

Tape delay emulation in Max/MSP

Sky Frostenson

Final paper for Musical Applications of DSP, Winter 2004

On many occasions over many years I have tried to find a way to emulate the tape delay effects used by Jamaican dub producers like Scientist and King Tubby. While it was never enough of an obsession to drive me to purchase an actual effects unit, I have been pricing them (vintage Roland Space Echo RE-201's are currently going for about \$300 on Ebay), and I have also been constantly on the lookout for free or reasonably priced VST plugins that claim to emulate the Space Echo or Echoplex effect. Almost all of the hardware and software emulators I came across seemed to cost as much if not more than the actual vintage units, which prompted me to wonder about the issues involved in emulating these devices. I should note that I don't typically suffer from analog or retro fetishism, but the case of tape delay devices seemed like it would serve as a good case study for this particular assignment.

In order to find out how the actual devices work, I consulted an electrical engineering graduate student friend of mine who researches DSP techniques and applications. I also searched through DSP mailing list archives to see if anyone had outlined a specific approach that might work in Max/MSP. From personal communication with my friend, I was able to sketch out a basic processing algorithm which went more or less like this:

input -> pre-emphasis filter -> soft-saturation -> tape filter + noise-> delay -> post-emphasis filter -> output

Since tape is better at recording some frequencies more than others, the pre-emphasis filter is used to boost desired frequencies while dampening others. Higher frequencies in particular are typically emphasized more in cases where the signal is saturated or distorted, because the harmonics for distorted high frequencies typically lie outside audible range. Distorted low frequencies and their harmonics are more easily perceived and are therefore typically de-emphasized to minimize the impact of distortion upon listeners (unless the distortion is intentional). As the tape records and re-records over the same media, the signal becomes saturated as the tape is less and less able to absorb electromagnetic charge, which can cause the signal to begin to approximate a square wave over time. A soft saturation algorithm and a "tape filter" would attempt to reproduce these physical phenomena in a digital signal. Noise would also be introduced to the signal at this point, in an attempt to simulate the effects of dust or other impurities on the recording and playing heads. The delay is (obviously enough) the temporal length of the signal delay, which in the case of the original device is determined at least in part by the length of the tape loop. Lastly, the signal is run through an inverse or post-emphasis filter in an attempt to restore the originally de-emphasized frequencies before output.

Researching various DSP mailing lists, equipment manuals, online forums, and whatever

other sources I could find all suggested that I was at least on the right track. It looked like the basic algorithm was correct, but as far as exact technical specifications went, I would be left to my intuition. The closest I came to any technical information specifically regarding my target devices came from a discussion on the music-dsp list¹ where a contributor mentioned, “Years ago talked to the old engineer who invented the Echoplex. He said the upper mids are deliberately boosted, and the lows and highs deliberately attenuated, to 'make it sound good on guitar.'” The author also happened to be discussing the issue of wow and flutter in tape devices, which I hadn't yet thought about emulating. The consensus there, as I gathered from the discussion, was that the best way to emulate wow and flutter would be to modulate the signal with a randomized low frequency oscillator. If one were to make an attempt at even more authentic emulation, one might try to relate the wow parameters to the circumference of the capstan (the cylindrical piece that drives the tape) versus the tape speed.

At this point I began prototyping a model in Max/MSP, setting up a basic buffered delay loop using `groove~`, `tapin~` and `tapout~`. Next I added a low pass filter so the signal would be filtered through every iteration of the loop, and then an amplifier to allow some percentage of the signal to be attenuated each time through. Surprisingly, even with a setup as simple as this, I began to get the “thin” sort of receding echo effect I was looking for.

As I tested the patch with assorted sound files, and then as I began to test it with straight microphone input, I realized I was basically on my way to building a realtime delay effect processor, with no need to confine it to the simple role of a tape delay emulator. I was having much more fun playing with the signals, letting them slowly die out and then bringing them back with the resonance parameter in the filter, observing when I began to get aliasing effects, and seeing how long I could tweak the settings before things got too ugly to listen to – at which point I would have to start again with a fresh sound. I remembered that this sort of hypnotic effect was exactly what made me want to emulate dub production sounds in the first place. I also realized that while the original devices had inspired me, I was already making something that could do infinitely more than the old devices ever could. At this point my focus began to shift away from making a faithful emulation, and more toward making a tape delay-inspired processing patch, possibly for use in broadcasting or performance.

I began to focus more on the interface, in order to make it easier to load, use and manipulate the sounds. I added sliders for all the parameters I had up to that point – filter cutoff, filter resonance, attenuation, and noise. It came to my attention that implementing the “noise” part of the algorithm wasn't really doing much, other than polluting and overpowering the whole signal. Also, I think the signal was getting enough noise from my laptop's hardware and all the faulty connections all the way to the stereo – no need for any artificial addition at this point. I would have to either come up with a more precise way to soil the signal (or give it warmth, or whatever the preferred term might be), or be content with the ambient noise.

1 <http://aulos.calarts.edu/pipermail/music-dsp/2002-September/017513.html>

In the process of dealing with noise I realized that many of the individual steps in the algorithm I had sketched out might be too delicate or too extravagant for my purposes anyway, given the hardware and source material I was working with, and the potential payoff. Whole DSP mailing list threads had been devoted to arguments about the best way to implement many of the features I was considering, and at the end of most of them, the participants in the discussion admitted that it was usually hard to tell the difference anyway. These people were electrical and recording engineers, probably using the latest and best sound hardware (or at least using something better than mine), and so I reasoned that if it didn't make much difference to them, then it certainly wouldn't make much difference to me, if I could get away with quicker and dirtier techniques that approximated the effects I was going for.

This became the case with the soft saturation effect. After looking extensively for a ready-made algorithm for soft saturation that I might be able to easily plug into my patch, I found exactly one. The code was a little cryptic, and early attempts at plugging it in didn't seem to work. While searching for the code, however, I also came across other discussions about saturation and nonlinear transfer functions, and I came to realize that it might just be basic enough that Max/MSP would already have an object for that purpose. When I came across `overdrive~`, it looked like I was right. I could be wrong here, but my understanding is that soft saturation isn't really any different from other distortion or overdrive effects, except for the fact that it's supposed to happen more slowly (as more of a side effect than an intended one). So, I reasoned, as long as the user applies it slowly, `overdrive~` should work for my purposes – and it seems as if this is the case.

I didn't do any strictly scientific studies on the pre- and post-emphasis filtering effects, but in the spirit of approximating effects I decided that a single state-variable filter seemed to come close enough, and in any case I liked the effect of being able to switch between any or all of the filter states at any time, so I left it in. As for the idea of emphasizing certain frequencies over others, I didn't even notice enough of a difference during my own tests between low pass and high pass filtering effects to justify looking up the official RIAA specifications.

Another issue, and the only real problem, as I see it, was the aliasing effects I would get when distorting the signal too hard or quickly. To a certain extent it seemed that some clipping and rectification was the effect I was going for, and in some cases I even liked the results of the aliasing, provided it could be toned down (or out) quickly enough. I made attempts to tune the adjustable parameters and to set the sliders up in such a way that hard and fast distortion at least won't be the first thing a user does, but I may have to do more in this area. I did some research regarding aliasing filter implementation but I suspect that here I will also have to do more. To a limited extent I justify the issues of aliasing and exaggerated distortion by noting that every performance instrument has its flaws, and its own peculiar ways of producing excruciatingly painful or irritating sounds, so if my patch were to be used in a performance context, why should it be any different? Still, I recognize that it doesn't hurt to strive for perfection.

The final problem which arose (but which I was able to remedy) was the issue of being able to change the delay time in real time without clearing the delay line. After

discussion with the course instructor and some quick experimentation I realized that by having two tapout~ objects attached to the same tapin~ object, I could modify the delay time of one or the other without affecting the overall delay time. For some reason in my mind this raised certain associations with turntablism, so I arranged the delays and the rest of the parameter sliders in a mixer orientation, with a crossfader between the delays. This, I found, allows for some very interesting and dynamic multitap delay effects. This feature especially makes the prospect of extending the patch to a real-world controlling device seem promising. In keeping with the crossfader theme, I also added a crossfader between the file source and the analog-digital converter source. I did this mainly because I found it useful during testing, but I would ultimately like to be able to include even more sources and have some sort of mechanism available for easy switching and mixing among various sources.

I would very much like to extend this patch into a larger scheme of control arrangements for non-orthodox realtime sampling and mixing from and between any variety of media sources. The relative ease with which I was able to construct a reasonably complex processing system (at least I think it's reasonably complex) seems to suggest that it may well be possible to build even further on this idea, to the point where one could feasibly build a system that reliably and dynamically mixes itself with minimal intervention. My ultimate goal in this respect is to build a semi-autonomous 24/7 metamedia mixing system which will broadcast (or at least netcast) worldwide. With this simple patch I may have taken the first step.

References:

Barbati, Stefania, and Serafini, Thomas; A Perceptual Approach on Clipping and Saturation; <http://www.simulanalog.org/clip.pdf>.

Dodge, Charles, and Jerse, Thomas A.; Computer Music: Synthesis, Composition, and Performance; Schirmer Books, 1997.

Gier, Hermann; Machine Head Model 9737: Digital Tape Saturation Processor: Manual. Sound Performance Lab, 1999.

Harmony Central: Effects: Effects Explained; <http://www.harmony-central.com/Effects/effects-explained.html>.

<http://www.sweetwater.com/insync/word.php?find=TapeDelay>

Music-DSP mailing list archives: <http://shoko.calarts.edu/~glmrboy/musicdsp/musicdsp.html>& <http://aulos.calarts.edu/pipermail/music-dsp/>

[The Dub Scrolls - A collection of Dub production techniques; www.interruptor.ch/dub.html](http://www.interruptor.ch/dub.html)