Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00), Verona, Italy, December 7-9, 2000

A MODULAR REAL-TIME PC-BASED AUDIO PROCESSING TOOL FOR EFFECT DEVELOPERS, ENGINEERS, MUSICIANS, AND EDUCATORS

Yusuf Jafry, PhD

Sounds Logical, Leiden, The Netherlands
www.soundslogical.com
yjafry@soundslogical.com

ABSTRACT

A modular real-time PC-based audio processing software tool has been developed which offers a high degree of user control, intended for use by audio effect developers, signal processing engineers, sound designers, musicians, and educators. The key technical features are described, followed by a range of example applications in the field of signal processing and audio effect design.

1. INTRODUCTION

The disciplines of audio signal processing and computer music encompass a wide range of mathematical and engineering techniques which are often "hidden behind the graphics" of commercially-available audio manipulation software products. Whereas these products span a broad range of audio applications, they tend to be of a "black box" nature, and, thereby, are usually inappropriate for engineering prototyping or educational environments where flexibility, visibility, and controllability are the key requirements. For example, the developer, engineer, educator, or student wants to try out his/her own filter design and hear what it sounds like on audio material of their choice, without having to resort to writing their own code (at least not immediately!). "WaveWarp" is a software tool developed specifically for those applications where flexibility is essential.

2. SOFTWARE DESCRIPTION

2.1. Overview

The architecture of the software is modular, comprising a library of pre-compiled DSP components which can be connected together in any desired fashion (series, parallel, feedforward, and feedback). All components are processed sample-by-sample, enabling the design of sample-perfect signal flow networks in an intuitive "WYSIWYN" (what-you-see-is-what-you-need) manner. Moreover, the audio engine is intrinsically multi-rate, enabling on-the-fly integer-factor sample-rate conversion between components. This facilitates, for example, the rapid construction of elaborate polyphase filter networks, again in a straightforward "WYSIWYN" manner – a task which is notoriously cumbersome and error-prone when carried out "long-hand". Additionally, the software architecture is intrinsically multi-channel, enabling the straightforward construction of customizable "surround sound" designs. The software runs in real-time on a standard Pentium®-class PC, and all processing is fully "native", requiring no peripheral hardware except a Windows®-compatible sound-card. Multiple sound-cards and/or multi-channel sound-cards are fully supported. In addition to audio file and live I/O support, the software interfaces directly with other applications via industry-standard protocols (DirectX, etc), enabling seamless integration into existing computer studio/laboratory environments. The software also includes interfaces to the MATLAB technical computing environment, providing seamless access to powerful analysis, design, and visualisation capabilities.

2.2. Modular DSP component library

The main categories of modular signal processing components included in WaveWarp are listed in Table 1.

Table 1 *List of WaveWarp's modular processing component categories.*

CATEGORY	DESCRIPTION
Audio files	Audio files in WAV and ASCII format for use as data sources or sinks. Audio files can be played back with arbitrarily controllable sample ordering (e.g. for granular synthesis).
I/O devices	Windows-compatible soundcard drivers and DirectX ports for using sound-card I/O and 3rd-party editor/sequencer applications as data sources or sinks.
Basic connections	Basic connection components such as summers, multipliers, switches, etc., plus basic arithmetic components (which operate primarily on parameter control signals).

Delays	Digital delay components (including simple delay, feedback delay, reverse delay, controllable time-varying delay etc.)
Digital filters	Recursive (IIR) digital filters including Butterworth, Chebyshev, Inverse Chebyshev, & Elliptic designs; generalized 2nd order highpass, lowpass, bandpass, bandstop, peak & notch designs; all-pass designs; non-recursive (FIR) digital filters including windowed lowpass, highpass, bandpass, bandstop designs, in both direct and fast (FFT-based) implementations. Most filters include an ASCII file interface for implementation of off-line filter designs (e.g. via MATLAB).
Displays and scopes	Real-time digital displays, oscilloscopes, and spectrum analysers.
Distortion	Non-linear amplitude distortion and wave-shaping components.
Dynamic range controllers	Compressors, expanders, limiters, and noise gates, plus the basic blocks for building customized dynamic processors.
Flangers and chorus	Flanger and chorus components plus the basic blocks for building customized flangers and chorus (based on time-varying modulated delays).
MATLAB	MATLAB-enabled components which interface seamlessly with the MATLAB environment, utilising MATLAB for design, visualisation, and real-time processing. Components include FIR and IIR digital filters, "MATLAB in the loop" real-time processors, oscilloscopes based on MATLAB GUI's, MATLAB-based signal and envelope generators, etc.
Mixers	Multi-channel mixers.
Multirate	Integer-factor down-samplers, up-samplers, decimators, interpolators, and filterbanks.
Noise reduction	Noise reduction components (based on spectral subtraction).
Panners	Panners (static and time-varying).
Phasers	Phaser components plus the basic blocks for building customized phasers.
Pitch shifters	Simple (time-domain) pitch shifters.
Reverbs	Reverb components plus the basic blocks for building customized reverbs.
Signal generators	Signal and envelope generators including sine wave, triangular wave, square wave, periodic and pseudo-random white noise and telegraph noise, chaotic sequences, impulse and pulse trains, ADSR envelope generators, etc., plus amplitude and/or frequency controllable oscillators. Many of the oscillators and signal generators include an ASCII file interface for importing off-line wavetable and envelope designs (e.g. from MATLAB).
Spectral transformers	Frequency-domain effects such as convolution, spectral cross-synthesis, spectral shaping, morphing, etc.

3. EXAMPLE APPLICATIONS

3.1. Example: reverberation based on a random FIR filter

The schematic in Figure 1 depicts an artificial reverberator (room simulator) based on an exponentially-decaying pseudo-random FIR filter. The design, adapted from [2], is novel in the fact that the FIR filter taps are chosen randomly rather than from measured room responses or geometrical ray-tracing models. The FIR filter represents the "early reflections segment", with a feedback delay path ("around" the FIR filter) to create the dense reverberant field. Low-pass filters are added in the forward and backward paths to represent the absorption in the air (found to significantly improve the subjective quality of the reverberant effect [2]). Figure 2 contains a screenshot of the WaveWarp implementation of this reverberator algorithm. Again, the layout closely resembles the corresponding block diagram (Figure 1), thereby illustrating the convenience of WaveWarp's modular architecture. In this example, all the filters are designed in MATLAB then imported to WaveWarp via a seamless interface, thereby combining the powerful design capabilities of MATLAB with the high-speed real-time audio processing capabilities of WaveWarp. As noted in [3], MATLAB is too slow for practical computation of the large convolutions typically required for realistic reverberation (and other computationally intensive signal processing algorithms), especially at the high sample rates required for professional audio. WaveWarp, on the other hand, is designed specifically for such purposes.

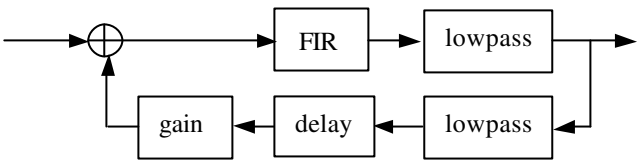

Figure 1 *Schematic of the reverberation algorithm adapted from [2].*

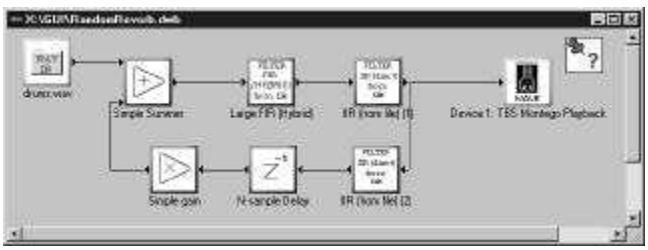

Figure 2 *WaveWarp implementation of the reverberation algorithm depicted in* Figure 1. *The FIR filter has 8092-taps, the feedback delay is equal to the FIR length, and the IIR low-pass filters are first-order. In this example, all filter designs were carried out in MATLAB then imported to WaveWarp for execution in real-time at 44.1 kHz with zero latency. The input signal in this case is a stored audio file in WAV format ("drums.wav").*

3.2. Example: "MATLAB-in-the-loop" real-time processing

The screenshots in Figure 3 illustrates the use of WaveWarp's seamless interface to the MATLAB programming environment. Specifically, WaveWarp sends the real-time audio stream (in this case, from the WAV file "Wavewarp.wav") into MATLAB where it is processed in real-time (by any arbitrary algorithm written in MATLAB), then returned to WaveWarp for further processing or playback. Via this intuitive and versatile interface, it is straightforward to enter any valid MATLAB expression to define the desired real-time input-output relationship on a buffer-by-buffer basis. The example in Figure 3 utilises the built-in MATLAB function "Y=flipud(X)" which has the (entertaining) effect of reversing the audio (chunk-by-chunk). Naturally, this can be replaced by any other expression which makes use of MATLAB workspace variables, m-file scripts, m-file functions, and compiled mex functions (for speed).

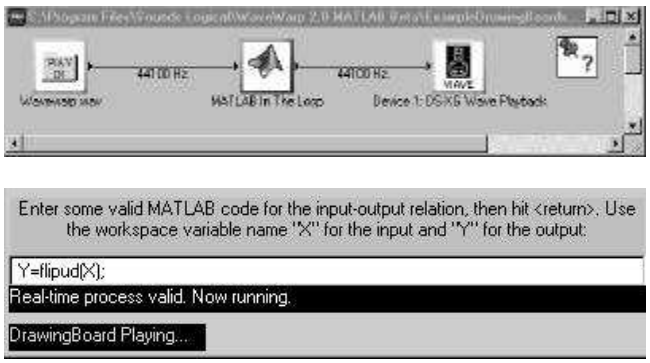

Figure 3 *Demonstration of WaveWarp's real-time "MATLAB-in-the-loop" capabilities. The "MATLAB In The Loop" component (in the upper panel) corresponds to a MATLAB GUI (lower panel) which enables the user to enter any desired input-output relation.*

3.3. Example: controllable audio file playback and "granular sythnthesis"

The screenshot in Figure 4 illustrates the use of WaveWarp's controllable audio file playback mechanism which enables the individual samples of an audio file to be played back in any arbitrary order (rather than in the usual manner of one after-the-other in succession). This enables interesting and elaborate effects to be achieved at low computational expense. For example, by selecting successive groups of samples (or "grains") and playing each group at a user-definable arbitrary rate and repeating the playback of each group for a user-definable arbitrary number of repetitions, a wide range of sounds can be synthesised from a single audio file.

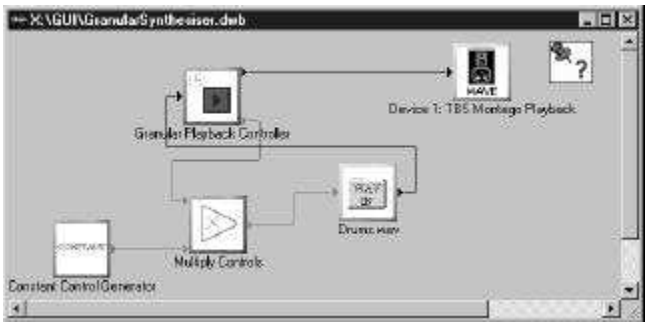

Figure 4 *WaveWarp implementation of a "granular synthesiser" built from scratch.*

3.4. Example: educational demonstration of aliasing

The example layout in Figure 5 is for educational purposes, illustrating the phenomenon of aliasing associated with digital sample-rate conversion. This example makes use of the multirate processing capabilities of WaveWarp whereby the sample rate can vary (by integer factors) throughout the network. In this case, the original (sine-wave) signal with a sample rate of 44100 Hz is down-sampled (by a factor of two) to 22050 Hz. Two down-samplers are compared for educational purposes: a simple one (upper branch of network) which merely omits every second sample, and a more elaborate one (lower branch) which contains a built-in Nyquist filter for alias protection. As can be observed in the spectrum analysers (lower panel), the alias suppression attained via the Nyquist filtering (right plot) is clearly observed (and audible) as a reduction in the 'spike' corresponding to the dominant aliased term.

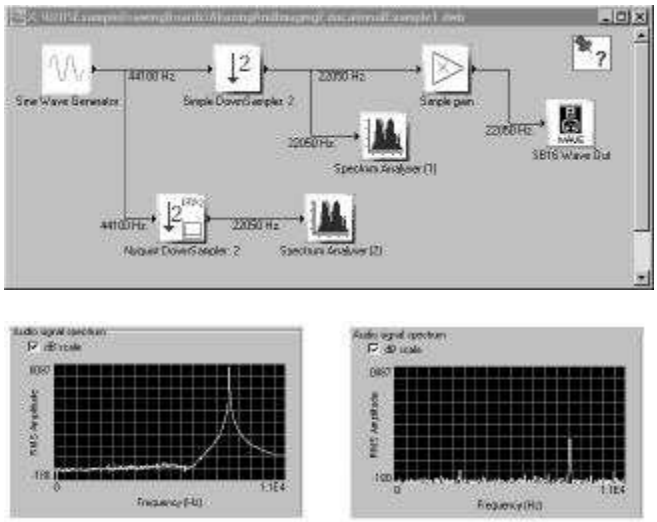

Figure 5 *WaveWarp demonstration of the well-known aliasing phenomenon.*

3.5. Example: educational demonstration of a two-channel filterbank

Figure 6 contains the block diagram of a two-channel "filterbank without filters" (adapted from [1]) which serves to emphasise that perfect reconstruction can be achieved from down-sampled data as long as all information in all channels (or "phases") is retained. In this example, the output is identical to the input -- except for the unit delay. This is achieved in spite of the fact that the signal is down-sampled then up-sampled by a factor of two. The key to the perfect reconstruction is that the "even" (upper branch) and "odd" (lower branch) "phases" are retained throughout.

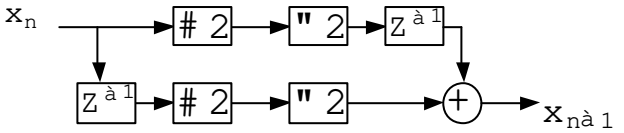

Figure 6 *An educational example of a two-channel "filterbank without filters" (adapted from [1]). In a real application, there would be filtering (and other processing) applied between the down- and up-samplers.*

Figure 7 contains a screenshot of the WaveWarp implementation of the two-channel "filterbank without filters" from Figure 6. Note the convenience of WaveWarp's modular architecture, whereby the processing components are connected together in an intuitive manner, closely mirroring the layout in the block diagram of Figure 6. Also note how the multirate capabilities of WaveWarp enable the signal(s) to be re-sampled (by an integer factor , in this case by two) at any point of the network.

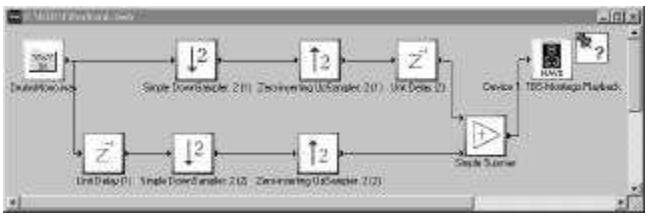

Figure 7 *WaveWarp implementation of the two-channel "filterbank without filters" from Figure 6. In accordance with expectations, the output is observed to be an exact replica of the input delayed by one sample, thereby demonstrating WaveWarp's "sample-perfect" multirate architecture.*

3.6. Example: four-octave-band equaliser

The screenshot in Figure 8 illustrates a practical use of WaveWarp's multirate processing capabilities to create a four-octave-band equaliser built from cascaded two-channel filterbanks in combination with with individual gains for each spectral band. For efficiency, the filterbanks used here are hard-coded components rather than built from scratch (as in the previous educational example). Specifically, they utilise IIR all-pass structures in polyphase form. WaveWarp has many such hard-coded components in addition to the basic building blocks.

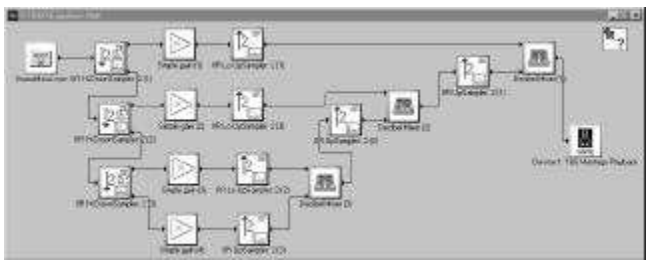

Figure 8 *WaveWarp implementation of a four-octave-band equaliser built from scratch.*

4. CONCLUSIONS

A PC-based audio processing software tool has been developed with the explicit purpose of providing a high degree of flexibility and control to the user. Examples of usage have been presented, focusing on applications in signal processing and audio effect design.

5. REFERENCES

[1] Strang, G. and Nguyen, T., "Wavelets and Filter Banks", Wellesley-Cambridge Press, 1996.

[2] Rubak, P. and Johansen, L.G., “Artificial reverberation based on a Pseudo-random Impulse Response II”, Paper No. 4900(G6), AES 106th Convention, Munich, Germany, 1999.

[3] Beltran, F. A., Beltran J.R., Holzem, N. and Gogu, A., “Matlab Implementation of Reverberation Algorithms”, Proc. Workshop on Digital Audio Effects (DAFx-99), Trondheim, Norway, 1999.