Extending the Musical Capabilities of a Multimedia Authoring Environment

Luciano Vargas Flores
lvf@inf.ufrgs.br

Rosa Maria Viccari
rosa@inf.ufrgs.br

Marcelo Soares Pimenta
mpimenta@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul
Instituto de Informática – Laboratório de Computação & Música
Av. Bento Gonçalves, 9500 Bloco IV - Agronomia - Campus do Vale
Porto Alegre - RS - Brasil CEP 91501-970

Abstract

The ToolBook II™[1] multimedia authoring environment has been successfully used to develop Music educational software under the projects of the Computer Music Lab of the Federal University of Rio Grande do Sul (UFRGS, Brazil). In this paper are reported the extensions made to this environment's musical capabilities during the development of the Rhythmic Training System (or STR, which stands for Sistema de Treinamento Rítmico). These extensions were necessary to allow the random generation of rhythmic sequences, in order to simulate a "rhythmic dictation", an exercise traditionally carried out in Music classes. Such a task demanded the manipulation of MIDI files at the level of musical events, and that is not directly offered by ToolBook's high level of abstraction functions.

The solution proposed by this paper is to employ dynamic-link libraries (DLLs), which can be easily linked to ToolBook applications. In the application example presented here, a sequencing function available in ToolBook plays a temporary MIDI file, created by the function offered in the DLL based in parameters computed by the calling program. The use of temporary files has also proved to be a satisfactory solution to simplify the communication hold via the MIDI protocol.

Although the described solution satisfies STR's design specific requirements, the proposed alternatives for both the extension of ToolBook's musical capabilities and the exchanging of MIDI messages may be applied to other similar projects.

Keywords: Music Education, Music Interface, MIDI, Multimedia, Multimedia Authoring Environment, ToolBook.

[1] ToolBook II is registered trademark of Asymetrix Corporation.

1. Introduction

The research group from the Computer Music Lab of the Federal University of Rio Grande do Sul (UFRGS) is involved in the development of Music educational software. The penultimate of these software to be created is a system that offers support to musical theory classes that involve rhythm. It's called the Rhythmic Training System (or STR, which stands for Sistema de Treinamento Rítmico) and it was presented as a prototype at the Fifth Brazilian Symposium on Computer Music (Fritsch et al., 1998).

STR was implemented using the multimedia authoring environment ToolBook II Instructor version 6.0, from Asymetrix Learning Systems. This was a natural choice, because this environment presents all the expected characteristics of a hypermedia modeling environment (Soares, 1992, p.85) and has been also used as the tool for implementing a former system for the PC platform, called SETMUS (Sistema Especialista para Teoria MUSical, or Musical Theory Expert System – Zucco et al., 1997). Therefore, the research group already had the expertise in successfully using ToolBook to build educational software. In addition, this laboratory plans to integrate all the developed systems in one sole Music education environment, and utilizing the same programming language is going to simplify this task.

However, this choice brought also a problem to be resolved, that was related to the emission of examples and musical dictations requested by the user. These musical passages sometimes had to be generated in real-time by the program.

The prototype upon which the STR was based, the SETMUS for PC, works with musical samples in the Microsoft "Wave" file format (with extension .WAV), that is, digitized waveform uncompressed files. In that system this approach is satisfactory, because what matters in the SETMUS, being a system that recognizes arpeggios and scales, is only the notes' sequence, their order. Neither the notes' durations nor the elapsed time between them, which constitute basically the rhythm, are important in that case. So, in programming the SETMUS, to sound a sequence of notes it was only necessary to play small "Wave" files in the required order, each one of them corresponding to one note of the sequence. Due to the file processing issues involved, the elapsed time between the played notes results to be variable, depending on the availability or not of each note file in the computer's memory. But, as it was said before, in SETMUS this delay is tolerable, because it does not interfere with the musical information being transmitted, since there is no rhythm being dealt with.

The STR, on the other hand, is made to the study of rhythm theory, requiring a computational solution that can handle the musical events' timing, as long as, in this case, the musical notes' durations and the delays between their executions have to be exact. This exigency eliminated, in that project, the possibility of employing the same alternative found in SETMUS, that is the use of digitized waveform files sequences.

Coding in the "Wave" format would be sufficient if the only need was to present audio examples of rhythm to the user. In such a case just one file per example could be used, and these couldn't be modified during execution time, in its timbres or tempos, for instance. Yet one of STR's main modules is a rhythmic dictation, in which the student has to hear rhythmic sequences and then select on the screen the matching rhythmic figures in the correct order. For this drill to offer every time a new dictation, what would challenge the student's reasoning (instead of repeating the same sequence he/she ever heard sometime), it is essential to build a new dictation each time that that is requested. This must be made by the program itself, during execution time and using some random manner of choosing the note sequence. Besides, after this choosing, the sequence can't be executed by just playing wave files in the order calculated by the program, because of the already mentioned file-seeking delays. These delays would alter the desired rhythmic result and confuse the interpretation of the sequence by the student.

The solution was to rely the sound emissions upon a system, the MIDI standard, that would enable exact timing of note sequences and the musical events manipulation in order to build these sequences.

This work’s purpose is to describe the proposed solution to this problem, that is the extension of ToolBook’s musical functionality by means of dynamic-link libraries and the use of temporary MIDI files as an alternative to data exchange. The paper is structured as follows: In section 2 are exposed the reasons that lead to the choice of this particular solution, section 3 describes the application of this solution to STR’s project, in section 4 the results of this usage are presented and, finally, section 5 shows the conclusions these results originated.

2. Extending ToolBook’s musical capabilities

In STR’s design, the decision made was for applying the MIDI (“Musical Instrument Digital Interface”) protocol to all the sound emission processes in the software, instead of using Microsoft’s “Wave”, the digitized wave file format accepted by the ToolBook authoring tool.

Adopting the MIDI protocol brings some advantages to the development of Music educational systems, if compared with the utilization of digitized audio:

· Precision in musical execution;

· Availability of more resources to the user, like the possibility to choose the timbres to be used in the notes execution and the control of the tempo in which are played the dictations and the musical examples (Yavelow, 1992, p.1259); and

· Greater facility to manipulate MIDI files at the level of musical events, with immediate consequences on simplifying the execution time mounting of the rhythmic dictations presented to the student.

The encountered solution included, besides this option for the MIDI, the development of a dynamic-link library, or DLL, of MIDI functions that are by these means available for use in STR’s routines.

The need to extend ToolBook’s capabilities through a DLL is justified by the limitations found in this multimedia developing environment concerning the generation of MIDI music. Basically, ToolBook allows with greater ease the inclusion of MIDI files as “sound clips” of the “book” under development, which wasn’t enough to the purposes of this project.

This environment also offers certain functions to deliver messages directly to the computer’s soundboard, mainly through Windows’ MCI interface (Messick, 1995). However, these functions turned out to be inadequate, as they showed unstable behavior when tested under different hardware platforms during STR’s project. Hence the conclusion was that the development of a DLL would allow the access to functions projected specifically to satisfy the needs of this project in a more effective and robust way.

On defining the DLL’s functions, it was considered unnecessary the addition of some for sequencing purposes, that means, some that could play MIDI files, because the one available in ToolBook is acceptable. So, it was reached an economic, elegant and efficient solution to the DLL’s basic mechanism: it would serve to edit a temporary MIDI file, with content and control parameters passed in the function call. After being created, the temporary file could be played with ToolBook’s own sequencing function. This solution arrived as a simpler alternative than transferring memory pointers between the DLL and the ToolBook program.

3. The “strmidi1” DLL: definition and example of use

The MIDI function FazArqDitado, encapsulated in the “strmidi1.dll” library, has the following calling format:

```
FazArqDitado(inicount, metron, instrum, tempo, sdictat, s2dictat)
```

The six parameters are:

`inicount`	- boolean variable, indicating whether to include or not an initial counting (a sounding reference for the tempo) during one measure before executing the note sequence of the dictation.
`metron`	- boolean variable, indicating if the note sequence of the dictation should be accompained by metronome ticks.
`instrum`	- string containing an integer from 1 to 128 corresponding to a General MIDI standard instrument code, indicating with which timbre should the dictation notes be played.
`tempo`	- string containing an integer from 40 to 208, indicating dictation’s tempo in number of quarter notes per minute.
`sdictat`	- string containing the sequence of duration figures corresponding to the rhythmic dictation, which coding will be explained below.
`s2dictat`	- a parameter not yet implemented (it must be passed an “empty” string), that will allow the definition of a second sequence of duration figures to be executed simultaneously with the first one, constituting a rhythmic dictation in two voices.

The description of the rhythmic dictation to be played, passed as the fifth parameter to the FazArqDitado function, consists of a string of code words separated by spaces. Each one of the code words denotes a rhythmic figure to be played as a musical note or as a rest. The basis to mount the code word is an integer from 1 to 7, which represents one of the seven durations available in STR, according to the correspondence below:

1	- whole note	5	- sixteenth note
2	- half note	6	- 32nd note
3	- quarter note	7	- 64th note
4	- eighth note		

The durations can also be dotted, what is represented simply by including a period next to the number (originating the, in fact, fourteen durations available in the program):

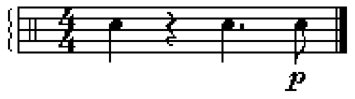
3. - duration with augmentation dot (dotted quarter note)

This part of the code, of course, denotes only the durations of the musical event, note or rest, relative to the employed tempo. To complete the code word it’s necessary to specify the type of musical event. In addition, there was included a syntax to represent whether a note should sound less intense than the others, allowing for the use of the intensity as a rhythmic feature. Both these complements to the code word are indicated by the use of the letter “p”, as in the following example:

p3 - a letter “p” BEFORE a duration symbol indicates a rest of the same duration (in this case, a quarter rest).

3.p - a letter “p” AT THE END of the whole code word indicates that this note should be played with less intensity (the “p” was chosen as it conforms to the dynamic symbol for “piano”). It shouldn’t be used after a rest since this doesn’t make sense in this context (a rest can’t be “played with less intensity”).

That is, therefore, the complete syntax to the coding of the rhythmic dictations. It is not too complex because it contains only the elements needed to accomplish the purposes it was designed for, to be used by the STR software. As an illustration, see the following rhythm line:



To pass this rhythm line as a parameter to the FazArqDitado function it must be coded into the following character string:

“3 p3 3. 4p”

The temporary files generated by this library function, always named after “ditatemp.mid”, record the dictations as rhythm lines to be played only with C4 notes and using the instrument chosen through the third function parameter.

It’s important to point out that, because of design choices made during STR’s project, the dictations will have always the length of just one 4/4 measure. So, the initial counting and the metronome ticks will sound for only one 4/4 measure each. It’s up to the programmer to control the length of the rhythm lines passed in the function call, for the DLL does not handle this internally. The initial counting and the metronome ticks were implemented as beats on one of the percussive instruments available in the General MIDI standard (executed by default in MIDI channel 10).

As it is, the MIDI function is limited, since it’s intended just to satisfy STR’s software requirements. Nevertheless, this example served as a case of study to test if the extension of ToolBook’s musical capabilities via DLL would permit its satisfactory employment in the development of Music educational software.

4. Results

Through the implementation of the MIDI function, the process of recording a temporary file for later execution turned out to be efficient in practice. The elapsed time between the user clicking on an interface button to trigger this process and the music starting to play is imperceptible to humans, seeming even to be instantaneous. Further more technical analyses are being planned, with more exact measurements of this sound response time.

Another satisfactory consequence of this approach is that the resulting software is easy to maintain, since it’s possible to add new functions to the builded DLL, which could then be used in future versions of the STR software, as well as in new Music educational software to be developed under the ToolBook II environment.

5. Conclusions

The achieved results indicate the success of the adopted approach in solving the problem of music generation in STR's project. The extension of ToolBook II's multimedia authoring environment musical functionality by means of a DLL, and also the use of temporary MIDI files as a lesser complex alternative to manipulate MIDI messages for communication purposes, proved to be practicable.

Therefore, this solution may here be proposed to be employed in the development of Music software under the ToolBook II environment, in which shows up the need to exchanging MIDI messages or data between different parts of the system.

6. References

FRITSCH, E. F. et al. (1998). **Desenvolvimento de Software Educacional para a Música: STR - Sistema de Treinamento Rítmico.** In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO E MÚSICA, 5., 1998, Belo Horizonte. **Anais do XVIII Congresso Nacional da Sociedade Brasileira de Computação, v.3.** Belo Horizonte: Escola de Música / UFMG, 1998. p.209-218.

MESSICK, Paul (1995). **Maximum MIDI: Music Applications in C++.** USA, Greenwich: Manning, 1995.

SOARES, Luis Fernando G. et al. (1992). **Fundamentos de Sistemas Multimídia.** Porto Alegre: Instituto de Informática da UFRGS, 1992. (Escola de Computação, 8., 1992, Gramado.)

YAVELOW, Christopher (1992). **Macworld Music & Sound Bible.** San Mateo, California: IDG Books Worldwide, Inc., 1992.

ZUCCO, L. A.; FRITSCH, E. F.; VICCARI, R. M. (1997). **SETMUS: Sistema Especialista para Teoria Musical para Plataforma PC.** In: SALÃO DE INICIAÇÃO CIENTÍFICA, 9., 1997, Porto Alegre. **Livro de Resumos...** Porto Alegre: UFRGS, 1997. p.23-24.