Audio Effect Applets for Music Processing

Mijail Guillemard, Christian Ruwwe, and Udo Zölzer
Helmut-Schmidt-Universität, 22043 Hamburg, Deutschland
Email: {mijail,christian.ruwwe,udo.zoelzer}@hsu-hh.de

Abstract

The increasing availability of inexpensive computer power allows the possibility to create new educational tools for music processing. In this report we investigate the development of a java-based learning platform for digital audio effects. The main purpose of this framework is to provide a software environment easy to use, and designed for a first insight into the perceptual experience with basic signal processing concepts. Among the multiple technologies that can be used for implementing this tool, we look for the one that better fits the criteria of easiness in implementation, usage, and deployment. An important aspect we consider is to avoid platform dependencies, allowing to reach a maximum number of potential users. We explain the reasons for selecting a java-based solution, and how this option fulfills the special constraints we have for this framework. In this report we will shortly explain two aspects of the implementation decisions. On the one hand, an important goal is to provide a simple to use graphical user interface, that presents nevertheless the main relevant parameters for a particular audio processing algorithm. On the other hand, we explain the acoustic aspects of the algorithmic implementations (focusing in particular on the well known fast convolution routine).

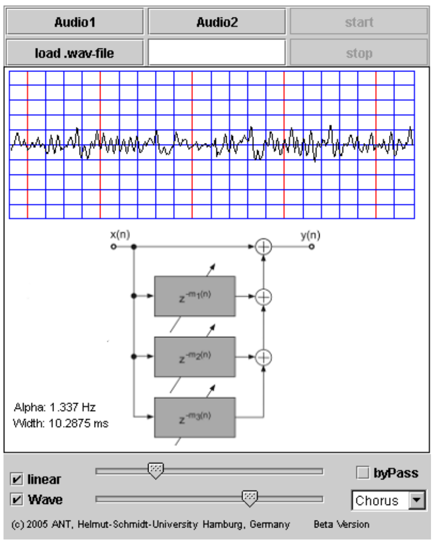
Introduction

Understanding the basics of acoustic effects is a task that requires to grasp theoretical and experimental concepts. When learning the technical background of digital audio effects it is useful to experiment with these concepts in a practical environment. In a previous report [1], we presented an educational software platform for experimenting with basic signal processing algorithms. The main constraints we have for this tool, are the easiness in implementation, usage, and deployment. In order to avoid platform dependencies the solution we proposed is to use the java language. We prepared a set of java applets presenting basic ideas on filters, psychoacoustic, nonlinear distortion, dynamic range control, and delays effects. In this note we focus on the technical acoustical aspects of some of these applets. In the following section we give a short description of the delays applet, which includes effects as the vibrato, chorus, and flanger. We then present an overview of two applets dealing with recursive and nonrecursive filters. In the case of nonrecursive filters we present an illustrative room simulation using the fast convolution algorithm.

Delays Applet

The delays applet presents some effects based on the concept of a modulation function. The tremolo effect is presented as a first algorithm in the delays applet. The idea is to apply an amplitude modulation to an input signal: $y(n) = m(n)x(n)$. In this case the modulation function has the particular form $m(n) = 1 + a\sin(\omega_0 n/f_s)$, where f_s is the sample rate. Two important parameters are the modulation frequency ω_0, and the modulation depth a. These parameters can be controlled with two sliders in the applet interface. In order to achieve a perceived variation in amplitude, the modulation frequency controlled by ω_0 does not exceed 20 Hz [2, 4].

The second algorithm in the delays applet is the vibrato effect. In this case a delay modulation function is applied to the input signal: $y(n) = x(n - m(n))$. Here the modulation function is given by $m(n) = M + a\sin(\omega_0 n/f_s)$. In this algorithm the fixed delay M can be controlled with one of the sliders, standard values are in a range up to 5 ms. As m is a noninteger valued sequence, a linear interpolation routine is used for computing the sample values. The modulation frequency in this implementation does not exceed 5 Hz.



Alpha: 1.337 Hz
Width: 10.2875 ms
[✓] linear
[✓] Wave
[ ] byPass
Chorus
(c) 2005 ANT, Helmut-Schmidt-University Hamburg, Germany Beta Version

Figure 1: The chorus effect in the Delays Applet

The combination of several delay lines is the concept used in a chorus effect. In this applet we use three delays lines: $y(n) = x(n) + \sum_{i=1}^{3} x(n - m_i(n))$. The modulation functions are given by $m_i(n) = M_i + (r_i * h_{LP})(n)$, where r_i are random variables, and h_{LP} is a low pass filter. The idea is to simulate three musicians playing together in unison. In a real-life situation a perfect synchronization is not possible, a small non-constant delay can be perceived. This nondeterministic behavior is simulated with the filtered random variables $r_i * h_{LP}$. In this applet, h_{LP} is a first order low pass filter parametrized by a value α. One slider is used to control the parameter α, and the other slider is used to control the fixed delays.

The flanger effect is our fourth example in the delays applet. In this case we add to the input signal a time-varying reflection: $y(n) = x(n) + x(n - m(n))$. As in the case of the vibrato, $m(n)$ is controlled with the graphical interface via the modulation frequency and the modulation depth.

Equalization Applet

Several second-order recursive filter structures are implemented in the equalization applet. These include low and high pass filters, shelving filters, and peak filters. The objective of this applet is to control the filter response by easy and independent adjustment of the width, center frequency, and filter gain. We use the parametrizations of the filter coefficients described in [3]. For the case of the low and high pass filters, a single parameter controls the coefficients via the center frequency. This can be adjusted with the horizontal slider of the applet.

The case of the shelving filters requires an additional parameter for the cut or boost factors. For peak filters, a third parameter controls the filter bandwidth. In the case of shelving and peak filters, the algorithm implementation considers two parametrizations depending on the cut and boost cases. Two vertical sliders can be used for controlling the filter bandwidth, and filter gain factors.

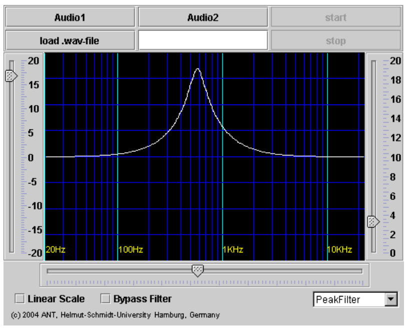


Figure 2: The peak filter in the Equalization Applet

Fast Convolution Applet

The fast convolution applet presents an illustrative application of nonrecursive filters. Given a finite impulse response h the objective is to compute the convolution operation $y(n) = \sum_{i=0}^{N-1} h(k)x(n-k)$. In this applet we implement a fast convolution scheme for long sequences. As described in [3], this procedure consists of partitioning the input sequence x as well as the impulse response h. The convolution of the partial sequences is computed in the frequency domain with the FFT and inverse FFT. An overlap-add procedure is then used for assembling the resulting convolution sequence.

This applet uses the fast convolution scheme for a simple room simulation experiment. We simulate a room impulse response by modulating the amplitude of a white noise signal. The modulation of the noise signal can be manipulated by dragging three control points in the graphical interface. Two important components of a room impulse response are the first reflections and the subsequent reverberation. Two control points can be used for modifying the first reflections. The subsequent reverberation is simulated with the exponential decay controlled by the third control point. The fast convolution scheme for long sequences is then applied, and different room characteristics can be obtained by dragging the control points.

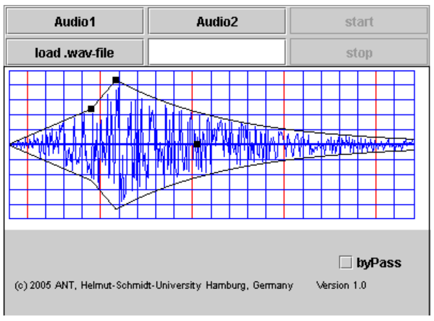


Figure 3: An illustrative room impulse response

Conclusion

We have presented an educational software tool for audio effects. Its main objective is to be used as a first experience with digital audio processing algorithms. A java platform has been selected in order to provide a system easy to use and install. Several applets have been implemented demonstrating basic concepts on recursive filters, room simulation, and delays effects.

The focus of this note is on a short technical description of some of these applets. We include a general review of the main algorithm parameters, and their corresponding controls in the applet graphical interface. These effects can be used online through the internet, or offline after downloading and proper installation. The source code is released under the GNU license, and can be accessed at `http://ant.hsu-hh.de/jdafx`.

References

[1] M. Guillemard, C. Ruwwe, and U. Zölzer, j-DAFX - Digital Audio Effects in Java. *Proceedings DAFx 2005*, pp. 161-166.

[2] J. Dattoro, Effect design, part2: Delay-line modulation and chorus. *J. Audio Eng. Soc.*, 45(10):764-788, October 1997.

[3] U. Zölzer, *Digital Audio Signal Processing*. John Wiley & Sons, 1997.

[4] U. Zölzer et al, *DAFX - Digital Audio Effects*. John Wiley & Sons, New York, 2002.