



Australian Journal of
Educational Technology

Developing and distributing audio applications with Visual Basic and ToolBook

Larry Nelson
Curtin University of Technology

Visual Basic and ToolBook are powerful environments for developing software, and both support multimedia applications. This paper compares the capabilities of these systems for developing interactive audio applications, and finds that, as always, there are trade-offs to consider. ToolBook provides an environment more akin to that of an authoring system, but is not as parsimonious as Visual Basic when it comes to preparing executable files for distribution to users.

One of the exciting promises of multimedia is that it can allow us to develop substitutes for old media which are easier to use, and in certain cases, of better quality.

Take language learning aids, for example. An important, ever-present aspect of getting up to speed in a new language involves vocabulary building and pronunciation practice. To meet these goals language learners have often turned to flip cards and audio tapes.

Flip cards are usually small, being not much larger than a standard business card. They have a word or phrase printed on one side, in one language, and have the same word or phrase printed on the other side, in another language. They're convenient and easy to carry around, but they provide absolutely no help when it comes to pronunciation.

Enter audio tapes. They are also convenient, and easy to carry even when we have to lug around a playback unit as well. But they have no automatically synchronised textual display of the words or phrases being pronounced on the tape.

The marriage of these media has been juggled by users for years. We get out our pack of flip cards, the audio tape which corresponds to the vocabulary items on the flip cards, plug the tape into the playback unit, and away we go. The tape rolls through the cards one by one, and we

listen diligently through our head phones as we flip through the cards in synchronisation.

But there are problems. What happens when we want to replay a word? Rewind the tape? Yes, but this is an inexact process - we'll wind too far, or not far enough. What happens when we have mastered all but a small core of the flip cards? We can extract this core of cards from the pack, and run with a sub-pack But we can't do the same with the tape.

Enter multimedia. We'll use software to develop 'stacks' of flip cards. On clicking a card, the user will see it flip to its other side, and will simultaneously hear a native speaker as she, or he, pronounces the word or phrase found on the 'foreign language' side of the card. The card may be played again, or dismissed in favour of the next one. Cards which have been mastered can be ticked, and shuffled off to an inactive stack.

Sounds good, and in fact it is. But savvy readers will ask about the resources needed to put together such a blend of media, and about the equipment required for playback. Our DEET [1] funded 'Talk Project' at Curtin University (Nelson, 1993, 1994) has looked into such questions. We have standardised on a Visual Basic-based development system which lets us put out a new list of words or phrases in about a day, with everything fitting onto a single 1.44 megabyte floppy diskette. Along the way we looked at an alternative to Visual Basic, ToolBook. Below I mention some of the matters to consider when deciding which of these two systems to apply [2].


Pasting MCI audio controls on a Visual Basic form

The Visual Basic programming environment is now into its third edition (Microsoft, 1993), but most of our multimedia developmental work has been done with Version 2.0. The procedures used for playing and recording audio files are the same in Versions 2.0 and 3.0.

There are two methods which may be used in Visual Basic for developing applications which play back pre-recorded audio files. One involves the use of a Visual Basic MCI [3] control; the other sets up a Visual Basic function which asks a standard Windows DLL [4] for access to playback services.

Using an MCI control under Visual Basic requires the presence of a utility file known as MCI.VBX. Visual Basic eXtension (VBX) files first appeared with Version 2.0, where they were distributed as part of a professional tool kit, and only partially supported by Microsoft. In the third edition of Visual Basic, Microsoft released both standard and professional versions; the latter includes a library of VBX files which are fully supported by Microsoft.

Visual Basic has a floating tool bar from which users select and paste the controls they want to use. Developers have the ability to add and subtract icons from these bars so that they are tailored to local needs.

Selecting the MCI icon from a Visual Basic tool bar  and then pasting to a Visual Basic form (corresponding to pasting to a "page" in ToolBook) produces the following control on screen:



The nine buttons on this control typify the actions we commonly associate with media device control panels: previous, next, play, pause, back, step, stop, record, and eject. The number of these buttons which appear in design mode can be set by the user, and the entire control can be hidden at run time if developers prefer to design their own media interface.

Once an MCI control has been pasted to a Visual Basic form, code such as the following can be used to play an audio clip:

```
Let MMControl.DeviceType = "WaveAudio"
Let MMControl.FileName="c:\library\indo\male\selamat1.wav"
MMControl.Command = "Open"
MMControl.Command = "Play"
```

Most of our 'Talk' lessons provide recording facilities as well as playback; these allow users to record their own voice, and to aurally compare their pronunciation with that of a native speaker's. The Visual Basic instruction which directs the MCI control to open up a microphone for users to speak into is:

```
MMControl.Command = "Record"
```

Many times a developer will want her or his software to suspend execution until a media clip has completed playing. For Visual Basic MCI controls, the appropriate command is "Wait".

The use of an MCI control in Visual Basic is an easy and powerful way to develop audio applications. A potential disadvantage relates to having to include MCI.VBX in the suite of run-time files which are distributed to users; this potential limitation is further discussed later.

MMSystem.DLL as an alternative to using MCI.VBX

Waveform sound files, ie. audio clips formatted to WAV file conventions (Microsoft, 1991), may be played by making a call to the `sndPlaySound` API [5] function found in the MMSystem DLL. This library is automatically installed when Windows is first set up on a machine.

Adequate instructions for using this function may be found in one of the Microsoft collections of technical articles [6]. When the function has been properly defined in a global Visual Basic module, or in a form's general declarations section, statements such as the following may then be employed:

```
Let SoundFileName$ = "c:\library\indo\male\selamat1.wav"
wFlags% = SND_SYNC
x% = sndPlaySound(SoundFileName$, wFlags%)
```

These lines of code will play the nominated file, and pause the execution of any following statements until the sound has finished playing.

In contrast to controlling sound playback via the MCI Visual Basic control, the `sndPlaySound` function does not require that an extra file, `MCI.VBX`, be distributed with run-time modules. We have used `sndPlaySound` in a number of projects without problem; its disadvantage is that there is no similar way of recording sound - the `MMSYSTEM` DLL does not yet have a `sndRecordSound` function.

Using MCI audio controls in ToolBook

ToolBook (Asymetrix, 1994) is a HyperCard-like system which initially appeared with Version 3.0 of Windows, where a personal information manager with a TBK extension was included as a free sample (`DayBook.tbk`).

The multimedia extensions to ToolBook appeared in 1991. They make access to Windows MCI controls a bit more straightforward than they are under Visual Basic. Whereas Visual Basic requires an 'MMControl' to be pasted onto a form, Multimedia ToolBook's MCI support comes ready to roll. One can, for example, slap down an ordinary ToolBook button, and then write a 'buttonDown handler' which might include the following code:

```
Set SoundFileName to "c:\library\indo\male\selamat1.wav"
Get tbkMCI("open" && SoundFileName &&"alias waveFile", "")
Get tbkMCI("play waveFile wait", "")
```

These lines of code will play the indicated wave file, and pause the execution of any following code until the sound has finished playing.

As is the case with Visual Basic, the library of `tbkMCI` functions includes one which allows audio recording from within an application:

```
Get tbkMCI("record waveFile", "")
```

In Version 3.0 of Multimedia ToolBook, released in Australia in mid-1994, MCI functions are accessed much as above, except that calls to `tbkMCI()` functions are replaced by using `callMCI()` functions. The new `callMCI()`

functions are native to ToolBook 3.0's version of "OpenScript", Asymetrix's programming language; in Version 1.5, use of `tbkMCI()` functions in OpenScript required an explicit link to Windows MCI support via use of the `TBKMM.SBK` system book.

In Visual Basic the `MCI.VBX` file is required if MCI commands are used; in Multimedia ToolBook 1.5 the corresponding file is `TBKMM.DLL`, while in Multimedia ToolBook 3.0 the file is `MTB30MM.DLL`. Using `TBKMM.DLL` in Version 1.5 of ToolBook also requires linking in the `TBKMM.SBK` file mentioned above.

MMSystem.DLL as an alternative to callMCI()

ToolBook, like Visual Basic, can be made to play audio files through use of the `MMSystem.DLL` file. The advantage of using this DLL is that it obviates the need for the support files required when MCI controls are used; when one is distributing run time modules, fewer files generally equates to less work and worry. The disadvantage is that, as noted above, `MMSystem.DLL` is one-way only; it can play back sound, but it has no in-built routines for recording.

The manner in which ToolBook can be made to work with `MMSystem` varies depending on whether one is using ToolBook 1.5, or ToolBook 3.0. In the earlier version it is necessary to expressly link the DLL via a handler containing a `linkDLL` definition. Once the link has been established, and the `sndPlaySound` function declared in a global module such as `enterBook` or `enterPage`, the function is then used as if it were a natural part of OpenScript's lexicon. (In fact we have to date not done this, but the documentation is readily available in ToolBook's on-line help, where the best citation is found under the `linkDLL` topic. Also see Pierce, 1990).

In ToolBook 3.0 there is no need to specifically link `MMSystem.DLL` in order to gain access to the `sndPlaySound` function. The latest version of OpenScript includes the `playSound()` function which is passed the name of the WAV file to play, plus a control word indicating whether or not execution is to be suspended until the sound finishes. ToolBook 3.0 documentation suggests that this function "works best with files less than 100 kB in size" [7]. We have used the function, but have not rigorously tested the suggested limit as all but a very small minority of our language voice clips are comfortably within 100 kB; this limit, incidentally, corresponds to about four and a half seconds of 8-bit, 22k Hz monaural audio, and to about nine seconds of 8-bit, 11 kHz monaural audio.

The mmCommands in ToolBook 3.0

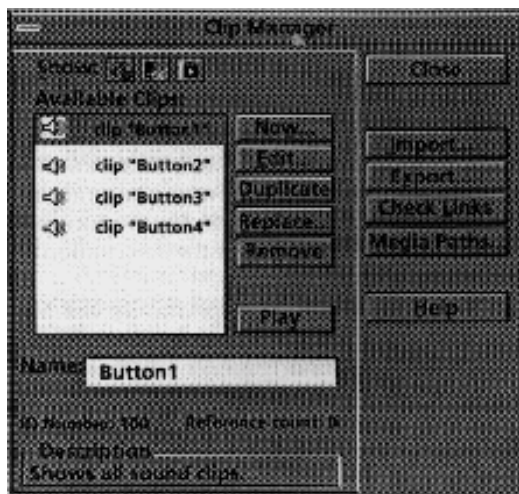
The latest version of ToolBook includes a new set of OpenScript commands designed to make it easier to play media. These are what may be called the 'mmCommands' as they all begin with `mm:` `mmOpen`,

mmPlay, mmPause, etc. Unlike the callMCI() function, the mmCommands have the same format no matter what type of media is used; if the media is visual, the mmCommands may be used in conjunction with a new object called a 'Stage', which is somewhat analogous to the picture box control in Visual Basic.

For developing applications with audio files, the mmCommands are easier to use than corresponding callMCI() functions simply because they have the format of a primitive; however, they cannot be used to record audio, ie., there is no mmRecord command.

A general comment about the multimedia appearance of ToolBook 3.0 may be of interest to some readers: there is enhanced support for using media clips of all sorts. The 'Stage' object is a useful one, considerably facilitating the work involved in rolling visual clips, particularly when compared with the corresponding effort which can be required in Visual Basic and ToolBook 1.5.

The new version of ToolBook also supports clip libraries, and has a clip manager, pictured below.



I have used the 'Sound Finder' utility bundled with the Microsoft Sound System to catalogue and manage hundreds of audio clips; it works well. The new somewhat similar facilities for working with clip libraries from within the ToolBook 3.0 environment itself are likely to appeal to many developers. These facilities are not as comprehensive as they might be - they do not permit extensive documentation to be associated with a clip, nor do they provide technical details about the format of the clip's recording - but they are nonetheless a worthy step forward as they make it easier to pre-load, cue, and manage the variety of media files which an application might use.

Summary of methods for using audio

Visual Basic and ToolBook 1.5 have considerable similarities when it comes to developing applications which make use of audio. Both allow full access to Windows MCI commands; in Visual Basic, MCI access is linked via a special multimedia control object, and the MCI.VBX file must accompany the application. In ToolBook, MCI calls are available in OpenScript, and may be used in any object's code. ToolBook, like Visual Basic, requires additional support files when MCI applications are distributed.

ToolBook 3.0 adds new support for the use of audio clips; these represent an advantage for developers - the new integrated features give the sense of a system especially designed to support the needs of multimedia users, as opposed to one wherein multimedia support is patched in through external functions and utilities.

ToolBook and Visual Basic do not require the use of MCI controls to play back sound. Both systems can link into MMSystem.DLL to access the sndPlaySound function. The advantage of this approach to playing sound is that extra files such as MCI.VBX and MMTBK.DLL need not be distributed with applications.

Visual Basic and ToolBook developers who want their software to permit on-line recording of sound must use MCI calls. There is no recording function in MMSystem.DLL equivalent to sndPlaySound, and the new mmCommands in ToolBook 3.0 do not support recording.

The sound of run time

Developing an application for distribution must include matters related to packaging. From its inception Visual Basic has made the creation of executable (EXE) versions for run time a straightforward process. Visual Basic EXE files, however, cannot stand alone - they must be accompanied by a run-time support library, known as VBRun200.DLL, or VBRun300.DLL, depending on the version of Visual Basic used. These files have respective sizes of 357 kB and 399 kB. The MCI.VBX file, when needed, adds another 30 kB.

ToolBook 1.5 does not have a capability for converting a book to an EXE file; instead one distributes the original book along with 'Runtime ToolBook', TBook.EXE, plus three support library files: TBKComp.DLL, TBKUtil.DLL, and TBKBase.DLL. These four files have respective sizes of 396 kB, 105 kB, 59 kB, and 354 kB, which sum to a collective 900 kB. If MCI commands are used, TBLMM.DLL and TBKMM.SBK must be added in, at 18 kB and 24 kB respectively.

Multimedia ToolBook 3.0 permits a book to be compiled to an EXE form, but this is more properly seen as a bootstrap code module, not a true

executable. The resultant EXE file must be supported by the new Runtime Multimedia ToolBook system, which as a minimum consists of 14 files with a total size of more than 2,600 kbytes (more than 2.6 megabytes).

The run-time differences between Visual Basic and ToolBook 1.5 are pronounced, with ToolBook 1.5's support files taking more than double the space required by Visual Basic. Multimedia ToolBook 3.0 substantially exacerbates the imbalance, extending the disparity to a factor of five.

As attractive as the multimedia support in the latest version of ToolBook is, our own interactive audio work would be quite encumbered if it had to carry along the excess baggage of Runtime Multimedia ToolBook 3.0. We have been able to package interactive "Talk" lessonettes on a single 1.44 MB floppy disk, having as many as 40 simple flip cards, or 18 short phrases or sentences, with both male and female native speakers, and record as well as playback facilities. We have used Visual Basic 2.0 for this, and can fit the compiled EXE file, VBRun200.DLL, MCI.VBX, and all lesson material (including the WAV files) on one floppy diskette, and still have room for a setup program which will install the lesson on the user's hard disk. In fact, our lessons will actually run from floppy disk, users sacrifice some speed when running from a floppy, but not as much as some readers might suspect.

Summary

Visual Basic and ToolBook can both be used to produce interactive audio lessons. The programming capabilities of Visual Basic 2.0, Visual Basic 3.0, and ToolBook 1.5 have considerable similarities, but ToolBook 1.5 run-time files occupy more than 900 kB of space, twice the amount required by Visual Basic.

The latest multimedia version of ToolBook, Multimedia ToolBook 3.0, has substantially superior support for creating audio applications, providing developers with more tools, and a more professional environment, but at cost: the run-time system associated with the new ToolBook consists of more than a dozen files, and occupies more than two and a half megabytes of disk space.

Much of our work has involved developing applications which are used by teachers and students on stand-alone personal systems, often situated on a desk in their homes. Using Visual Basic as our development and delivery system has allowed us to produce software which requires a minimum of support files, easing the work involved when it comes to the distribution and installation of our programs.

The improved authoring tools found in Multimedia ToolBook 3.0 are very attractive; there is no doubt that they will be comfortable to work with. However, most of our "clients" are unlikely to have ToolBook installed on their machines. Were we to switch our development work to ToolBook we

might find it easier to produce systems, but the plethora of files required by Runtime Multimedia ToolBook would greatly increase the complexity of system distribution and installation. I expect most of our Windows-based work will continue to be built on Visual Basic, at least until CD ROM drives are truly pervasive within our target audience.

Notes

1. Australian Commonwealth Department of Employment, Education and Training.
2. Our investigations into the application of Visual Basic and ToolBook have benefited greatly by the work done by some of our postgraduate students, particularly Andre Jamiaay and Ulu Emanuel from the Indonesian Ministry of Education and Culture.
3. Media control interface.
4. Dynamic-link library.
5. Applications programming interface.
6. For example, see article ID Q86281 in the Microsoft Visual Basic Knowledge Base Articles, under the title "Programming Issues". This article is included in the vbknowlg.hlp file distributed with the professional edition of Visual Basic 3.
7. Per p. 2-222 of the *OpenScript Reference Manual* pertaining to version 3.0.

References

- Asymetrix (1994). *ToolBook User Manual*. Bellevue, Washington: Asymetrix Corporation.
- Microsoft (1991). *Microsoft Windows Multimedia Programmer's Reference*. Redmond, Washington: Microsoft Press.
- Microsoft (1993). *Microsoft Visual Basic 3.0 Language Reference*. Redmond, Washington: Microsoft Corporation.
- Nelson, L. (1993). Interactive language learning system for IBM compatible computers. Final report, project 92/9022. Canberra: Department of Employment, Education and Training, Commonwealth of Australia.
- Nelson, L. R. (1994). A working multimedia equivalent to Jurassic language learning flip cards and tapes. In *Proceedings of the 2nd International Interactive Multimedia Symposium*. Perth: Promaco Conventions Pty Ltd.
- Pierce, J. R. (1990). *ToolBook Companion*. Redmond, Washington: Microsoft Press.

Author: Larry Nelson is a senior lecturer at Curtin University's Faculty of Education, where he lectures in research procedures and computer applications in education. His current R&D interests include computer-assisted language learning (CALL), hypermedia authoring systems, and software for use in testing and measurement. Some of the work mentioned here was completed as part of a DEET "ILOTES" (Innovative Languages Other than English) Project. Email address: Nelson_LR@cc.curtin.edu.au

Please cite as: Nelson, L. (1994). Developing and distributing audio applications with Visual Basic and ToolBook. *Australian Journal of Educational Technology*, 10(2), 119-127.
<http://www.ascilite.org.au/ajet/ajet10/nelson.html>