Using Physical Modelling Synthesis For Implementing PC Based Softsynth

Jagruti Sanghavi†, Dr.N.G.Bawane†† and Akansha Pilley†††

†Student of G.H.Raisoni college of Engineering, Nagpur University, Nagpur, India
††Faculty of G.H.Raisoni college of Engineering, Nagpur University, Nagpur, India
††† Student of G.H.Raisoni college of Engineering, Nagpur University, Nagpur, India

Summary
A software synthesizer, also known as a softsynth or virtual instrument is a computer program for digital audio generation. Softsynths can be cheaper and more portable than dedicated hardware. In this paper we proposed a scheme in which a dedicated hardware musical keyboard is developed which sends only MIDI notes. A keyboard is interfaced to PC through USB.A synthesizing of musical notes is done by PC to produced sound. Synthesis is done by comparing different physical modeling technique. This is the low cost solution towards dedicated hardware synthesizers. The paper aims towards using existing use of processing power of PC with existing memory system to process MIDI notes to play different instruments through general purpose PC. Also it makes easy for keyboard player to play music with dedicated hardware keyboard. Associated with the electronic music movement, a synthesizer is an electronic instrument, sometimes accessed through a keyboard, that creates and combines waveforms used stored acoustic instrumental samples, called wavetable synthesis, or electronically, using FM synthesis.Synthesis is done by comparing different physical modeling technique. Proposed system aimed at developing a synthesizer system using computer devices where all the processing done by computer system. Main idea behind project is to avoid using readymade synthesizer IC and utilization of computer processing to get maximum computer power. The physical model is usually formulated as a partial differential equation resulting from a mechanical analysis. The resulting synthesis algorithms consist of a parallel arrangement of second order digital filters [2]. Their coefficients are obtained by analytic expressions directly from the parameters of the physical model. More elaborate computational models include nonlinearities and excitation mechanisms. The resulting algorithmically models are suitable for real-time implementation on modern desktop or laptop computers and mobile devices. Low-delay algorithms permit control from sequencer programs or haptic devices. A VST-plug-in demonstrates the capabilities for real-time synthesis and parametric control.

Key words: *MIDI, Softsynth ,physical modeling.*

1. Introduction

A music synthesizer makes sounds by using an electrical circuit as an oscillator to create and vary the frequency of sounds in order to produce different pitches. As long as the pitch is within the range of frequency that can be heard by a human ear, it's known as a "musical pitch". You can use a keyboard to vary these pitches at discrete intervals that correspond to the notes on the musical scale. If you put several oscillators together, you can combine several pitches to create a "chord". How do you vary the tone of a particular pitch? That is done by playing a given pitch with waveforms of different shapes (common waveforms include sine, square, saw tooth, and triangle waveforms). Since the harmonic structure of these waveforms differs, our ears interpret them as different tones. The sound you will hear can also be modified by voltage-controlled amplifiers (VCA) and voltage-controlled filters (VCF).

Synthesizers are able to only mimic the sounds of non-synthetic instruments, but also to create sounds that absolutely cannot be played by anything but a music synthesizer. That is because a music synthesizer is well-suited to delicate manipulations of its oscillators. Nevertheless, it's a lot easier for a synthesizer to create entirely new sounds than to mimic the sounds of acoustic instruments because the waveforms of acoustic instruments are so complex. Interestingly, once complex sound that synthesizers so far have been very bad at reproducing is the human voice (although improvements are being made in this technology). The entire electronic music scene would be virtually impossible without the use of synthesizers (no doubt some wish it were). Nevertheless, the number of sounds that a musician has to work with has been exponentially increasing in recent decades, and we have only scratched the surface of the creative possibilities.

Synthesizing using physical modeling in software. Many sound synthesis methods like sampling, frequency modulation (FM) synthesis, additive and subtractive synthesis model sound. This is good for creating new sounds, but has several disadvantages in reproducing sounds of real acoustic instruments. The most important disadvantage is that the musician does not have the physical based variability he has with real musical instruments. Therefore it is difficult to phrase a melody with these methods. Because of these disadvantages there are various methods for sound synthesis based on physical

Manuscript received March 5, 2010
Manuscript revised March 20, 2010

models that do not model the sound but the sound production mechanism. They all start from physical models in form of partial differential equations (PDEs). They can be obtained by applying the first principles of physics. But due to the differential operators the resulting PDEs cannot be solved analytically.

2. Theoritical Foundation

Below are various physical-model representations

- Ordinary Differential Equations (ODE)
- Partial Differential Equations (PDE)
- Difference Equations (DE)
- Finite Difference Schemes (FDS)
- Transfer Functions (between physical signals)

ODEs and PDEs are purely mathematical descriptions (being differential equations), but they can be readily ``digitized'' to obtain computational physical models. Difference equations are simply digitized differential equations. That is, digitizing ODEs and PDEs produces DEs. A DE may also be called a finite difference scheme. A discrete-time state-space model is a special formulation of a DE in which a vector of state variables is defined and propagated in a systematic way (as a vector first-order finite-difference scheme). A linear difference equation with constant coefficients--the Linear, Time-Invariant (LTI) case--can be reduced to a collection of transfer functions, one for each pairing of input and output signals (or a single transfer function matrix can relate a vector of input signal z transforms to a vector of output signal z transforms). An LTI state-space model can be diagonalized to produce a so-called modal representation, yielding a computational model consisting of a parallel bank of second-order digital filters. Digital waveguide networks can be viewed as highly efficient computational forms for propagating solutions to PDEs allowing wave propagation. They can also be used to compress the computation associated with a sum of quasi harmonically tuned second-order resonators.

A .ODEs (Ordinary Differential Equations)

Ordinary Differential Equations (ODEs) typically result directly from Newton's laws of motion, restated here as follows:

$$f(t) = m\ddot{x}(t) \qquad (1)$$

The initial position *x(0)* and velocity *v(0)* of the mass comprise the *initial state* of mass, and serve as the *boundary conditions* for the ODE. The boundary conditions must be known in order to determine the two constants of integration needed when computing *x(t)* for *t>0.*

If the applied force *f(t)* is due to a spring with spring-constant k, then we may write the ODE as

$$x(t) + m\ddot{x}(t) = 0 \qquad (2)$$

(Spring Force + Mass Inertial Force = 0)
This case is diagrammed in Fig.1.2.

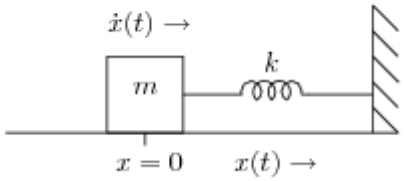

Fig 1 Mass –Spring Diagram

B. PDEs (A partial differential equation)

A *partial* differential equation (PDE) extends ODEs by adding one or more independent variables (usually spatial variables). For example, the wave equation for the ideal vibrating string adds one spatial dimension x (along the axis of the string) and may be written as follows:

$$Ky''(x,t) = \epsilon\ddot{y}(t) \qquad (3)$$

(Restoring Force = Inertial Force)

Where *y(x, t)*denotes the *transverse* displacement of the string at position x along the string and time t, and

$$y'(x,t) \triangleq \frac{\partial y(x,t)}{\partial x} \qquad (4)$$

denotes the *partial derivative* of *y* with respect to x.

The physical parameters in this case are string tension *k* and string mass-density ϵ.

C. Difference Equations (Finite Difference Schemes)

There are many methods for converting ODEs and PDEs to difference equations. One method is to replace each derivative with a *finite difference*:

$$\dot{x}(t) \triangleq \frac{d}{dt} \quad (x,t) \triangleq \lim_{\delta \to 0} x(t) - \frac{x(t-\delta)}{\delta} \approx x(nt) - x(n-1)T/T \tag{5}$$

Consider a mass m driven along a frictionless surface by a driving force *f (t)*, as in Fig.1, and suppose we wish to know the resulting velocity of the mass *v (t)*, assuming it starts out with position and velocity 0 at time 0. Then, from Newton's

$$f = ma \tag{6}$$

relation, the ODE is

$$f(t) = m\,\dot{v}(t), \tag{7}$$

and the difference equation resulting from the backward-difference substitution is

$$f(nt) = m\frac{v(nt) - v[(n-1)]T]}{T}, n = 0,1,2 \tag{8}$$

Solving for v *(nT)* yields the following *finite difference scheme:*

$$v(nt) = v[(n-1)T] + \frac{T}{mf(nt)}, n = 0,1,2 \tag{9}$$

Finite difference scheme in explicit *form can* be implemented in real time as a *causal digital filter*. There are also *implicit* finite-difference schemes which may correspond to *non-causal* digital filters .

D. Transfer Functions

A discrete-time *transfer function* is the *z* transform of the *impulse response* of a linear, time-invariant (LTI) system. In a physical modeling context, we must specify the input and output signals we mean for each transfer function to be associated with the LTI model. For example, if the system is a simple mass sliding on a surface, the input signal could be an external applied force, and the output could be the velocity of the mass in the direction of the applied force. In systems containing many masses and other elements, there are many possible different input and output signals. It is worth emphasizing that a system can be reduced to a set of transfer functions only in the LTI case, or when the physical system is at least *nearly linear* and only *slowly* time-varying (compared with its impulse-response duration).

II. DIGITAL WAVEGUIDE MODELLING ELEMENTS

The ideal wave equation comes directly from Newton's laws of motion *f=ma* . For example, in the case of vibrating strings, the wave equation is derived as

$$Ky'' = \epsilon\ddot{y}$$

(Restoring Force Density = Mass Density times Acceleration), Where

$$K \triangleq \text{string tension}$$
$$\epsilon \triangleq \text{linear mass density.}$$

(1)

Defining c= sqrt (k/ ϵ)

We obtain the usual form of the PDE known as the *ideal 1D wave equation*.

$$y'' = \frac{1}{c^2}\ddot{y} \tag{2}$$

where *y (t ,x)* is the string displacement at time t and position x . For example, y can be the transverse displacement of an ideal stretched string or the longitudinal displacement (or pressure, velocity, etc.) in an air column. The independent variables are time t and the distance x along the string or air-column axis. The partial-derivative notation is more completely written out as

$$\ddot{y} \triangleq \frac{\partial^2}{\partial t^2} y(t,x)$$
$$y'' \triangleq \frac{\partial^2}{\partial x^2} y(t,x).$$

(3)

As has been known since d'Alembert , the 1D wave equation is obeyed by arbitrary *traveling waves* at speed c

$$y(t,x) = y_r(t - x/c) + y_l(t + x/c) \tag{4}$$

In digital waveguide modeling, the traveling-waves are *sampled:*

$$\begin{aligned} y(nT, mX) &= y_r(nT - mX/c) + y_l(nT + mX/c) \qquad (\text{set } X = cT) \\ &= y_r(nT - mT) + y_l(nT + mT) \\ &\triangleq y^+(n-m) + y^-(n+m) \end{aligned} \tag{5}$$

where T denotes the time sampling interval in seconds, $X = cT$ denotes the spatial sampling interval in meters, and y+ and y - are defined for notational convenience. This vibration of a string can be modeled simply using a digital waveguide. This consists of two delay lines representing two traveling waves moving in opposite directions. By summing the values at a certain location along the delay lines at every time step, we obtain a waveform. This waveform is the sound heard with the pickup point placed at that relative location. The delay elements are initialized with a shape corresponding to the initial displacement of the string. For simplicity a triangular wave is used even though in reality the initial displacement of a plucked string will not be shaped exactly like a triangle. Simply using two delay lines in this fashion would require arbitrarily long delay lines depending on the length of the desired output. By feeding the delay lines into each other a system can be created that can run for an arbitrary amount of time using fixed size delay elements.

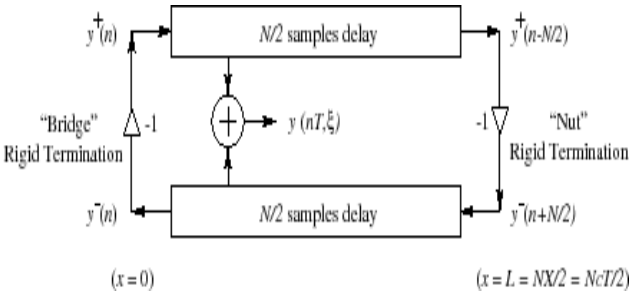

Figure 2 : Digital waveguide of rigidly terminated string.

2. Reason for selection of project

Proposed system is a combination of computer system and MIDI keyboard to generate synthesizer instrumental frequency by synthesizing sound using physical modeling using digital loop filter method. Whenever we install device driver for sound card in computer system it will install logical driver input device, output device and synthesizer device that is MIDI device. Computer sound card is capable of playing different instrumental frequency using synthesizer device. By developing software application we can utilize these synthesizer capabilities of sound card. As we are developing software application we can make it dynamic at any extend by proving power of input and output device available in the system. Only problem with such a software application if lack of flexibility in playing piano like keys. It is not possible to play music using computer keyboard and hence many artist avoid such a software system. So to overcome this problem proposed system is developed in such a manner that it will provide power of computer processing and the flexibility of piano keyboard. The project is further divided in different module like processing unit, piano keyboard interfacing with computer system and output devices. So the main reason for developing this project is to provide maximum PC power to synthesizer. Many synthesizer ICs available in the market and will these are build for specific task to generate different instrumental frequency on speaker system. In many recording studio artist use PC system for further processing.

3. Research methodology to be employed

3.1 API Technology

Windows APIs are dynamic link libraries (DLLs) that are part of the Windows operating system. You use them to perform tasks when it is difficult to write equivalent procedures of your own. For example, Windows provides a function named FlashWindowEx that lets you make the title bar for an application alternate between light and dark shades. The advantage of using Windows APIs in your code is that they can save development time because they contain dozens of useful functions that are already written and waiting to be used. The disadvantage is that Windows APIs can be difficult to work with and unforgiving when things go wrong. Windows dynamic-link libraries (DLLs) represent a special category of interoperability. Windows APIs do not use managed code, do not have built-in type libraries, and use data types that are different than those used with Visual Studio .NET. Because of these differences, and because Windows APIs are not COM objects, interoperability with Windows APIs and the .NET Platform is performed using platform invoke, or PInvoke. Platform invoke is a service that enables managed code to call unmanaged functions implemented in DLLs. For more information, see Consuming Unmanaged DLL Functions. You can use PInvoke in Visual Basic .NET by using either the **Declare** statement or applying the **DllImport** attribute to an empty procedure. Windows API calls were an important part of Visual Basic programming in the past, but are seldom necessary with Visual Basic .NET. Whenever possible, you should use managed code from the .NET Framework to perform tasks instead of Windows API calls. This walkthrough provides information for

those situations in which using Windows APIs is unavoidable.

3.2 MIDI technology

The Musical Instrument Digital Interface (MIDI) protocol has been widely accepted and utilized by musicians and composers since its conception in 1983. MIDI data is a very efficient method of representing musical performance information, and this makes MIDI an attractive protocol not only for composers or performers, but also for computer applications which produce sound, such as multimedia presentations or computer games. However, the lack of standardization of synthesizer capabilities hindered applications developers and presented new MIDI users with a rather steep learning curve to overcome. Fortunately, thanks to the publication of the General MIDI System specification, wide acceptance of the most common PC/MIDI interfaces, support for MIDI in Microsoft WINDOWS and other operating systems, and the evolution of low-cost music synthesizers, the MIDI protocol is now seeing widespread use in a growing number of applications. This document is an overview of the standards, practices and terminology associated with the generation of sound using the MIDI protocol.

3.3 MIDI vs. Digitized Audio

Originally developed to allow musicians to connect synthesizers together, the MIDI protocol is now finding widespread use as a delivery medium to replace or supplement digitized audio in games and multimedia applications. There are several advantages to generating sound with a MIDI synthesizer rather than using sampled audio from disk or CD-ROM. The first advantage is storage space. Data files used to store digitally sampled audio in PCM format (such as .WAV files) tend to be quite large. This is especially true for lengthy musical pieces captured in stereo using high sampling rates. MIDI data files, on the other hand, are extremely small when compared with sampled audio files. For instance, files containing high quality stereo sampled audio require about 10 Mbytes of data per minute of sound, while a typical MIDI sequence might consume less than 10 Kbytes of data per minute of sound. This is because the MIDI file does not contain the sampled audio data, it contains only the instructions needed by a synthesizer to play the sounds. These instructions are in the form of MIDI messages, which instruct the synthesizer which sounds to use, which notes to play, and how loud to play each note. The actual sounds are then generated by the synthesizer. For computers, the smaller file size also means that less of the PCs bandwidth is utilized in spooling this data out to the peripheral which is generating sound. Other advantages of utilizing MIDI to generate sounds include the ability to easily edit the music, and the ability to change the playback speed and the pitch or key of the sounds independently. This last point is particularly important in synthesis applications such as karaoke equipment, where the musical key and tempo of a song may be selected by the user.

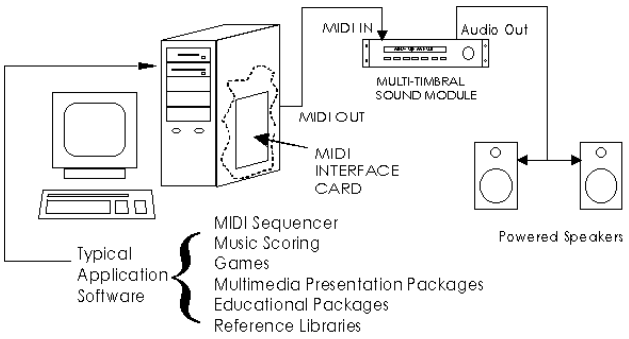

Figure 3. A PC-Based MIDI System

4. Conclusion

Proposed systems aims towards implementing low cost synthesizers using existing PC power .The inputs are provided by only buttons and processing is done at PC. Many sound synthesis methods like sampling, frequency modulation (FM) synthesis, additive and subtractive synthesis model sound. This is good for creating new sounds, but has several disadvantages in reproducing sounds of real acoustic instruments. The most important disadvantage is that the musician does not have the physical based variability he has with real musical instruments. Therefore it is difficult to phrase a melody with these methods.

Because of these disadvantages there are various methods for sound synthesis based on physical models that do not model the sound but the sound production mechanism. They all start from physical models in form mathematical equations such as of partial differential equations (PDEs). They can be obtained by applying the first principles of physics. These mathematical equations are then realised into equivalent circuits by digital wave guide method to get real melody of instrument.

References

[1] C DVDs , March-April 2004 , Software - Powerful dynamics and effects ,Computer Graphics Digital Library ,Volume:24, Issue:2, pp111-111.

[2] Farshad Arvin, and Shyamala Doraisamy, (2009),"A Real-Time Signal Processing Technique for MIDI Generation", WASET Journal, Volume 50, pp:593-597.

[3] Carla Scaletti (Winter 2002) "Computer Music Languages, Kyma, and the Future" ACM-Portal Computer Music Journal Volume 26 , Issue 4 , pp 69-82 .

[4] Braun, J.P.; Gosbell, V.J.; Perera, S (2004) "Power quality waveform generator based on the CSound software sound synthesizer", 11th IEEE International Conference on Digital Object Identifier Volume 12,Issue 15 Sept , pp.391 – 396

[5] Victor E. P. Lazzarini, (April 2000) "The SndObj Sound

Object Library"Volume 5 , Issue 1 Pages: 35 – 49.
[6] Michael Droettboom,April, 2002 "Selected Research in Computer Music " Submitted in partial fulfillment of the requirements for the degree of Master of Music in Computer Music Research at The Peabody Conservatory of Music, The Peabody Institute of the Johns Hopkins University.
[7] Victor Lazzarini (summer 2009) "Distortion Synthesis" A tutorial with Csound examples Issue 11,July 15.
[8] Kapil Krishnamurthy, "GENERATION OF CONTROL SIGNALS USING PITCH AND ONSET DETECTION FOR AN UNPROCESSED GUITAR SIGNAL" Final paper at Center for Computer Research in Music and Acoustics Stanford University.
[9] Aaron Hechmer, Adam Tindale , George Tzanetakis (October 28 – 31, 2006), "LogoRhythms: Introductory Audio Programming for Computer Musicians in a Functional Language Paradigm", 36th ASEE/IEEE Frontiers in Education Conference,pp-1-6.
[10] Ross Bencina (31st August 2001) "Implementing Real-Time Granular Synthesis"
[11] Thomas Ciufo,"Design Concepts and Control Strategies for Interactive Improvisational Music Systems" ,Special Studies/Music Brown University,
[12] M. Helmuth July 1996 "Granular synthesis composition with StochGran and Max " A tutorial on science Direct –Computers and Mathematics with applications ,volume 32,issue1,pp 57-74.
[13] Phil Burk, (1998)JSyn – "A Real-time Synthesis API for Java"International computer music conference pp.1-4.
[14] Lazzarini, V., J. Timoney and T. Lysaght 2008,"Split-Sideband Synthesis". *Proceedings of the ICMC 2008*, Belfast, UK.
[15] Lazzarini, V., J. Timoney and T. Lysaght 2007,"Adaptive FM synthesis". Proceedings of the 10th Intl. Conference on Digital Audio Effects (DAFx07). Bordeaux: University of Bordeaux: 21-26.
[16] Lazzarini, V., J. Timoney and T. Lysaght 2008,"The Generation of Natural-Synthetic Spectra by Means of Adaptive Frequency Modulation". *Computer Music Journal*, 32 (2): 12-22.
[17] Lazzarini, V., J. Timoney and T. Lysaght 2008,"Asymmetric Methods for Adaptive FM Synthesis". *Proceedings of the International Conference on Digital Audio Effects*, Helsinki, Finland.
[18] Dave Phillips (Winter 2003) " Computer Music and the Linux Operating System A Report from the Front " Computer Music Journal, 27:4, pp. 27–42.
[19] Stephen Travis Pope, (1995)"Computer Music Workstations I Have Known and Loved"Computer music journal, pp.1-7.
[20] Gareth loy,,Curtis Abott (June 1985)"Programming languages for computer music synthesis, performance, and composition " ACM Computing Surveys (CSUR) Volume 17 , Issue 2 ,Pp: 235 - 265 .
[21] James McCartney Winter 2002 Rethinking the Computer Music Language: SuperCollider, Computer Music Journal , MIT Press, Vol. 26, No. 4, Pages 61-68.
[22] Mikael Laurson Mika Kuuskankare Vesa Norilo, (Spring 2009) *"An Overview of PWGL, a Visual Programming Environment for Music"ProjectMuse*-Computer Music Journal - Volume 33, Number 1, pp. 19-31.
[23] Lazzarini, V., J. Timoney, 2008, "New Perspectives on Distortion Synthesis for Virtual Analogue Oscillators". *Submitted to Computer Music Journal.*.
[24] Rodet, X. 1984. "Time Domain Formant-Wave-Function Synthesis". *Computer Music Journal*, 8 (3):9-14.
[25] Alexander Müller and Rudolf Rabenstein ,(September 1-4, 2009) "PHYSICAL MODELING FOR SPATIAL SOUND SYNTHESIS "12th Int. Conference on Digital Audio Effects Como, Italy, pp:1-8
[26] Karjalainen, M. (July 2008) " Efficient Realization of Wave Digital Components for Physical Modeling and Sound Synthesis" IEEE Transactions on Audio, Speech, and Language Processing,Volume: 16, Issue: 5 On page(s): 947 - 956 .
[27] Bilbao, S. (March 2007) "Robust Physical Modeling Sound Synthesis for Nonlinear Systems "Signal Processing Magazine, IEEE ,Volume: 24 Issue: 2
On page(s): 32 - 41
[28] Vesa Välimäki Henri Penttinen, Jonte Knif Mikael Laurson Cumhur Erkut , (January 2004)"Sound synthesis of the harpsichord using a computationally efficient physical model" EURASIP Journal on Applied Signal Processing, Volume 2004 , Pages: 934 – 948.
[29] Trautmann, L. Petrausch, S. Rabenstein, R," Physical modeling of drums by transfer function methods " Acoustics, Speech, and Signal Processing, 2001. Volume: 5 ,pp: 3385 – 3388.
[30] Jens Ahrens, Rabenstein Rudolf, and Sascha Spors, "The theory of wave field synthesis revisited," in *124th AES Convention*. Audio Engineering Society, May 2008, Paper number 7358.
[31] Alexandros Kontogeorgakopoulos and Claude Cadoz, "Cordis Anima physical modeling and simulation system analysis," in *Proc. SMC'07, 4th Sound and Music Computing Conf.*, Lefkada, Greece, July 2007, pp. 275–281.
[32] Panagiotis Tzevelekos, Thanassis Perperis, Varvara Kyritsi, and Georgios Kouroupetroglou, "A component-based framework for the development of virtual musical instruments based on physical modeling," in *Proc. SMC'07, 4th Sound and Music Computing Conf.*, Lefkada, Greece, 2007, pp. 30– 36.
[33] Stefan Petrausch, Rudolf Rabenstein, (September 8-11, 2003) "SOUND SYNTHESIS BY PHYSICAL MODELING USING THE FUNCTIONAL TRANSFORMATION METHOD: EFFICIENT IMPLEMENTATIONS WITH POLYPHASE-FILTERBANKS" Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03), London, UK,