Dynamic MIDI Editor: A Windows Application for Computer Music

LUIZ ANTONIO PINHEIRO MARTINS
Oficina de Análise e Síntese da Imagem e do Som
Departamento de Ciência da Computação - UFMG
Caixa Postal 702 - 30161.970 - Belo Horizonte - MG - Brazil
email: pinheiro@dcc.ufmg.br

Abstract

The Dynamic MIDI Editor (DyME) is a Windows application for computer music that allows the user to apply the editing process of MIDI instruments over a musical sequence. This paper introduces DyME and tries to show that it extends the capabilities of MIDI protocol, allowing the definition of dynamic instruments. DyME offers the real time capabilities of MIDI and dynamic control of timbre, which is actually possible only in SWSS systems. DyME writes "parameter change" commands in MIDI files, integrating editing and sequencing processes.

Introduction

Systems for computer music has been developed along two main lines, in spite of the possibilities of interaction between them (Ballista, Casali, Chareyron, & Haus, 1992; Jaffe, Smith, & Porcaro, 1994) and the different synthesis methods developed: Software Sound Synthesis (SWSS) and Musical Instrument Digital Interface (MIDI). In the former, musical material is completely generated by software, which defines notes and instruments. In the latter, computer stores only notes and other control information, with the timbre defined by the instrument selected in a synthesizer.

Synthesis power of SWSS systems is greater than that of any MIDI synthesizer, because it is not limited by hardware and synthesis method of a specific synthesizer model. The various types of modules found in a synthesizer can be arbitrarily defined by software. Thus, any commercial synthesizer can be rebuilt, using acoustical compilers such as cmusic, cmix, or csound (Pope, 1993). Besides, each instrument defined in an acoustical compiler can use a different synthesis method, simulating different synthesizers.

On the other hand, the editing process of MIDI instruments is interactive. While in SWSS a relatively long processing time is needed before the sound file can be played, in MIDI the synthesis process is controled in real time, while the parameters that define an instrument are modified.

However, the instrument editing usually is used to define in advance an instrument that, further, can be selected to play a musical passage. In MIDI, the possibilities of instrument transformation during play are very limited, because editing and sequencing are normally separate processes.

DyME is an environment for the definition of dynamic MIDI instruments. These instruments are conducted along the MIDI sequence by sending "system exclusive - parameter change" messages. That way, the editing process is implemented dynamically, in agreement with the expressive needs of the musical flow.

Static Editing

After the introduction of the MIDI protocol, the music industry experienced an enormous development and from day to day more musicians have access to MIDI instruments. The definition of instruments by editing process became part of the composer's work, besides his or her other musical tasks. MIDI devices are very popular thanks to features such as high reliability, ease of use, low cost and related software availability.

However, it is clear that MIDI instruments do not present the same flexibility as a conventional instrument with relation to parameters such as dynamics, articulation, vibrato, attack and tune, for example. Besides the inherent problems with the MIDI protocol (Moore, 1988), the behavior of MIDI instruments is musically poor because it is maintained almost constant along the musical flow, except by alterations defined by touch and periodic oscillations. Editing is usually a static process, and the behavior of the instrument is bounded in advance by that process.

Dynamic Editing

The possibilities of dynamic control over the editing process of MIDI instruments are very limited, unless a special tool is used. Similar instruments can be selected sequentially by the "program change" command to simulate the desired effect, but in this case the number of changes would be limited by the number of synthesizer programs, which is not enough for smoothly controlling that process. Besides, the "program change" command can be used only over pauses, because it introduces sound discontinuities (clicks) when it is applied over notes in execution.

"System exclusive is the great escape hatch of MIDI" (Loy, 1985). In some commercial software sequencers, it is possible to send "system exclusive" strings to the synthesizer, including the "parameter change" messages used by DyME. In this case, for each parameter change it is necessary to consult the address and range of the parameter that will be modified, to translate it to hexadecimal form and to write, one by one, those strings. These programs are not efficient to manage hundreds of "parameter change" strings, as can be done with DyME, because they are not a special environment for this task.

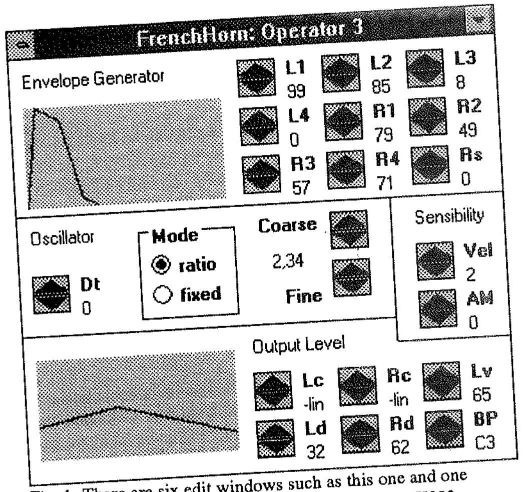
DyME tries to fill the lack of resources for dynamic control of timbre observed in MIDI, by increasing the flexibility of MIDI instruments. In this manner, various kinds of timbre transformations can be implemented: Sometimes it is desirable to implement many changes on weak parameters to produce a smooth transformation in timbre. Otherwise, it is convenient to change a strong parameter that has immediate effect on the sound characteristics (Jaffe, 1994). In any case, the dynamic editing process is associated to the idea of dynamic instruments which adapt to expressive needs of the musical flow.

The dynamic control of timbre can be implemented on SWSS systems, where the definition of instruments is almost unlimited: related instruments can be called sequentially along the score to reach that objective, or a control stream can be used on the instrument. DyME does not propose a very original process in computer music, but claims to extend the capabilities of the MIDI instruments in this field. It takes advantage of the fact that the MIDI synthesis process is more interactive than the same process using SWSS systems. Besides, its high level interface controls the dynamic editing process in an easy and efficient way.

Naturally, the dynamic editing process implemented by DyME is limited by the hardware of the synthesizer used. However, the musical result of this process is very attractive, greatly overcoming the possibilities normally presented by that synthesizer. DyME shows clearly that the musical limitation of MIDI instruments is caused by the absence of resources for dynamic control of timbre in the MIDI protocol. Unfortunately, the only way to solve this problem is using system exclusive commands, that are specific to a particular synthesizer model.

A Dynamic Editing Session

The main characteristic observed in DyME is its operative simplicity. Anybody who has already performed a conventional editing of a MIDI instrument can implement dynamic editing by just adding one step to that process: applying the editing process previously defined on the selected part of a sequence. DyME compares the configuration preceding the instrument editing with the subsequent one and approaches them, through "parameter change" messages, which are written on a midifile. In this process, it interpolates the parameter values depending on which option has been selected.


FrenchHorn: Operator 3

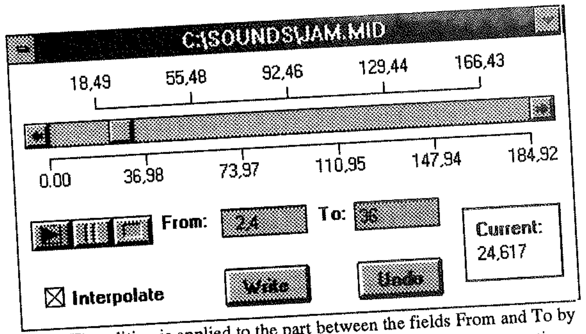
Envelope Generator: L1 99, L2 85, L3 8, L4 0, R1 79, R2 49, R3 57, R4 71, Rs 0

Oscillator: Dt 0; Mode: [✓ ratio] fixed; Coarse 2,34; Fine

Sensibility: Vel 2, AM 0

Output Level: Lc -lin, Rc -lin, Lv 65, Ld 32, Rd 62, BP C3

Fig. 1: There are six edit windows such as this one and one edit window for general parameters of the TX802.


C:\SOUNDS\JAM.MID

18,49 55,48 92,46 129,44 166,43

0,00 36,98 73,97 110,95 147,94 184,92

From: 24 To: 36 Current: 24,617

[✓ Interpolate]

Fig.2: The editing is applied to the part between the fields From and To by the button Write. The Undo button undoes the last write operation.

The dynamic editing process is as follows. The user opens a midifile recorded in advance and selects the option Receive which requests the parameters from the current instrument in the synthesizer. The next part resembles a static editing process: in the edit windows (there are seven edit windows in DyME) the parameters are modified. Thus, another parameter configuration and a new timbre are defined.

However, there is a difference between this process and the static one: the user has in mind the musical passage which will receive that dynamic editing. Rather than defining a closed instrument to be further selected, the user is controling a dynamic transformation from the previous timbre to that defined by him or her. In the last part of the process, the alterations implemented on the parameters are applied to the musical sequence by the Write button, that is present in the dynamic edit window. In this window the sequence can be played, and the editing process previously defined is reproduced during play. If the result of the last operation has not been interesting, it can be undone by the Undo button. When the user returns to the edit windows to restart the process, they are updated to the point where the execution of the midifile has stopped.

DyME supports midi files format 1.0. Each write operation creates a new track of "parameter change" messages in the midifile. It is possible to record up to 32767 "parameter change" tracks minus the number of tracks recorded previously by the sequencer. A unique write operation can record up to hundreds of "parameter change" messages on the midifile.

Implementation

DyME was implemented in the Microsoft Visual Basic programming system, with a DLL (dynamic linking library) defined in C++, using the compiler Borland C++ 3.1 (Norton, & Yao, 1992). The ease for interface development observed in Visual Basic and the support offered by this platform to API windows (including multimedia) justify its use. The DLL are used to tasks such as communication with the synthesizer, and reading and writing of midi files, where interpreted Basic code would threaten the processing efficiency.

The synthesizer used is the Yamaha TX802, that uses FM synthesis. The use of this synthesis method (Chowning, 1973) is coherent with the objectives of the work. On this module, it is possible to solve the MIDI deficiency on the microtonal field and to control the synthesis process since its source, which is not true for synthesizers that use sampled sounds. Further versions of DyME will be determined by the system exclusive of the synthesizer used. All the parameters that can be changed by that system exclusive will be able to be dynamically controled by DyME.

OOP Concepts Used by DyME

Visual Basic is an object-based (not object-oriented) programming system (Booch, 1994). In VB, the user never explicitly use concepts like inheritance, class, and polymorphism. Besides its object-based characteristics, Visual Basic is event-oriented also. The program is structured in a bottom-up way, and the user is responsible for the execution order of the program.

The manner the program was built reflects itself in its operation. For example, when the user creates an edit window called Operator 1, it is an object of the Operator class (in Visual Basic classes are represented by foms and controls). This object shares the functionality of that class, being updated by the musical play. A great number of buttons are used in the edit windows. The Spin button class follows generic methods, which are implemented in a different manner in each instance of a button object.

Limitations

Like any system that controls "system exclusive" streams, DyME is dedicated to a synthesizer model, and its interface defined especially for that synthesizer model. Naturally, it is possible and desirable to implement different versions of DyME for other commercial synthesizers.

It was observed in the TX802 (version 07/25/87) module that it is not possible to select the program that will receive the "parameter change" message by the channel information present in that string. The synthesizer received "parameter change" strings only in the program indicated by its cursor. So, using that module it was not possible to send "parameter change" messages to various channels simultaneously and to take advantage of its multitimbral resources.

Conclusion

The day is very near in which cheap SWSS systems will be able to work in real time. Maybe then it will not be necessary to use DyME, which is limited by the specific architecture of a synthesizer model and inherent problems of MIDI protocol. For the present, DyME proposes to extend the possibilities of MIDI synthesizers to timbral control, gaining the advantages of their great diffusion and real time capabilities.

References

Ballista, A., Casali, E., Chareyron, J., & Haus, G. (1992). A MIDI/DSP Sound Processing Enviroment for Computer Music Worksation. *Computer Music Journal* 16(3): 57-72.
Booch, G. (1994). *Object-Oriented Analysis Design with Applications*. Benjamin/Cummings.
Chowning, J. (1973). The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. *Journal of Audio Engineering Society* 21(7): 526-534
Jaffe, D., Smith, J., & Porcaro, P. (1994). The Music Kit on a PC. *Proceedings of the I Brazilian Symposium of Computer Music*: 63-69.
Jaffe, D. (1994). An Overview of Criteria for Evaluating Synthesis and Processing Techniques. *Proceedings of the I Brazilian Symposium of Computer Music*: 53-60.
Loy, G. (1985). Musicians Make a Standard: The MIDI Phenomenon.. *Computer Music Journal* 9(4).
Microsoft Windows: Multimedia Programmers Reference. Microsoft Press, 1990
Microsoft Visual Basic Programmer's Guide. Microsoft Press, 1993
Moore, R. (1988). The Dysfunctions of MIDI. *Computer Music Journal* 12(1):19-28
Norton, P., & Yao, P. (1992). *Programando em Borland C++ para Windows*. Berkeley Brasil Editora.
Oppenheim, D. (1989). *MIDI File Especification*. International MIDI Association.
Petzold, C. (1992). *Programando em Windows 3.1*. Makron Books.
Pope, S., T. (1993). Machine Tougues XV: The Three Packages for Software Sound Synthesis. *Computer Music Journal* 17(2): 23-53.
TX802 Owner's Manual - Yamaha Music Foundation, 1988

Acknowledgements

This work is part of my Master degree dissertation, defended in May 23, 1995 in the Department of Computer Science of Federal University of Minas Gerais (UFMG). It was supervised by Wilson de Pádua Paula Filho, and supported by National Council of Scientific and Technological Development (CNPq).

Software Auto-Instrucional em Teoria Musical

MÁRCIO DA COSTA PEREIRA BRANDÃO
MÍRIAM RICA SAMBUICHI
Laboratório de Processamento Espectral
Departamento de Ciência da Computação
Universidade de Brasília - Brasília, DF
CEP 70910-900 BRASIL
e-mail: brandao@cic.unb.br

RESUMO

Um courseware multimídia introdutório em teoria musical, desenvolvido para o ambiente Windows, é aqui descrito. Os tópicos básicos abordados são agrupados nas seguintes lições: definição de música e seus componentes; claves e localização das notas no teclado; divisão proporcional de valores; compassos; tom, semitom e alterações; classificação dos intervalos naturais; escalas maiores e menores. A apresentação é feita utilizando a notação musical convencional, sendo ilustrada com exemplos sonoros pertinentes. A navegação se dá através de botões que permitem ao aluno seguir em frente, voltar a telas anteriores ou mesmo mudar de lição. O aprendizado pode ser avaliado através de exercícios de fixação apresentados ao longo das lições ou de testes de avaliação no final de cada lição.

1. Introdução

Um software auto-instrucional - ou courseware - possibilita que se aprenda sozinho, de uma forma interativa, num ritmo individualizado e utilizando o próprio tempo disponível. Eles ainda não substituem por completo o processo de ensino/aprendizado tradicional, mas mostram-se eficazes na execução de tarefas tais como repetir, ordenar, calcular, relacionar, ler e escrever, por exemplo. Atualmente, observa-se que um número cada vez maior de pessoas tem se interessado por esta forma de aprendizado graças à integração de recursos multimídia (texto, sons e imagens) na apresentação de informações de forma agradável e interativa.

O objetivo deste trabalho é descrever as etapas de desenvolvimento (Franco & Rêgo, 1991) de um courseware multimídia introdutório em teoria musical (Sambuichi, 1994). O conteúdo deste courseware limita-se a alguns pontos da teoria musical, sendo a linguagem voltada para um público que não necessita possuir um conhecimento prévio do assunto.

2. Etapas do Desenvolvimento

A metodologia adotada para a construção deste courseware seguiu as seguintes etapas:

- Análise
- Definição dos Objetivos.
- Planejamento.
- Desenvolvimento
- Programação
- Validação
- Revisão