Synthesizing Trills for the Chinese *Dizi*

Lydia Ayers
Computer Science Department, Hong Kong University of Science and Technology
email: layers@cs.ust.hk

Abstract

The *dizi* is a Chinese transverse flute that produces a characteristic nasal buzzing tone. This project uses frequency modulation with a function table to make realistic *dizi* trills that sound better than with the overlapping or line segment methods, and the new method is easier to use. We used one function table for frequency modulation and another for amplitude modulation.

1 Introduction:

The *dizi* is usually made of bamboo, with a cane membrane over one hole (see Figure 1) that produces a characteristic nasal buzzing tone rich in upper partials. This project uses frequency modulation with a function table to make realistic *dizi* trills.

Figure 1. *Dizi*

We studied the musical characteristics of *dizi* trills to answer the question: What is the most accurate and most efficient method to synthesize those characteristics using the wavetable synthesis instrument design from our previous work? Is the method good enough to fool listeners into believing that humans played the synthesized examples?

After comparing the results, we concluded that trills modeled using the function table method sound better than with the overlapping or line segment methods, and the new method is easier to use. We used one function table for frequency modulation and another for amplitude modulation. Our method is highly successful and the trills sound realistic enough to fool listeners into believing that humans played our synthesized examples.

2 Background:

Previous research modeled the spectra of ordinary single tones of many wind instruments (Risset and Mathews 1969; Morrill 1977; Horner and Beauchamp 1995; Horner and Ayers 1998a and 2002). We previously designed a wavetable model of *dizi* tones that is easy to use and sounds like the original individual tones (Horner and Ayers, 1998; Horner, Ayers and Law, 1999). We used the same wavetable synthesis design to model the individual *dizi* tones in this project. Isolated synthesized tones may be indistinguishable from acoustic tones on listening tests, but musicians don't always play isolated tones! They connect tones to make "musical" performances. Since wavetable synthesis is straightforward and matching works well for isolated *dizi* tones, we will focus our discussion on modeling trilled tones.

Our previous work used the phase vocoder, which centers bandpass filters on the harmonics (Dolson 1986, Beauchamp 1993, Roads 1996, Wun 2001), to analyze the single *dizi* tones. We chose the spectra from the spectral snapshots of the original tone (Horner 1993). However, the phase vocoder, fails if a harmonic swings outside the range of its filter, and this can happen when the frequencies change as much as they do in trills.

3 Synthesizing Trills:

Overlapping can simulate slurs when the decay of one tone can overlap the attack of the following tone. But this technique doesn't work on trills because the notes are too short for any articulation, much less cross-fading, and they can even have clicks. Connecting together very short tones, therefore, is often an immediate give-away that the music is synthesized.

Connecting the frequencies of two notes with a line segment and morphing the transition using parameter interpolation (Rodet and Lefèvre 1997) obviously can give a smoother slur, but how can we slur varying numbers of notes in trills? In Csound, slurred groups of different numbers of notes need different numbers of input parameters. Using the maximum number of input parameters in every note statement, and just using placeholder numbers for those we don't need results in a design that is too

cumbersome to use for long trills. A whole note trilled quickly in a slow tempo could easily contain more than 32 individual notes! Can a repeating function controlling the frequency of a single tone produce better trilled notes than overlapping tones does?

We created a hybrid instrument which cross-fades two unison signals sharing one frequency line segment and phase, but using their correct wavetables (see Figure 11). Amplitude envelopes cross-fade the two signals as with overlapped notes, but it is smoother because the frequency change in the transition more closely resembles one on a real *dizi*, and using the same phase (and the other frequency parameters, such as noise and vibrato) also makes the cross-fading itself less noticeable. We also increased the noise in the cross-fade as the amplitude decreases, so that the maximum amount of noise is in the middle of the cross fade, at the point of minimum amplitude.

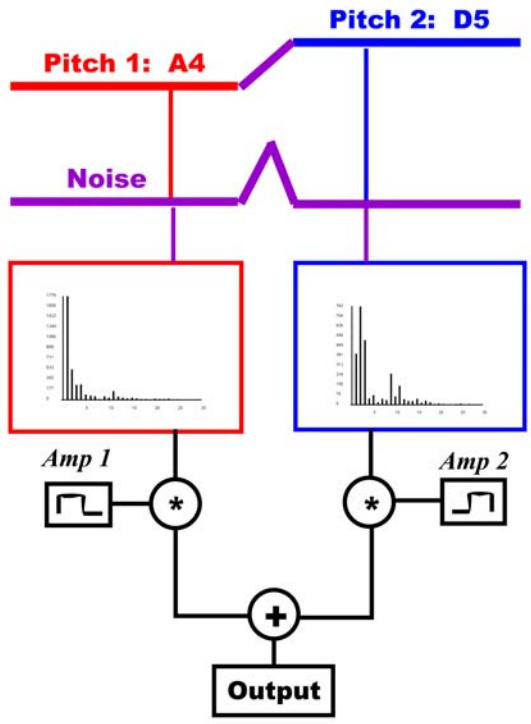

Figure 2. Cross-Fading Two Trilled Notes with Noise in Transition

Each note must get the correct wavetable, so our repeating function must fade two signals in and out together while it alternates their frequencies.

4 Trill Frequency Modulator:

What is the average repeating pattern that best represents the basic shape shown in this trill as well as others? The function does not need to model pitch variation of the average tone, change of speed or jitter, so it can represent one average cycle of the trill, and adjusting the parameters randomly within their typical ranges can vary each cycle.

The frequency modulator must use a function that can approximate the MQ (McAulay and Quatieri, 1986) frequency analysis graph of the trill shown in Figure 3.

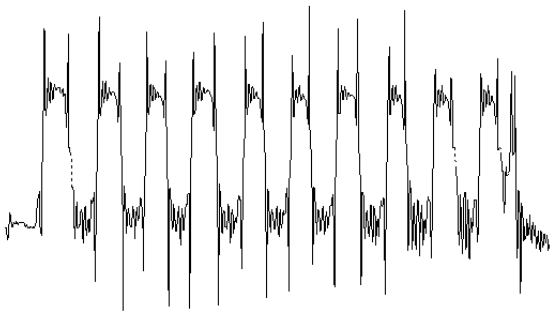

Figure 3. MQ Frequency Analysis of an A5 to B5 Trill

Figure 4 shows a close-up of two cycles of this trill. Averaging one cycle provides a good shape for a trill frequency function.

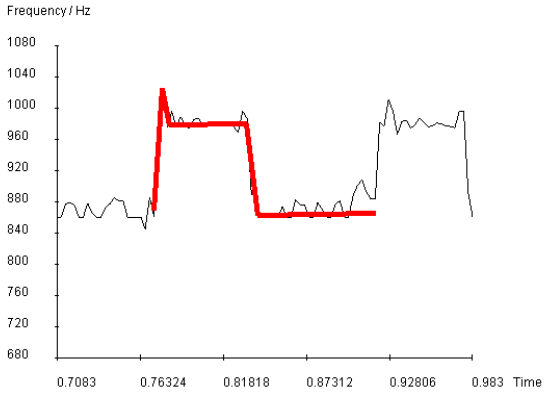

Figure 4. Averaging One Cycle of the Trill for a Frequency Function

The function must oscillate between the frequency of the lower note and the frequency of the higher note. The cycles begin with a slight overshooting of the required frequency, perhaps 20% (see Figure 5).

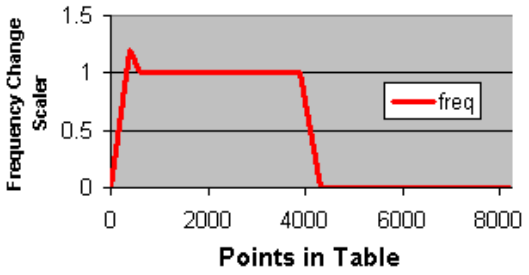

Figure 5. Trill Cycle Frequency Function

The trill should start on the lower note, so we adust our function by a phase shift of 180%. As a refinement, we use random variation on the trill rate.

5 Trill Amplitude Modulator:

Figure 6 shows the waveform of two cycles of the trill. The waveform shows a drop-off in amplitude in the transitions between the trilled notes. It also shows that one of the notes is a little bit louder than the other.

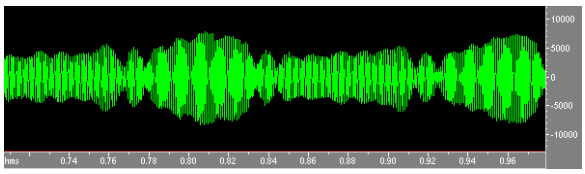

Figure 6. Two Cycles of Waveform of *Dizi* A4 to B4 Trill

What is the optimal shape for the trill amplitude function? We tried a square trill shape, stored it in a function table and then used modulators in that shape to control both the fluctuating frequency and amplitude values of the trills. But with this function the result did not sound natural enough because it didn't model the transitions, which seem to take about 10% of the oscillation cycle.

We then created a function similar to an amplitude envelope, with 40% of the cycle time for each of the frequencies and 10% of the total cycle time for each of the changes between the frequencies (see Figure 7). We stored the new trill shape in a function table and then used an amplitude modulator in that shape to alternately fade the amplitudes of the two signals in and out (see Figure 8).

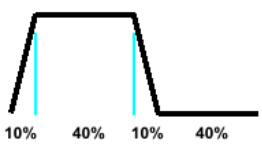

Figure 7. First Trill Function

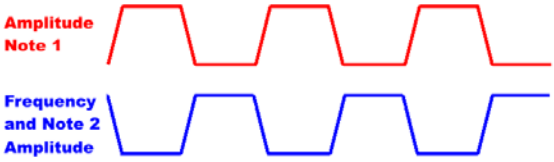

Figure 8. Alternating the Function to Produce a Trill

The PV analysis of the two trill cycles shown in Figure 6 shows in more detail the sharp amplitude drops during the transitions between the two notes (see Figure 14). To get a better trill, we need to model the changes in amplitude that occur during the transition. The most important amplitude changes occur at the same time as the most important frequency changes, but we cannot use the same function to control both the frequency and amplitude because the amplitude peak is in the middle of the cycle, and the frequency peak is at the beginning, so we take the average amplitude of a trill cycle for a separate function (see Figure 14). In addition, the amplitude shows an extra spike at the beginning and ending of the period.

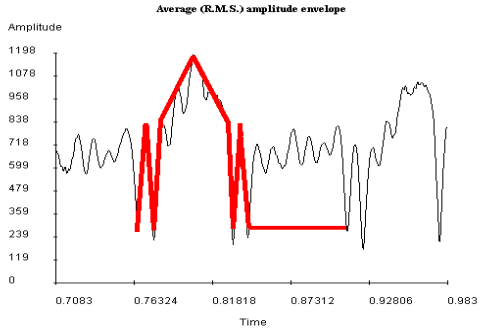

Figure 9. Averaging a Cycle of the Trill for an Amplitude Function

The function must oscillate between the amplitude of the lower note and the amplitude of the higher note. The cycles begin and end with a slight amplitude spike during the transition between the two notes.

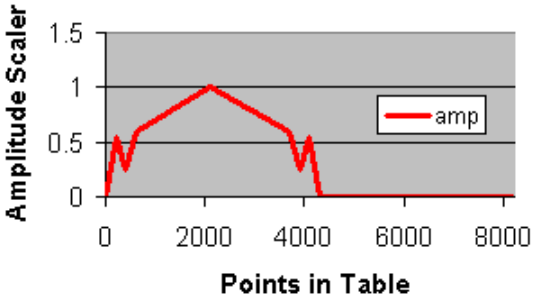

Figure 10. Trill Cycle Amplitude Function

Several score parameters can easily control the trill rate in a line segment. For example, our design uses four score parameters to control an initial, middle and final tremolo rate, and the time required to change from the first tremolo rate to the second. A separate score parameter controls the changing amplitude of the trill as it would for a single sustained tone. The flexibility is especially useful, and much easier than typing many notes and trying to manually adjust their start times, durations and amplitudes to get a naturally-varying rate and amplitude quality.

6 Tremolos:

Tremolos are similar to trills, but have the additional problem of a larger frequency change. To make tremolos sound believable using this technique, the spectrum of each note must be correct, and refining this parameter improves some of the trills as well. If two notes are close to the original note modeled, their spectra may be similar enough to use the same wavetable. But if two notes alternate by leap, it is likely that the spectrum of one of the notes will be very different from that of the original note, causing its timbre to be distorted, and it may sound synthetic. Large tremolos between the two notes can also cause aliasing (where some harmonics fold over because they are too high to be represented by the sampling rate).

7 Conclusion:

After comparing the results, we concluded that trills modeled using the function table method sound better than with the overlapping or line segment methods, and the new method is easier to use. We used one function table for frequency modulation and another for amplitude modulation. Our method is highly successful and the trills sound realistic enough to fool listeners into believing that humans played our synthesized examples.

8 Acknowledgements:

Special thanks go to William **Chan** Chun Nin, who has been invaluable for recording *dizi* samples and consultation on playing the *dizi.* And, thanks to the RGC Grant # HKUST 6020/02H for funding this research.

References:

Beauchamp, J. 1993. "Unix Workstation Software for Analysis, Graphics, Modification, and Synthesis of Musical Sounds," Audio Engineering Society Preprint No. 3479.

Dolson, M. 1986. "The Phase Vocoder: A Tutorial," *Computer Music Journal*, 10(4): 14-27.

Horner, A., Ayers, L. and Law, D. 1999. "Synthesis Modeling of the Chinese Dizi, Bawu, and Sheng," *Journal of the Audio Engineering Society,* Vol. 47, No. 12, pp. 1076-1087.

Horner, A. and Ayers, L. 1998. "Modeling Acoustic Wind Instruments with Contiguous Group Synthesis," *Journal of the Audio Engineering Society,* Vol. 46, No. 10, pp. 868-879.

Horner, A. and Ayers, L. 1998. "Modeling Chinese Musical Instruments," *Proceedings of the 135th Meeting of the Acoustical Society of America,* Vol. 4, pp. 2541-2542.

Horner, A. and Ayers, L. 2001. *Cooking with Csound, Part 1: Woodwind and Brass Recipes*, AR Editions.

Horner, A. and Beauchamp, J. 1996. "Piecewise Linear Approximation of Additive Synthesis Envelopes: A Comparison of Various Methods," *Computer Music Journal,* Vol. 20, No. 2, pp. 72-95.

McAulay, R. and Quatieri, T. 1986. "Speech Analysis/Synthesis Based on a Sinusoidal Representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 34, No. 4, pp. 744-754.

Roads, C. 2001. *The Computer Music Tutorial*, MIT Press.

Rodet, X. and Lefevre, A. 1997. "The Diphone program: New features, new synthesis methods and experience of musical use," *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, pp. 418-419.

Serra, X. and Smith, J. 1990. "Spectral Modeling Synthesis, A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, Vol. 14, No. 4, pp. 418-419.

Wun, S. 2001. Ported Pvan from Unix to PC.