Patch for guitar

Miller Puckette
Center for Research in Computing and the Arts
University of California, San Diego
msp@ucsd.edu

ABSTRACT

Now that it's easy to get multiple channels of sound into and out of a computer with pretty low throughput latency (thanks partly to the well-designed linux operating system), an obvious and attractive application is signal-processing an electric guitar separately string by string. This permits a wide variety of non-linear processes which, if properly designed, preserve the periodicity of the individual string while making possible a wide variety of new sounds. Pitch-synchronous algorithms also become available. It is also possible to separate the effect of amplitude change from that of changing harmonics, thus preserving the playability of the instrument.

1. INTRODUCTORY POLEMIC

Although computer music overall has seen great advances in the past fifty years, in one area, using computers in live instrumental music performance, our understanding is still crude. The main advances in live computer music performance have largely consisted of building an infrastructure, so that now a musician can combine a computer, audio and control hardware, and software (Pd being one possibility) to make a live interactive computer music application. But visits to a club or concert hall reveal that the computer is mostly used as a recording and/or sequencing device, rather than as a musical instrument. The main payoff of computing in music performance has been to reduce and replace the role of musicianship.

If playing a musical instrument and/or singing were pure drudgery this would be all to the good, but it seems likely that much of musical knowledge is built up through the physical act of making music in time. Musical cultures, at least up to now, have been at least partly maintained in a performance tradition, and even composers (all but the most theoretically inclined) rely on the sort of musicianship that is learned and transmitted through music making. And even though there is clearly much music to be made using studio techniques and/or sequencers, there is also much to be made instrumentally. The computer is very handy for the former, but is still only with difficulty applied to the latter.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Pd07, Montreal, Quebec, Canada
Copyright 2007 Copyright remains with the author(s).

2. GUITAR PROJECT

With these thoughts in mind, I've been at work on a long-term project to design a rather personalized computer music instrument to try to bring out and confront some of the difficulties encountered by musicians trying to use computers in live performance. The instrument is based on a compact electric guitar (Steinberger/Gibson) with an added six-string separated pickup (Roland). Not finding an inexpensive and compact 6-channel preamp on the market, I designed and built a very crude one. This is interfaced to a computer using a multichannel PCI interface (Midiman). A Pd patch, running in linux, then performs a variety of interesting transformations on the six audio signals, and mixes them to stereo for output.

This is entirely different from standard "guitar synthesizers" which pitch track the strings to drive synthesizers. Such instruments make lots of audible mistakes, and they also suffer from the added latency the comes from the pitch tracker. In the instrument described here, the latency of the whole affair is only that of Pd itself, about 10 milliseconds (it's probably not hard to reduce it to 5 or 6 using real-time kernel patches but I preferred to use off-the-shelf linux).

The rest of this paper describes the design of the patch as it now stands, starting with the overall block diagram and control strategies, then describing some novel waveshaping tactics used.

3. ORGANIZATION AND CONTROL STRATEGY

Figure 1 shows a block diagram of the audio chain (with each individual string going through a separate copy of it). Individual strings are analyzed both to estimate their individual fundamental frequencies (using the `sigmund~` object) and to detect attacks (using `bonk~`). The current version of `bonk~` uses a 256-sample window; to improve robustness this is run at half sample rate—22050Hz if the patch runs at 44100—giving an analysis window of about 11.6 msec instead of the usual 5.8, at the expense of slightly more delay. The analysis delay only affects the updating of control parameters; the signal path itself is not delayed by the analyses.

The signal processing chain has one novelty, a waveshaping algorithm designed to allow: (1) decoupling of amplitude effects from the amplitude of the original signal; (2) replac-

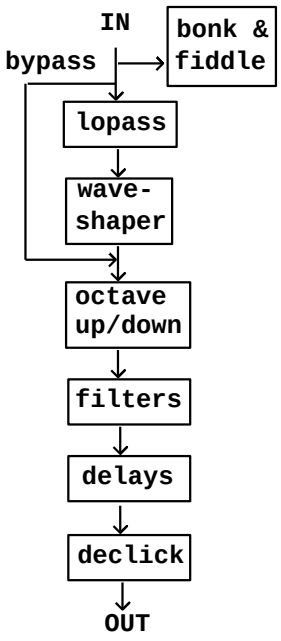

Figure 1: Block diagram.

ing a sinusoid with a stored wavetable; and (3) specification of formants in the same way as in the PAF synthesis technique.

The signal processing chain has about fifty parameters per string, 9 of which are controlled by ADSR envelopes triggered by the string via the `bonk~` object. The ADSR parameters, as well as the remaining processing parameters, are organized into presets. Each string may have a different preset. The presets may be chosen statically or may change during play as a function of the timing and order of detected attacks on the various strings. If attacks are used to trigger the recall of presets, the presets are changed as early as possible during the attack, making it necessary for presets to be recalled very quickly.

To manage this, and to deal with the fact that certain parameter changes (such as delay times) cause discontinuities in the signal anyway, a switch-and-ramp unit [2, Section 4.3.2] is placed at the end of the signal processing chain, activated whenever a "clicking" parameter is changed, and thus *a fortiori* when a preset is recalled.

4. WAVESHAPING ALGORITHM

Waveshaping has been around for many years [1], but it proves difficult to use on real signals for two reasons: first, the timbre of the output depends on the amplitude of the input. (Although such a dependency is sometimes desirable, it should be a matter of choice, not a given). Second, the amplitude of the output can vary capriciously with the input amplitude. These problems can both be partly circumvented by treating signals in the complex plane [3], replacing the incoming signal by a pair of signals in 90-degree phase quadrature. In this paper we take this idea one step further by treating the amplitude and phase of the complex-valued signal separately.

Suppose we're given a signal,

$$x[n] = a \cdot \cos(\omega n)$$

and we wish to process it to result in the third harmonic, $\cos(3\omega n)$. If we happen to know the amplitude a (suppose it's $a = 1$) we can use waveshaping (nonlinear distortion):

$$z[n] = f(x[n])$$

with the waveshaping function f equal to:

$$f(r) = 4r^3 - 3r$$

and using elementary trigonometry we end up with:

$$f(1 \cdot \cos(\omega n)) = \cos(3\omega n)$$

So we get the third harmonic as desired. Furthermore, f is essentially the only function that will do the job. But now consider the effect if a is different from 1. For higher values the leading r^3 term dominates, so that if $a = 10$ the output has peak amplitude 1000. And for lower ones the leading term quickly disappears, so that when $a = 0.1$ we end up with $f(r) \approx -3r$ so we mostly hear the fundamental instead of the desired third harmonic.

The fix is to start by replacing the incoming signal by a pair of signals in 90-degree phase quadrature (using the `hilbert~` abstraction):

$$x[n] = a \cdot \cos(\omega n)$$

$$y[n] = a \cdot \sin(\omega n)$$

from which we can extract time-varying estimates for the amplitude and phase:

$$a[n] = \sqrt{x^2[n] + y^2[n]}$$

$$\phi[n] = \text{Arctan}(y[n]/x[n])$$

To generate an output with the "correct" amplitude but any desired waveform $t(\phi/(2\pi))$ for $0 \le \phi/(2\pi) \le 1$, we can just output computed values of the expression:

$$z[n] = a[n]t(\phi[n]/(2\pi))$$

as shown in block diagram in Figure 2.

Figure 3 shows one example of a suitable waveform family to use with this technique. The parameters s, t are the slopes of the rising and falling segments and the parameter d controls the duty cycle of the waveform. With suitable values of the parameters this can give triangle, sawtooth, or rectangle waves.

Another possibility is to build formants using the PAF generator [2, Section 6.4], for which the function $t(\phi)$ is given by:

$$t(\phi) = c(\phi)m(\phi)$$

where the carrier function $c(\phi)$, given by

$$c(\phi) = (1 - q)\cos(k\phi) + q\cos((k + 1)\phi)$$

sets a center frequency equal to the fundamental times $k+q$, with k an integer and q a fraction between 0 and 1. The modulator function is set to

$$m(\phi) = e^{-(g\sin(\phi/2)^2}$$

sets a bandwidth approximately g times the fundamental. In both the PAF and the line-segment waveforms, the parameters may be attached either to envelope generators triggered

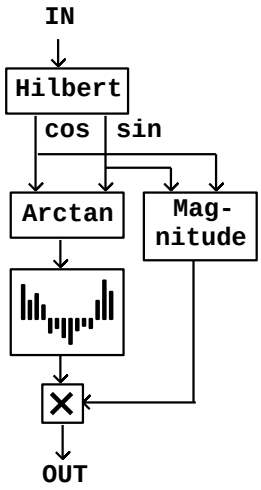

Figure 2: The waveshaping technique in its simplest form.

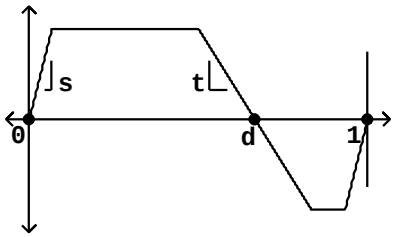

Figure 3: Waveform for use as output of waveshaping technique.

by note onsets, or they may be functions of the measured amplitude a.

Alternatively, one can simply make a series of separate wavetables for the first several harmonics and control their amplitudes explicitly as in additive synthesis.

4.1 Harmonics in the input signal

The input signal in reality is not a sinusoid, so it is worth considering what happens when the waveshaping techniques above are used on other signals. If the signal is periodic (roughly true here since the strings are picked up separately), we can assume its output is a sum of sinusoids of frequency ω and its harmonics. Since the "Hilbert" filter pair is linear, its output is a sum of sinusoids, in phase quadrature, with the same amplitudes and frequencies as the input.

If for any reason one of the sinusoids has much greater amplitude than the others (for example, more than twice the sum of the others), then we can approximate the extra signal as a perturbation. For example, suppose the input signal, after the Hilbert filter, is:

$$x[n] = \cos(\omega n) + a\cos(\xi n)$$

$$y[n] = \sin(\omega n) + a\sin(\xi n)$$

with $a << 1$. Then we get,

$$a[n] \approx 1 + a \cdot \cos((\xi - \phi)n)$$

$$\phi[n] \approx 2\pi k + \omega n + a \cdot \sin((\xi - \phi)n)$$

(The integer k is an arbitrary phase wrap number that drops out on applying the wavetable.) Inspecting the result we conclude that summed-in, low-amplitude sinusoids simultaneously modulate the amplitude and phase of the process.

Highly motivated readers might want to check that, if we use the lookup table

$$t(\phi/(2\pi)) = \cos(\phi)$$

so that in theory we reconstruct the signal $x[n]$ perfectly, then to order a there are two sidebands, one at which the phase modulation and amplitude modulation effects cancel each other, and the other of which reconstructs the perturbing sinusoid.

As a rough estimate, if the summed amplitudes of all the overtones is less than half the strength of the fundamental, the above approximation will hold reasonably well. At the other extreme, if the overtones actually exceed the fundamental, one sometimes sees the phase slip forward one or more extra cycles for a single cycle of the fundamental, with quite unpredictable results. Geometrically, this happens when the complex samples wind more than once around the origin of the complex plane.

In practice, it turns out to be most effective to filter the guitar strings individually, using a low-pass Butterworth filter, whose cutoff frequency for each string is chosen somewhere between the seventh and the fifteenth fret. For example, a fifth-order filter set to the seventh fret should attenuate the octave of the open string by about 15 dB, so that notes played anywhere up to the twelfth fret remain audible but the first harmonic should usually predominate. Bleed-through from the other harmonics then sounds as coherent modulation, so that timbral variation in the guitar playing comes through clearly in the final result. Of course, the filter

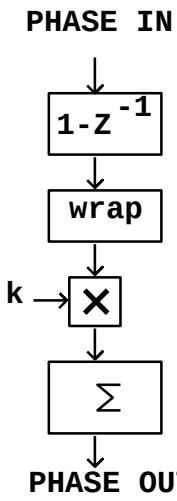

Figure 4: A phase manipulation requiring stored state: frequency multiplication by nonintegral k.

cutoffs may be set higher instead; in this case many interesting sounds come out but they are not so easily controlled or analyzed.

4.2 More waveshaping ideas

Two interesting details about the waveshaping algorithm. First, although most of the time in practice it is more interesting to process the strings separately, it is sometimes desirable to combine the signals of more than one string to produce intermodulation effects. A control in the patch allows to "bleed" the sounds of adjacent strings into each other. Nonadjacent strings are not allowed to intermodulate, in the spirit of keeping the result as predictable and as controllable as possible.

Second, the stage where the signal is in phase quadrature is a good occasion to apply any desired frequency shifting. Because the strings are pitch tracked, the frequency shift may be chosen as a multiple of the fundamental, in addition to a constant frequency offset in Hertz.

Except for frequency shifting, the waveshaping algorithms shown here have mostly been stateless, in the sense that each phase leads functionally to a (phase, amplitude) pair. A much larger class of potentially interesting algorithms opens up when we allow state (memory) to be part of the transformation. For example, although multiplying the frequency by an integer (doubling it, for example) is done by multiplying the phase by an integer, making fractional phase multiplication requires unwrapping, for instance as shown in Figure 4. Here we must take the first difference of the phase, "wrap" the resulting phase increment to lie between -1/2 and 1/2, multiply by the desired factor k, and then sum to recreate the new phase.

Another possible extension, not yet explored, is to allow the instantaneous amplitude to parametrize the waveshaping function. Doing this would not re-introduce the problem of unpredictable amplitudes cited earlier since we would still only be operating on the phase of the resulting signal, not its amplitude which would still be that of the input. As a simple example, one could have progressively higher amplitudes tune in higher partials using an amplitude-controlled formant generator.

5. AUDIO POST-PROCESSING

Next, the waveshaped sound is raised or lowered from -2 to +1 octaves, using the techniques of [2, Sections 5.2 and 7.10]. A continuous control effectively cross-fades between the four octaves available. Next, two "peaking" and one band-pass filter are applied in series; the three center frequencies and the two attenuation factors are controlled by envelope generators. Finally, a delay network is provided for pitch shifting, chorusing, or flanging.

6. CONCLUSION AND FURTHER WORK

This instrument has only been used once in public, with the Convolution Brothers at Metronom in Barcelona, for off-ICMC 2005. Since then, several enhancements have been made and the new system is overdue for another public trial.

Much more work needs to be done in finding intelligent ways to vary the processing parameters as a function of instrumental phrasing. For example, one could imitate wind-instrument tongueing patterns as a function of the timing of repeated attacks; or one could loop through a small sequence of presets to allow phasing between note and timbre cycles; or one could enhance or suppress individual harmonics to enhance consonances or dissonances within chords, to name only three relatively simple ideas.

Improvement is needed in two areas. First, it is hard to keep the amplitudes of harmonics well-behavedly low and simultaneously allow notes high on the neck of the guitar to sound loudly; this is a tradeoff in the design of the lowpass filter. Second, the `bonk~` object, which was designed for percussion instruments, should be tweaked in order to make it work better for non-percussion instruments such as this one.

7. REFERENCES

[1] M. Lebrun. A derivation of the spectrum of FM with a complex modulating wave. *Computer Music Journal*, 1(4):51–52, 1977.

[2] M. S. Puckette. *The Theory and Technique of Electronic Music*. World Scientific Press, Singapore, 2007. `crca.ucsd.edu/~msp/techniques.htm`

[3] T. Schouten. Complex wave shaping. In *PD Convention*, 2003. `puredata.org/community/projects/convention04/lectures/tk-Schouten/ComplexWaveshaping.pdf/`