configuration as being the minimum programmer-efficient development machine.

Recommended Development Machine

- 66MHz 40386 processor or higher
- 24MB of free hard disk space (plus 9MB extra if WIN31WH.HLP Windows API help is loaded)
- 8MB RAM or more
- 1.44MB (3.5") disk drive
- Mouse or pointing device
- SuperVGA running in at least 800x600 with the required number of colours
- Sound card (optional)
- CD (optional)

As a consequence our initial purchasing recommendations to Economics Consortium developers, our existing development hardware base is mostly at this configuration and so no further hardware cost is associated with running ToolBook 3.

3.2.2. Student's Hardware Requirements

- Minimum Student Machine
- 20MHz 30386 SX processor or higher
- 2.5MB of free hard disk space for ToolBook 3 runtimes (plus space for courseware)
- At least 4MB RAM (8MB recommended in User Manual, 6MB recommended on box)
- 1.44MB (3.5") disk drive
- Mouse or pointing device
- VGA or higher
- Sound card (optional)
- CD (optional)

Changes in this requirement have the largest impact on the project and are in an area which is least under its control. At the start of the TLTP project, the Economics Consortium took a census of hardware available for teaching undergraduates and of purchasing plans for the following financial year. This revealed a large installed hardware base of 386 SX machines and a slow upgrade trend towards 486 machines. Consequently, we are required to support as low as 16MHz 386 SX machines. Although this is below the floor of 20MHz specified by Asymetrix, it is our assessment that operational speed improvements under ToolBook 3 will give the same or better performance than the same WinEcon courseware under ToolBook 1.5 (largely due to the introduction of system support for resources replacing our own OpenScript implementation of resources). Indeed, in the case of WinEcon, there will be a significant speed increase on lower specification student machines - due to new features rather than any improvements in the graphical speed of 3.0.

There will be a reduction in the file size of the courseware modules as a result of employing bitmap resources (i.e. only one copy of a graphic is stored regardless of how many times it occurs in the book) of around 80% in the case of WinEcon which makes heavy use of graphics buttons and clip art. This substantially reduces the hard disk storage required on student machines - particularly where all 26 modules are installed. The impact of this on the project is that departments need commit less of their storage capacity to WinEcon and the opportunity cost of adopting it for student use is reduced, as the capacity to hold other software is reduced by less.

The Economics Consortium has undertaken to support networked student machines where they exist. We generally do this by downloading ToolBook runtime system files and one of WinEcon's 26 modules from the network's file server onto the student's machine - the software then runs without any network traffic other than access to a tests database and storing of student marks and so it runs at full speed. The size of the runtime system files under ToolBook 3 is more than twice that of 1.5 and so the start-up time for students will be increased. Offset against that is the reduction in the size of courseware module books resulting from the use of resources. Considering that graphics are the main storage overhead in books, this is likely to compensate for the increase in runtime system file size and therefore give approximately the same network start-up delay under 3.0 as under 1.5.

Unfortunately, there are a number of institutions who are committed to diskless workstations (i.e. PCs with no built-in hard disk) and these systems can cause substantial network traffic. The impact of larger EXE and DLLs on these systems will only become fully apparent after trialing. However, there appears to be no significant change in impact of upgrading to ToolBook 3 in terms of the size of RAM disk required where diskless machines have these and download software to the RAM disk. The Consortium in adopting ToolBook in the first place, adopted a strategy which accepts the risk of low performance on diskless workstations.

3.2.3. Software Requirements

These are the same for both developers and students running ToolBook 3.

- Microsoft MS-DOS 3.1 or higher
- Microsoft Windows 3.1 or higher

This is the de-facto industry and UK educational standard on PC hardware.

In summary, it is feasible for the Economics Consortium to upgrade to ToolBook 3 in terms of systems requirements although there remains a grey area relating to having undertaken no trials on diskless workstations. On balance, considering the relative rarity of such diskless machines in UK economics departments, it is a reasonable risk to take.

3.3. Software Feasibility

In addition to being systems feasible, the upgrade to ToolBook 3 must also be software feasible. Software feasibility depends upon the following issues.

- Upgradability of Existing Code
- Runtime ToolBook
- Skills Requirement

These are analysed below and a summary presented.

3.3.1. Upgradability of Existing Code

ToolBook 3 is NOT binary compatible with ToolBook 1.5. To upgrade an existing ToolBook book requires you to run an upgrade utility included with ToolBook 3. The resultant new book can not then be loaded back into ToolBook 1.5 and so if a parallel development approach to upgrading is planned, it will be necessary to maintain both 1.5 and 3.0 versions of your books. Within the Economics Consortium we judged parallel development impractical to configuration manage with such a large volume of courseware being developed by a geographically distributed team of programmers. We therefore approached upgrading as a permanent commitment to the new version and as an irreversible step. A high degree of certainty about the feasibility of upgrading existing code is required because of this.

Experimentation with upgrading a variety of non-TLTP ToolBook 1.5 books has shown that in most cases the upgrade is fairly straight forward and often requires no manual editing to make the new version work. However, books with advanced OpenScript techniques - particularly those involving DLLs - often report a successful upgrade only to then cause a Windows General Protection fault when run. In the case of the Economics Consortium, this has made it impossible to upgrade a single module to-date. This is due to the fact that all of our courseware is built around a template which employs several advanced Windows and ToolBook programming techniques. All the problems are associated with making our custom tools DLLs written in C to work under the new system (largely due to a change in the meaning of “” and NULL parameters). Fortunately, none of the problems which have arisen in the template have taken very long to solve and it is reasonable to assume that once the template is working, the courseware modules themselves will take relatively little effort to upgrade as none of them directly employs complex programming techniques - all complex programming is encapsulated in the template.

Once the template is upgraded to run under 3.0, it is then possible to replace a number of our custom tools written in OpenScript with the in-built system ones supported by ToolBook 3 (e.g. graphics

buttons, bitmap resources and editing enhancements). The cost of doing this is restricted to the cost of taking advantage of these new features within the template itself; no substantial changes to the existing courseware modules is required to take advantage of these new features. Such an upgrade would be far more extensive in its scope for projects without a core template and would reduce the feasibility of fully embracing the new features during the upgrade process.

3.3.2. Runtime ToolBook

Although it is now possible to turn your book into an executable (.EXE) file, ToolBook 3 still requires a set of runtime files to be installed on the target machine as well. As with ToolBook 1.5, these may be distributed royalty-free as part of your ToolBook application and this continues to be a major advantage over other authoring systems charging a royalty on each copy sold for projects intending to generate income through sales. (See the annotated file list in the appendices of this document for further information of the runtime files.)

Under ToolBook 1.5 it was necessary to purchase an installation programme from Asymetrix in order to allow users to install these runtime files and your courseware books. The whole process of building disks and splitting large files was extremely error prone and time consuming; it was also very difficult to entirely automate. Under ToolBook 3, a utility is included which automates the building of distribution disks, compression and splitting of files, creation of Program Manager Groups and running of other programmes after installation.

3.3.3. Skills Requirement

The Economics Consortium has a not inconsiderable investment in training programmers in OpenScript programming for ToolBook 1.5. For the adoption of version 3.0 to be feasible, there must be a high degree of transferability of authoring/programming skills to the new version of OpenScript. Similarly, the authoring interface for screen design must remain fairly compatible with version ToolBook 1.5 if the amount of retraining is not to become prohibitively expensive.

OpenScript 3 is a superset of OpenScript 1.5 and requires minimal training to employ the majority of the new features. Despite this, some features (eg. notification, widget construction, resource object management and palette manipulation) of the new version will remain unintelligable to all but the more experienced computing professionals. However, as long as inexperienced programmers are advised to avoid these features, this is not a problem. For programmers at the other end of the experience spectrum, these advanced features are a welcome addition and enable many functions to be performed in OpenScript which previously required DLL usage or creation. The level of Windows programming experience required to perform complex authoring or to create templates is now lower than before.

The interface has changed substantially from the original 1.5 version but is still recognisable to our programmers - not least because the editing enhancements implemented in our custom system-wide tools are now all standard features under ToolBook 3.

Upgrading to ToolBook 3 is technically feasible in terms of software upgradability. The continuing royalty-free runtime policy and new installation programme make it feasible in terms of the ToolBook Runtime and the evolution rather than revolution of OpenScript and the authoring tools make it feasible in terms of skill requirements.

3.4. Long-term Development Path

For the Economics Consortium's WinEcon software to have a long-term development path it must adequately address the following three areas.

- Portability
- Upgradeability
- Maintainability

The impact of upgrading to ToolBook 3 on these areas is discussed below.

3.4.1. Portability

Software portability is a critical issue for long term viability of the product of the Economics Consortium or of any other project; if a product ceases to match the hardware and operating system user base then it ceases to be of any practical use. The portability of courseware to new hardware and software bypasses the requirement to reauthor material for each new delivery environment. Failing this, courseware must be reauthored to run under another authoring system in the new environment and multiple versions of the software may need to be maintained in parallel.

As with ToolBook 1.5, Asymetrix has given no indication that ToolBook 3 will be converted to run under other operating environments such as Mac or Unix. Therefore, ToolBook 3 is, and seems likely to remain, a PC specific product in the short term. For the Economics Consortium, its customer base of economics departments has almost entirely PC-based student machines and this is extremely unlikely to change in the short-term to mid-term. The long term, on the other hand, can never be predicted accurately and it is not outside the realms of possibility that UK economics departments will move to non-PC hardware at some point in the future. In such a case, portability would rapidly rise up the Consortium's list of priorities. Consequently, it would be unwise to commit to a new authoring system now if there were a high risk of being permanently locked into a single delivery environment.

However, the hardware base is only part of the portability equation; the other major component is the operating system running on the hardware. Microsoft Windows 3.1 running over the MS-DOS operating system is currently the de-facto industry and educational standard on PC-based hardware. Microsoft has announced its intention to replace Windows 3.1 with Windows 4 - a version which will no longer require MS-DOS as the underlying system on PC compatibles. Also, in the longer term, Microsoft intend to replace Windows 4 with a non-PC specific operating system which may result in an increased diversion of hardware platforms educational software developers need to support.

Projects with a long term planning horizon must then place Asymetrix's product development strategy into the context of this evolving operating system and hardware scenario. Asymetrix have repeatedly stressed a long term commitment to ToolBook as its flagship product and, at present*, there seems little likelihood of this changing in the foreseeable future. Therefore, it seems reasonable to assume that Asymetrix will continue to update ToolBook to work under and take advantage of new operating systems and hardware.

Therein lies the future-proofing of any Windows-based authoring tool such as ToolBook. By developers continually adopting the latest release of ToolBook running under the latest version of Windows, they will have a development path through to new hardware. It is reasonable to assume that ToolBook 3 is intrinsically more portable than 1.5 because it is the latest version - Asymetrix are unlikely to convert ToolBook 1.5 to run under Windows 4.

* Asymetrix have recently released committed a lot of resources to their new InfoModeler database tool with “several large organisations poised to roll-out InfoModeller, including Marconi, Federal Express and the Evening Standard”. “Asymetrix now funds a research lab at the University of Queensland” looking at the FORML database object role modelling (sources: recent issues of Computing). The lure of the database market could, in the mid to long-term, push ToolBook into second place within Asymetrix - unless the two products were merged into one. Also, companies (eg. Borland) have run into serious financial problems in trying to move from an established market into another.

3.4.2. Upgradability

ToolBook 1.5 is effectively a frozen product and in the long term this will result in it being an unusable product due to the underlying operating environment changing over time. In the unbelievably unlikely event that it would continue to work entirely unchanged under Windows 4, 5, etc. it is certain that Asymetrix will not be modifying it to take advantage of the new features users will expect to see supported under these systems (eg. OLE-2 which, when added to ToolBook 3, will enable componentware tool add-ons similar to Visual Basic’s VBX custom controls).

Although it is possible to argue that there is no need to upgrade to the latest release until something does go wrong, there is no guarantee that it will be possible to upgrade from ToolBook 1.5 to, say, ToolBook 4 whereas it is fairly safe to assume that it will always be possible to upgrade from the previous version to the current one.

3.4.3. Maintainability

The release of the first version of a piece of courseware is only the beginning of a cycle of revisions and maintenance which will fix faults and keep the content up-to-date. This process must continue as long as the courseware is in wide use; indeed it will not remain in wide use unless this process is carried out.
How do ToolBook 1.5 and ToolBook 3 differ in this regard? From a technological standpoint there is very little difference between the two versions which would make it easier to maintain version 3 books than version 1.5 books. Improvements in the development environment, whilst leading to easier editing and debugging, are also accompanied by a multitude of extra features for a programmer to master - therefore giving no substantial net improvement in maintainability.

Despite there being little technological difference in terms of maintainability, there is an important strategic difference: it will become increasingly more difficult for projects to recruit staff with expertise in ToolBook 1.5 as time progresses and as ToolBook 3 inevitably replaces 1.5 as the normal version for educational software development. If the project is to be maintainable in the long-term, it must have access to programmers with the required skills and this, by itself, is a strong argument in favour of upgrading.

ToolBook 3 has a longer-term development path than ToolBook 1.5 provided that a continual cycle of upgrading to the latest version is adopted. In doing so, this iterative development path provides a degree of future-proofing for the courseware.

3.5. Summary of Technical Issues

On technical grounds, the Economics Consortium considers that ToolBook 3 is a highly desirable upgrade which is feasible in terms of systems, software and a long-term development path. From a strategic point of view, the long-term development path dominates the technical issues and, given the feasibility of upgrading, is a compelling argument in itself for adopting ToolBook 3.

Whilst there is little doubt that most projects will find ToolBook 3 a desirable upgrade, they must individually assess the feasibility and long-term development path for their own specific circumstances before committing to the upgrade. Projects with little or no existing material to upgrade are highly likely to begin development in ToolBook 3 whereas projects with a substantial legacy of existing courseware must carefully weigh-up the feature advantages of upgrading against the impact on their schedule and their longer term plans.

Appendix I - ToolBook 4.0 Product Information

This appendix is an abridged version of the official Asymetrix press release and product information for Multimedia ToolBook 4.0 and Multimedia ToolBook 4.0 CBT Edition. The descriptions and opinions given here are those of Asymetrix itself, not of the authors of this report.

Contents

1.0 Asymetrix Press Release

Bellevue, Washington—December 5, 1995 -- Today Asymetrix Corporation announced the newest versions of the ToolBook family, the world's easiest multimedia authoring system. The new versions include Multimedia ToolBook 4.0, now shipping, and the Computer Based Training (CBT) Edition shipping January 22, 1996. With the new releases, Asymetrix extends their target audience further into mainstream corporate and educational markets. The latest versions of the ToolBook family stress ease-of-use while keeping the power that multimedia publishers expect. One of the most versatile new features is drag-and-drop support for Visual Basic Controls enabling users to instantly leverage the thousands of Visual Basic applications without programming.

Asymetrix disclosed that future versions of the ToolBook line will further extend their reach to people who are using word processors today, but will be publishing media rich information tomorrow. Asymetrix also described how it is developing ground breaking new technology that will use the Internet to revolutionise the burgeoning Distance Learning Market.

> *"From individual entrepreneurs to large corporations, the ToolBook family is designed to give people with ideas the ability to deliver their information with the impact of multimedia," said Jim Billmaier, President and CEO of Asymetrix Corporation. "If you have information you need to capture, organise, and disseminate, whether you're trying to effectively market a product, train in-field personnel or publish a CD-title, you need Multimedia ToolBook. It's not just for the multimedia elite, such as professional producers. It's for anyone who's got ideas that need to be powerfully expressed."*

2.0 Multimedia ToolBook 4.0

The newest version, Multimedia ToolBook 4.0, has extended ToolBook's intuitive creation process by adding special ease-of-use features and powerful automated utilities. In addition, Asymetrix redesigned the ToolBook structure to utilise an extensible plug-in architecture. This allows users quick access to specific functionality they need today such as Visual Basic Control support. Future capabilities will include animation and HTML interfaces. Performance and quality were a major focus of the development with the result being up to 50% increase in application run-time performance. Specific features include:

- New Book Specialist automates the common authoring tasks and creates an entire application framework by answering a few simple questions.
- New Visual Basic Control support works through a visual drag-and-drop interface to instantly give user's access to thousands of existing Visual Basic Controls available on the market. These include the commercially available applications for spreadsheets, database access and financial analysis or custom in-house applications.
- New Object Browser and Property editor allow quick and easy visualisation and modification of any or all of a user's application contents.
- Palette Optimiser automatically adjusts an application's visual media creating a polished and unified colour scheme.

- Script remover utility removes and protects a user's valuable custom developed programs before distribution of finished titles.
- Shared scripts automatically turn custom scripts into objects. This allows multimedia authors to reuse common functionality, increasing their productivity and insuring application reliability.
- Windows 95 controls lets the user create applications that automatically adjust their interface to any Windows platform.
- New Internet features instantly launch Web pages enabling integration with browsers such as Netscape Navigator to build applications that automatically access the vast information resources of the WWW.

3.0 Multimedia ToolBook 4.0 CBT Edition

The Computer Based Training (CBT) Edition of Multimedia ToolBook 4.0 provides a cost effective solution for educators, government agencies, and businesses who need to train personnel and students. Conventional CBT creation can be extremely expensive to develop, design, and maintain. Multimedia ToolBook 4.0 CBT Edition provides a versatile and affordable system that allows teachers and instructional designers to create computer-based training, performance support systems, and other interactive learning applications. In addition to all of the features of Multimedia ToolBook 4.0, the CBT Edition includes:

- Enhanced Course Management System now includes Bookmarking that automatically tracks a student's progress giving the instructor additional input on student's advancement.
- CBT Specialists allow an instructor to create an entire CBT application framework by answering a few simple questions.
- Updated Catalogue of over 200 pre-scripted, drag-and-drop widgets that automatically add interactivity and test student comprehension of material.
- Professionally Designed Templates increase authors' productivity by allowing them to choose from hundreds of templates. Instructors focus on content, not layout.

4.0 Future Directions

"This is just the beginning of the new ToolBook line," said Raine Bergstrom, Product Manager for Authoring Tools. "We are releasing new versions of our products that focus on the needs of novice computer users wishing to publish more content rich information than they can with today's static information tools. We are also creating versions which will enable individuals and organisations to publish and use CBT content on the World Wide Web (WWW)." The company promises new versions of their products will be even easier-to-use, lower in cost, and have targeted functionality to meet specific user requirements. With new products in the ToolBook line, people will be able to do "Collective Authoring," a technique where teams around the world will contribute to the applications development process. This will enable corporations to more effectively capture and harness the collective knowledge of its key people no matter where they reside. Asymetrix's road map also showed products that deliver interactive information applications over the WWW. "The new Internet products will enable tomorrow's 'Brickless and Mortarless' learning institutions and give corporations the opportunity to re-train employees, even in remote sites," said Billmaier.

5.0 Product Description

Asymetrix Multimedia ToolBook is the leading multimedia authoring system for Windows.

Multimedia ToolBook gives you all the tools you need to create graphical Windows applications that combine user interactivity, powerful data and text manipulation, along with multimedia graphics, animation, audio and video.

Multimedia ToolBook is the tool of choice by leading corporations, government agencies, health care and educational institutions for creating a wide variety of computer based training (CBT), simulations, interactive learning courseware, information kiosks, and many more multimedia-rich applications.

5.1 Multimedia Authoring Ease of Use Redefined

Multimedia ToolBook sets the standard for ease of use. It allows you to quickly and easily develop applications with menu bars, dialog boxes, combo boxes, drag-and-drop, graphic buttons, radio buttons, check boxes, 2D and 3D buttons and a host of other Windows objects.

5.2 Windows 95, Windows 3.1 and Windows NT Compatibility

Windows 95 controls and interface objects are supported in Multimedia ToolBook. Windows controls display differently depending on whether an application is running on Windows 3.x or Windows 95. 3D buttons and inset/raised fields automatically change their appearance to look like the standard controls for that platform.

The exceptional user interface of Multimedia ToolBook along with an array of powerful editing tools provides tremendous leverage to your development efforts in time savings and faster learning.

Multimedia ToolBook's Windows interface uses many familiar buttons and icons to perform tasks like creating a button or text box or creating a drawing or multimedia object. Adding, positioning, and resizing objects in your application is as easy as drag-and-drop. If you are familiar with other Windows software, you will find it easy to learn how to quickly create dazzling interactive multimedia applications using Multimedia ToolBook.

A variety of editing tools and palettes are available that can be opened at the touch of a button.

These tools set Multimedia ToolBook apart from other authoring packages making multimedia magic accessible to all levels of corporate, government, and educational users.

5.3 Corporate Uses

- Computer Based Training
- Product Demonstration Software
- Information Kiosks
- Professional Presentations
- Sales and Marketing Information Management
- Customer Surveys

- Market Research Programs
- Manufacturing, Factory Simulation, and Quality Assurance

5.4 Educational, Healthcare, and Government Uses

- Computer Based Training
- Performance Testing, Evaluation and Support
- Educational Multimedia Titles
- Simulations, Battlefield and Educational Scenarios
- Information Kiosks

5.5 Easily Integrate Video, Graphics and Sound

Users expect sophisticated multimedia applications that combine text, graphics, animation, audio, and video. You can deliver them with Multimedia ToolBook. Create path-based animations without scripting. Include powerful video clips, with full Video for Windows run-time support.

Working with multimedia file types is easy, too; you can play both graphics and multimedia in a single window, without rebuilding the window for each media type. Resize and reposition the window at any time with simple, scriptless drag-and-drop.

Multimedia ToolBook gives you built-in sound mixing controls to allow control over sound volumes during clip playback. The mixing controls allow you to effortlessly smooth variations in clips from different sources, and synchronise the sound with time-based video or animation.

5.6 More Built-in Productivity Features for Faster Development

Multimedia ToolBook has extensive features to get you started and make you more productive faster than any other authoring system. The payoff is faster delivery of your application and quicker return on your development efforts.

- Use Auto-Script to choose from a library of predefined or custom object scripts.
- Develop and test applications quickly with built-in script editor, syntax checker, and debugger.
- Simply demonstrate what you want a script to do and the Script Recorder will build it for you.
- Execute OpenScript statements interactively from the Command Window for faster application development.
- Object Browser (new in 4.0) displays objects in your book hierarchically, similar to an outline. You can quickly select, edit, or delete objects.
- Shared Scripts (new in 4.0) can be used to assign the same behaviour to multiple objects. Shared scripts save you time and simplify the maintenance of your applications.
- Property Editor (new in 4.0) allows you to quickly browse, view and edit object properties.
- Automated Book Specialist creates entire application framework and automates common tasks for both
- Easily integrate controls to launch Internet browsers and navigate to URLs from your ToolBook application.

5.7 Editing Tools, Windows, and Palettes

- Palette Optimizer (new in 4.0)
- Property Editor (new in 4.0)
- Object Browser (new in 4.0)
- Bitmap Editor
- Line Drawing Palette
- Media Clip Manager
- Polygon Shapes Editor
- Colour and Pattern Palettes
- Menu Bar Editor
- Debugger
- Rulers and Grid Controls
- Script Editor
- Icon/Cursor Editor
- Media Palette Editor
- Object Tool Palette

5.8 New Palette Optimisation Feature for Accurate Colour Rendering

Many multimedia applications experience color shifts when using multiple graphics sources.

The new Palette Optimiser utility automatically creates a common palette that contains a blend of the dominant colours from all graphics and video to give your transitions a smooth and polished appearance.

5.9 Powerful Windows Application Interoperability

Visual Basic controls (VBX) is a new feature in Multimedia ToolBook 4.0 that lets you extend your application and import standard VBX Window objects - such as enhanced fields and list boxes, clocks and timers, gauges, grid controls, and many others. VBX controls are pre-scripted, enabling you to incorporate sophisticated capabilities into your application with little or no programming.

Take advantage of DDE, OLE, and DLL support to seamlessly integrate data and functionality from your favourite Windows databases and applications, using Multimedia ToolBook.

5.10 Easy to Use OpenScript Programming Language

OpenScript is a full-featured programming language that includes commands to accomplish a wide variety of tasks, from creating and managing new objects to linking functions in Windows DLLs. OpenScript is easy to use because of its English-like syntax, its wide range of commands, and its object-oriented nature. Using Multimedia ToolBook and OpenScript, you can program sophisticated Windows applications with a fraction of the time and effort needed to create similar applications in C, C++, or Visual Basic.

Put Hyper Text and Database Power into your Multimedia Applications.

Most corporate, government, and educational multimedia applications require considerable text and data manipulation. Support for rich-text format (RTF) lets you

easily import fully formatted text files from any word processor. Create Hotwords, hyperlinks, and achieve dramatic text effects using colour, inline graphics, superscripts and subscripts. Multimedia ToolBook supports the industry's most comprehensive text search engine. It includes full-text search and retrieval for word and Boolean searches and allows searches on both single words and combinations of words and phrases within user-specified parameters. You can use standard and customisable search interfaces. Also included is a full-text spelling checker, so your applications are letter-perfect.

Multimedia ToolBook has more database capability than any other multimedia authoring system. You can easily integrate an application into a wide variety of databases, from product specifications to customer histories and support data. It's the superior choice for computer based training, kiosk and other business support applications where extensive data is collected or manipulated. Multimedia ToolBook includes built-in Paradox and dBASE III database engines. And if you need more extensive connectivity with other databases, the optional ToolBook Database Connection allows easy access and integration with all ODBC compliant databases from your application.

6.0 Technical Highlights

6.1 New in Version 4.0

- Up to 50% faster run-time performance than version 3.0.
- Create and distribute on any Windows platform with Windows 95 Controls.
- Extend your application with windowed Visual Basic Controls (VBX).
- Make your application truly object oriented using shared scripts to assign common behaviour to multiple objects.
- Palette Optimiser eliminates colour shifts between different multimedia objects in your application.
- Automated Book Specialist simplifies start-up by automating the new book creation process.
- Object Browser shows all objects in your application in proper hierarchy, so you can quickly view and modify them on the fly. A real time-saver.
- Script Remover utility removes the text of your application's scripts, but leaves the executable script code intact. This prevents anyone from editing or viewing your scripts and reduces the file size of your final application.

6.2 Existing Features

- Easy drag-and-drop to position and size visual media, including bitmaps, video, and animations.
- Create customised Windows interfaces, complete with dialog boxes, pop-up menus, scrolling text windows, and custom palettes with only a few clicks of the mouse.
- Easy to understand Book metaphor that makes it easy for anyone to quickly learn and get maximum productivity.
- Integrate .WAV and .MIDI sound files into your applications with sound control dialog boxes.
- Create applications without programming using the Auto-Script library.
- Take advantage of powerful text manipulation features including rich-text format support, spelling checker, embedded TrueType fonts, and full-text search and retrieval.

- Create dialog boxes, palettes, tool bars, and application parent-child windows complete with text scrolling.
- Add drag-and-drop interaction, button graphics, and a wide variety of other unique features to make your applications stand out.
- Powerful debugger with auto-debugging feature.
- Easy to learn and use OpenScript programming language for developing custom application features and enhancements.
- OpenScript manual completely available in extensive on-line help system.
- Editing resources including audio, icons, cursors, bitmaps, menus, and colour palettes.
- Build powerful multimedia database applications with built-in Paradox and dBASEIII database engines.
- Simplify the distribution of your applications with the Media Packager, Setup Utility, and royalty free run-time engine.

7.0 System Requirements

- Microsoft Windows 3.1, Windows NT 3.5, or Windows 95 or higher
- A Windows-compatible computer with a 20MHz 80386 SX or higher processor
- A hard disk with at least 17MB of free disk space; installation may require more space depending on the options you choose during setup * At least 8MB of random-access memory (RAM); 12MB or more is recommended.
- A graphics adapter card (VGA, Super VGA, or other Windows-compatible card)
- A Windows-compatible mouse or other pointing device
- CD-ROM drive
- Sound card (optional)