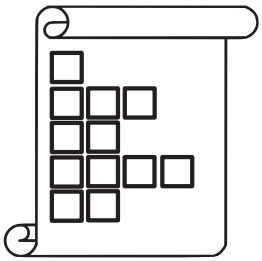
18
Technique 2
Tables



SECTION 18.1

Wavetable Synthesis

Wavetable synthesis employs lookup tables which contain stored functions. We have already seen how to shape a periodic function of time with another function, which we call *waveshaping*; for instance, we saw a trivial case when producing a cosine wave by connecting a phasor~ to a cos~ object. Waveshaping will be explored in more detail in the next section. The built-in cos~ object could be replaced with a lookup table that does the same thing; only, because the function is stored in a table rather than computed, we would then prefer to say it's a *wavetable lookup*. It has probably occurred to you that we can change the function to obtain any other waveforms, and we have already seen how to do this by filling a table with the sum of sinusoids. This should highlight the connection between additive synthesis and wavetables; a wavetable is a periodic function, starting and ending on zero, so it must be expressible as a sum of sines. Study figure 18.1 for a moment and hopefully you will see another connection.

Waveshaping and wavetable synthesis are somewhat connected, at least in a degenerate case. In the former we have a sinusoidal or more complex periodic waveform that is passed through a nonlinear function, often a lookup table, to modify its spectrum. The first column of figure 18.1 shows this when the function is linear. Our wavetable is a line running through zero between -1.0 and 1.0, and the index to the table is a sinusoidal wave. In this case, where we view the process as waveshaping we can use a nonlinear function and change the amplitude or shape of the input waveform but keep the function fixed. Notice how the index is placed centrally around the middle of a 128-point table by subtracting 64, because the input wave is bipolar.

In the case of wavetable synthesis we use a phasor as an index to a periodic function. We keep the indexing waveform fixed but dynamically modify the shaping function. This is shown in the second column of figure 18.1. The table is filled with a cycle of a cosine wave and we index it with a phasor. Notice how the index covers the table domain using the full 128 points because the input wave is unipolar and positive.

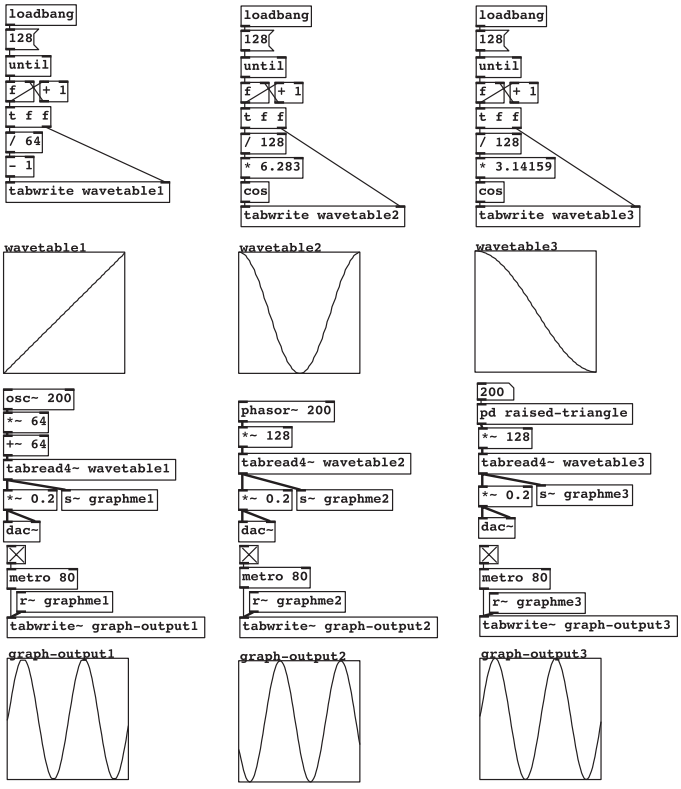


Figure 18.1
A useful identity: waveshaping and wavetable synthesis have a trivial equivalence.

The last column shows how we can use smaller tables for symmetrical functions by only storing half of it. Instead of a phasor we use a raised triangle wave to scan around the middle ($y = 0$) of the stored function. This provides an interesting advantage that the function endpoints do not have to match up. So we can use nonperiodic functions such as polynomials that have interesting turning points where we want.

SECTION 18.2

Practical Wavetables

Figure 18.2 demonstrates some techniques for employing wavetables. Remember that `tabread4~` is an interpolating reader, so your tables should really have an extra three points if you want to avoid small glitches. We see two ways of changing the sound. The first is to issue a command to change the table contents, either by computing a new function and filling the table, or using one of

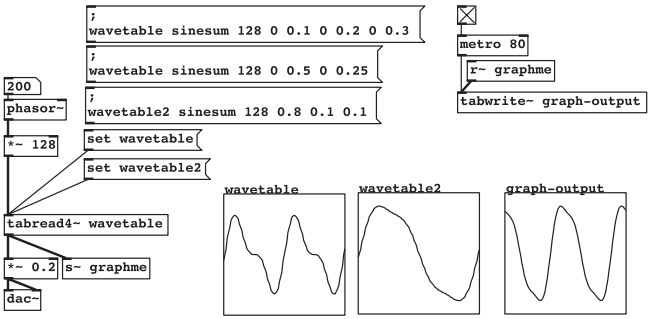


Figure 18.2
Using wavetables in Pure Data.

the built in wave constructors like `sinesum`. Another way is to `set` tabread4~ to reference a different table by giving it a message with the new table name as an argument.

This is okay for obtaining a fixed spectrum, but how do we make dynamic sounds? One way of doing this is wavetable switching by changing the index offset into a lookup table that stores several periodic functions. These could be single cycles of any evolving waveform. In effect this a perversion of timestretching because we can move forwards or backwards through an evolving sound at any rate without causing clicks, but we can't make jumps between arbitrary cycles.

This highlights the difficulty for wavetable synthesis, which is working out a way to change the table contents without causing a click, in other words to keep the phase changing smoothly. One method is to write behind the phasor index, but this causes problems rather like those of file locking if a table is shared between oscillators. A good solution is to crossfade between two or more tables so that you update one that isn't currently being read. This brings us nicely to the subject of wavescanning and vector synthesis techniques.

SECTION 18.3

Vector Synthesis

Vector synthesis is really a control strategy, but it is commonly used with a wavetable synthesis layer, so this is a good place to explore it. A useful object in Pure Data is the table lookup oscillator tabosc4~, which essentially does what we have seen above in a neat package.

Vector synthesis can be considered a form of additive synthesis that blends more complex spectra. It belongs in the family of so-called S+S (sample plus synthesis) methods where oscillators that replay prestored wavetables are combined with a fade matrix, envelope generators, and filters. This strategy is

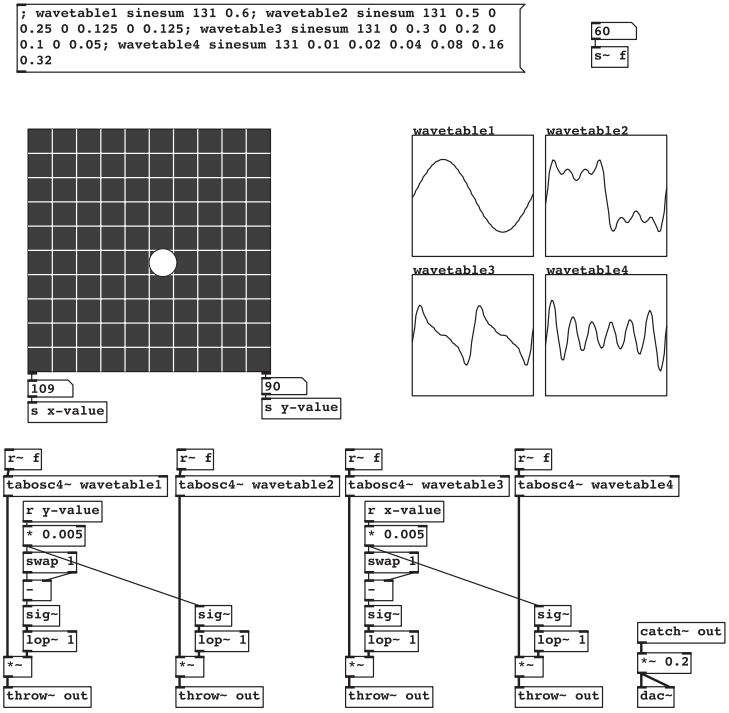


Figure 18.3
A 2D synth using an external GUI object.

typical of many digital music synthesisers from the 1990s such as the Sequential Circuits Prophet VS vector synthesizer.

In figure 18.3 we see four wavetables filled with preconstructed waveforms and mixed according to the position of a marker on a 2D plane. This patch requires the `grid` external by Yves Degoyon which is available in extended versions of Pure Data. I find it a useful tool for sound design, especially in combination with a controller surface like the Korg KaossPad, a joystick, or a multi-axis MIDI controller wheel via the `ctlin` object. Each position on the grid produces a unique blend of waveforms, and if movements can be recorded and then played back it forms a complex envelope generator.

SECTION 18.4

Wavescanning Synthesis

Wavescanning is a hybrid method that falls somewhere between waveshaping, granular synthesis, and wavetable synthesis by taking any sound file as a source and using a triangle or sine to index it. We pay no attention to any period/phase

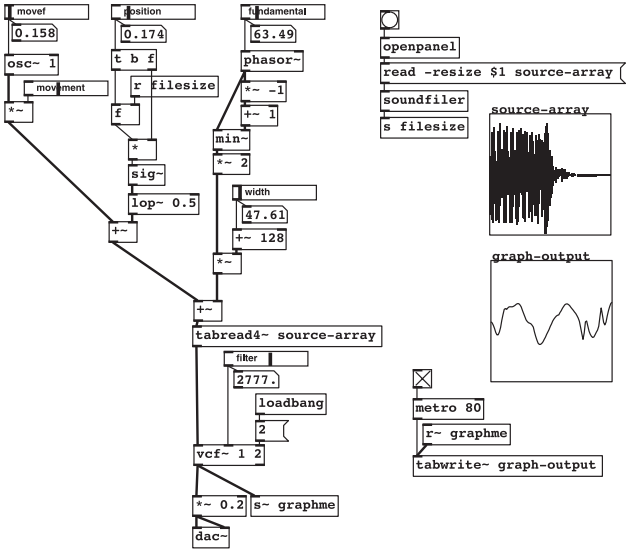


Figure 18.4
A wavescanner with useful musical controls.

boundaries within the sample; we simply use it as raw material. The cool part happens when we scan slowly through the file by adding an offset to the average index position. Figure 18.4 shows a patch that can produce some very interesting dynamic spectra like the PPG or Synclavier devices.

First we produce a triangle wave from a phasor. Multiplying it by a scale factor allows us to change the width of the scan, to squash or stretch the excursion of the index. Adding a slow moving LFO gives the sound a thicker and more unpredictable quality if we want. A filter after the table lookup can help reduce high harmonics that are formed as sidebands when the scan width is very narrow.

References

Bristow-Johnson, R. *Wavetable Synthesis 101: A Fundamental Perspective.* Wave Mechanics, Inc.

Horner, A., Beauchamp, J., and Haken, L. (1993). "Methods for Multiple Wavetable Synthesis of Musical Instrument Tones." *J. Audio Eng. Soc.* 41, no. 5: 336–356.