4 Varieties of Particle Synthesis

There are two kinds of experimentalists. One kind talks to theorists. The theorist makes predictions and the experimentalist then does the experiments. To do this is important, but then all you do is follow the theory. Another type designs his own experiments, and in this way is ahead of the theorists.
—Samuel C. C. Ting (Nobel Prize in Physics 1988)

Unlike particles probed by physicists, synthetic particles inhabit a virtual world. This world is invented, and the waveforms produced by the particles are algorithmically derived. They may be simple and regular in structure, forming smooth pitched tones, or complex and irregular, forming crackling noisy masses.

The engines of particle synthesis are not especially complicated. It is the combination of many simple elements that forms a complex time-varying sound. We shape the sound's evolution by controlling this combination from a high musical level. High-level controls imply the existence of algorithms that can interpret a composer's directives, translating them into low-level particle specifications. (See chapter 8's discussion of simplicity versus complexity in microsound synthesis.)

This chapter presents a catalog of particle synthesis techniques. These include glissons, grainlets, trainlets, pulsars, graphic and sonographic particles, formant particles, transient drawing, particle cloning, and physical and abstract models of particles.

Glisson Synthesis

Glisson synthesis is an experimental technique of particle synthesis. It derives from the technique of granular synthesis, presented in the previous chapter. I implemented glisson synthesis after revisiting Iannis Xenakis's original paper on the theory of granular synthesis (Xenakis 1960). In this article, Xenakis described each grain as a vector within a three-dimensional space bounded by time, frequency, and amplitude. Since the grain is a vector, not a point, it can vary in frequency, creating a short glissando. Such a signal is called a *chirp* or *chirplet* in digital signal processing (Mann and Haykin 1991). Jones and Parks implemented frequency-modulated grains with a variable chirp rate in 1988. My implementation of glisson synthesis dates to 1998.

In glisson synthesis, each particle or glisson has an independent frequency trajectory—an ascending or descending glissando. As in classic granular synthesis, glisson synthesis scatters particles within cloud regions inscribed on the

time-frequency plane. These clouds may be synchronous (metric) or asynchronous (ametric). Certain parameters of glisson synthesis are the same as for granular synthesis: start time and duration of the cloud, particle duration, density of particles per second, frequency band of the cloud, amplitude envelope of the cloud, waveform(s) within the particles, and spatial dispersion of the cloud. (See the description in the previous chapter.)

Magnetization Patterns of Glisson Clouds

The *magnetization pattern*—a combination of several parameters—determines the frequency direction of the glissons within a cloud. First, the glissandi may be deep (wide frequency range) or shallow (small frequency range) (figure 4.1a, b). Second, they may be unidirectional (uniformly up or down) or bidirectional (randomly up or down) (figure 4.1c, d, e). Third, they may be diverging (starting from a common center frequency and diverging to other frequencies), or converging (starting from divergent frequencies that converge to a common center frequency). The center frequency can be changing over time.

Implementations of Glisson Synthesis

Stephen Pope and I developed the first implementation of *glisson synthesis* in February 1998. The software was coded in the SuperCollider 1 synthesis language (McCartney 1996, Pope 1997). Later, I modified the glisson program and carried out systematic tests. In the summer of 1999, Alberto de Campo and I reimplemented glisson synthesis in the SuperCollider 2 language (McCartney 1998).

Experiments with Glisson Synthesis

Short glissons (< 10 ms) with a large frequency variation (> 100 Hz) resemble the classic chirp signals of digital signal processing; sweeping over a wide frequency range in a short period of time. An individual glisson of this type in the starting frequency range of 400 Hz sounds like a tap on a wood block. When the starting frequency range is around 1500 Hz, the glissons sound more like the tapping of claves. As the density of glissons increases and the deviation randomizes in direction, the texture tends quickly toward colored noise.

Medium-length (25–100 ms) glissons "tweet" (figure 4.2a), so that a series of them sounds like birdsong.

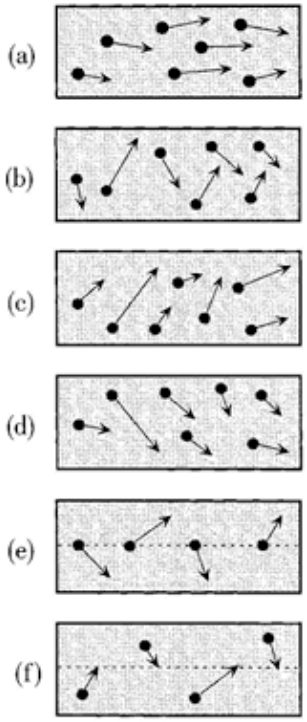


Figure 4.1 Magnetization patterns in glisson synthesis. The vertical axis is frequency and the horizontal axis is time. (a) Shallow (small frequency deviation) bidirectional. (b) Deep (large frequency deviation) bidirectional. (c) Upwards unidirectional. (d) Downwards unidirectional. (e) Diverging from center frequency. (f) Converging to center frequency.

Long glissons (> 200 ms) result in dramatic cascades of sound (figure 4.2b). At certain densities, they are reminiscent of the massed glissandi textures heard in such orchestral compositions as Xenakis's *Metastasis* (1954). A striking effect occurs when the glissandi diverge from or converge upon a common central frequency. By constraining the glissandi to octaves, for example, it is possible to generate sounds similar to the Shepard tones (Risset 1989a, 1997), which seem to spiral endlessly upward or downward.

Assessment of Glisson Synthesis

Glisson synthesis is a variant of granular synthesis. Its effects segregate into two categories. At low particle densities, we can perceive each glissando as a sepa-

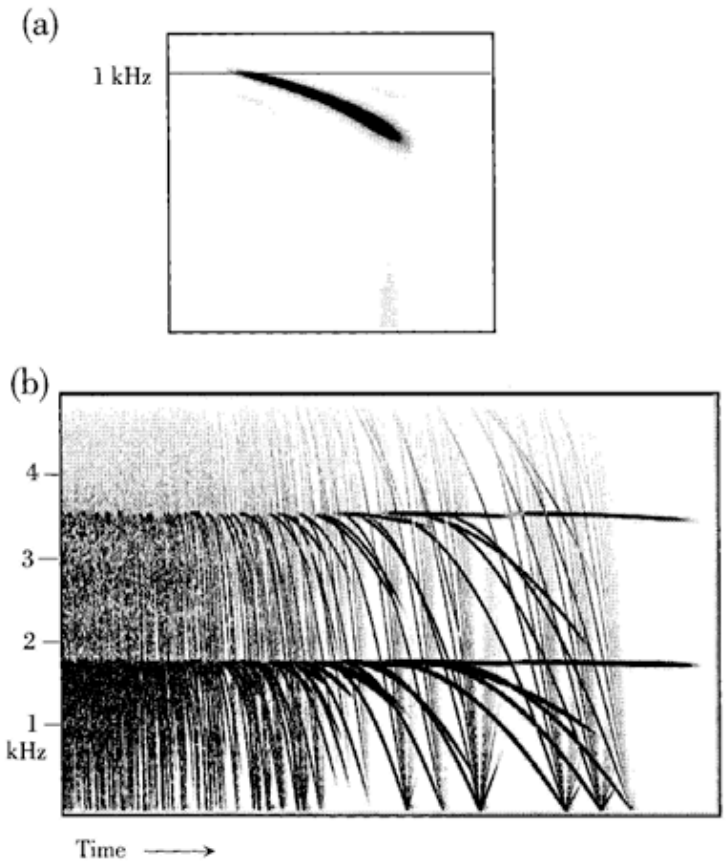


Figure 4.2 Glissons. (a) Sonogram of a single 25-ms glisson. Notice the gray artefacts of the analysis, reflecting the time-frequency uncertainty at the beginning and end of the particle. (b) Glisson cloud generated by a real-time performance. The glisson durations increase over the 16-second duration of the cloud.

rate event in a micro-melismatic chain. When the glissons are short in duration (< 50 ms), their internal frequency variation makes it difficult to determine their pitch. Under certain conditions—such as higher particle densities with greater particle overlap—glisson synthesis produces second-order effects that we perceive on the time scale of sound objects. In this case, the results tend toward a mass of colored noise, where the bandwidth of the noise is proportional to the frequency variation of the glissandi. Several factors can contribute to the sensation of a noise mass, the most important being density, wide frequency variations, and short glisson durations.

Grainlet Synthesis

Grainlet synthesis combines the idea of granular synthesis with that of wavelet synthesis. (See chapter 6.) In granular synthesis, the duration of a grain is unrelated to the frequency of its component waveform. In contrast, the wavelet representation scales the duration of each particle according to its frequency. Short wavelets represent high frequencies, and long wavelets represent low frequencies. Grainlet synthesis generalizes this linkage between synthesis parameters. The fundamental notion of grainlet synthesis is that any parameter of synthesis can be made dependent on (or linked to) any other parameter. One is not, for example, limited to an interdependence between frequency and duration.

I implemented grainlet synthesis in 1996 as an experiment in parameter linkage within the context of granular cloud synthesis (described in the previous chapter). Grainlet synthesis imposes no constraints on the choice of waveform, particle envelope, or any other parameter, except those that we introduce through parameter linkage.

Parameter Linkage in Grainlet Synthesis

Parameter linkage is the connecting of one parameter with a dependent parameter. As parameter *A* increases, for example, so does its dependent parameter *B*. One can also stipulate inverse linkages, so that an increase in *A* results in a decrease in *B*.

Parameter linkages can be drawn as patch diagrams connecting one parameter to another (figure 4.3). An arrow indicates a direct influence, and a gray

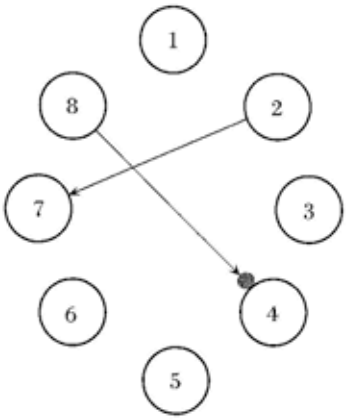


Figure 4.3 Parameter linkage in grainlet synthesis. Each circle represents a parameter of grainlet synthesis. An arrow from one parameter to another indicates a dependency. Here parameter 7 is dependent on parameter 2. If parameter 7 is spatial depth and parameter 2 is grainlet start time, then later grainlets have more reverberation. Notice that parameter 4 is inversely dependent on parameter 8, as indicated by the gray dot. If parameter 4 was grainlet duration and parameter 8 was grainlet frequency, then higher frequency grainlets are shorter in duration (as in wavelet resynthesis).

circle an inverse linkage. I wrote a program in the C language to realize these parameter linkages. In the first version, grainlet duration could be specified in terms of the number of cycles of the fundamental period. If the number of cycles is ten, for example, a grainlet at 100 Hz lasts 10×0.01 sec $= 0.1$ sec, while a grainlet at 1000 Hz lasts 10×0.001 Hz $= 0.01$ sec.

After initial tests, I generalized the parameter linkage from frequency and duration to dependencies between any two synthesis variables. The grainlet synthesis program generates a data file. A Csound program for granular synthesis interprets this file and synthesizes the sound. The synthesis parameters include the following:

- Cloud density (number of grainlets per second)
- Grainlet amplitude
- Grainlet start-time
- Grainlet frequency
- Grainlet duration

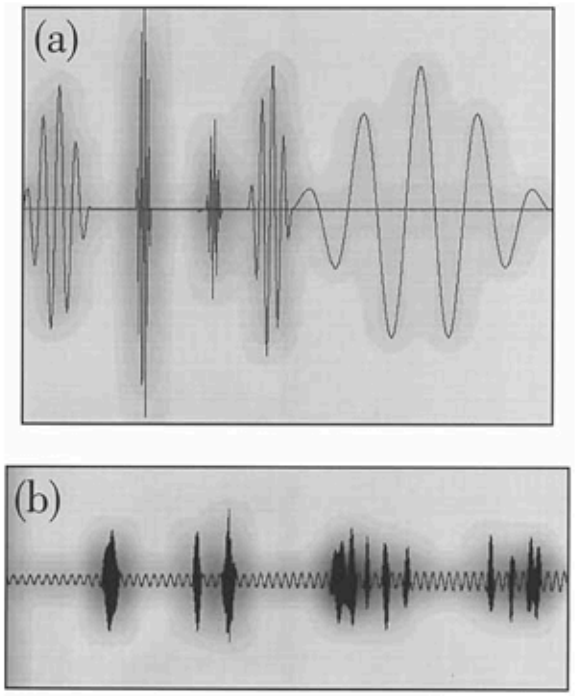


Figure 4.4 Collections of grainlets. (a) These grainlets are scaled in duration according to their frequency. (b) Superposition of short high-frequency grainlets over a long low-frequency grainlet.

- Grainlet waveform
- Grainlet position in the stereo field
- Grainlet spatial depth (amount of reverberation)

Frequency-Duration Experiments

The first experiments with grainlet synthesis simulated the relationship between grain duration and grain frequency found in wavelet representation (figure 4.4a and b). I later generalized this to allow any frequency to serve as a *point of attraction* around which certain durations (either very long or very short) could gravitate (figure 4.5).

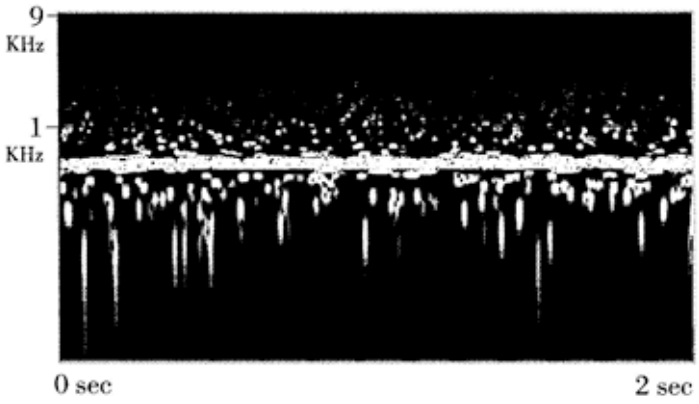


Figure 4.5 Inverse sonogram plotted on a logarithmic frequency scale, showing a frequency point of attraction around the grainlet spectrum. The grainlets whose frequencies are close to the point of attraction (700 Hz) are long in duration, creating a continuous band centered at this point.

Amplitude-Duration Experiments

These experiments linked grain duration with the amplitude of the grains. In the case of a direct link, long grains resulted in louder grains. In an inverse relationship, shorter grains had higher amplitudes.

Space-Duration Experiments

These experiments positioned grains in space according to their duration. Grains of a stipulated duration always appeared to emanate from a specific location, which might be any point in the stereo field. Grains whose duration was not stipulated scattered randomly in space.

Frequency-Space Experiments

These experiments positioned grains in space according to their frequency. Grains of a stipulated frequency appeared to always emanate from a specific location, which might be any point in the stereo field. Other grains whose frequency was not stipulated scattered randomly in space.

Amplitude-Space Experiments

These experiments assigned grains a spatial location according to their amplitude. Grains of a stipulated amplitude appeared to always emanate from a specific location, which might be any point in the stereo field. Other grains whose amplitude was not stipulated scattered randomly in space.

Assessment of Grainlet Synthesis

Grainlet synthesis is an experimental technique for realizing linkages among the parameters of microsonic synthesis. It appears to be a good technique for forcing high-level organizations to emerge from microstructure. Specifically, the clouds generated by grainlet synthesis stratify, due to the internal constraints imposed by the parameter linkages. This stratification is seen in textures such as a dense cloud of brief, high-frequency grains punctuated by low and long grains. Other clouds stratify by spatial divisions. Many parameter linkages are easy to discern, conveniently serving as articulators in music composition.

Trainlet Synthesis

A *trainlet* is an acoustic particle consisting of a brief series or train of impulses. Like other particles, trainlets usually last between 1 to 100 ms. To create time-varying tones and textures, an algorithm is needed that can spawn thousands of trainlets from a few high-level specifications. The main parameters of trainlet synthesis are the density of the trainlets, their attack time, pulse period, harmonic structure, and spectral energy profile. Before explaining the theory of trainlets, let us summarize the basics of impulse generation.

Impulse Generation

An *impulse* is an almost instantaneous burst of energy followed by an immediate decline in energy. In its ideal form, an impulse is infinitely narrow in the time dimension, creating a single vertical line in its time-domain profile. In practice, however, impulses always last a finite time; this is their *pulse width*. Electronic impulses in the real world vary greatly, exhibiting all manner of attack shapes, decay shapes, and transition times. These variations only make them more interesting from a musical point of view.

Table 4.1 Technical specifications of the Hewlett-Packard HP8005B pulse generator

Repetition rate	0.3 Hz to 20 mHz in five ranges; Vernier control within each range
Attack and decay transition times	<10 ns to 2 s in five ranges; Verniers for leading and trailing edges
Overshoot, preshoot, and ringing	<5% of pulse amplitude each
Pulse width	<25 ns to 3 s in five ranges; Vernier controls within each range
Width jitter	<0.1% + 50 ps of any width setting
Pulse delay	<100 ns to 3 s, in five ranges
Delay jitter	<0.1% + 50 ps of any delay setting
Period jitter	<0.1% of any period setting

Impulse generation is one of the most venerable methods of electronic sound synthesis. At infrasonic frequencies, we perceive impulses separately and notice their rhythmic pattern. At audio frequencies the impulses fuse into continuous tones. Electronic music studios of the 1950s were equipped with laboratory impulse generators (IGs). The IGs installed in the WDR Cologne studio had a pulse frequency range that extended from the infrasonic frequencies (1.1 Hz, corresponding to a pulse duration of 900 ms) to the high audio frequencies (10 kHz, or pulse duration of 0.1 ms).

Laboratory impulse generators have evolved considerably since the 1950s. Modern IGs are solid-state analog devices. Table 4.1 lists the specifications of a low-voltage IG in the price range of $600. Notice the wide frequency range, from 0.15 Hz (one impulse every 6.6 seconds) to 20 MHz. The spectrum of an impulse with a period of 50 nanoseconds (corresponding to a frequency of 20 MHz) is very broad. More expensive IGs, costing up to $25,000, extend the fundamental frequency range beyond 3 gHz. The main parameters of an IG are the shape of the impulse, the fundamental period of repetition, and the duty cycle of the pulse wave relative to the period. These parameters affect the rhythm or pitch of the output signal, and its overall spectrum.

Industrial impulse generators are not designed for flexible performance in real time. Notice the specifications in table 4.1, where repetition rate, transition time, pulse width, and pulse delay are broken into five or six ranges. This means that they cannot sweep continuously from one setting to another. They are also not designed for automated operation.

Moreover, the raw output of an IG is a series of clicks, not sufficient in themselves for music synthesis. Since a click has a broad bandwidth, a common

synthesis technique is to filter the impulses with a bandpass characteristic to carve out a specific peak in the spectrum. When the filter is very narrow, a single impulse causes the filter to resonate at a specific frequency, yielding the sensation of pitch. Another processing trick is to pass the impulse train through a variable delay and feedback loop, so that the density of impulses accumulates under the control of the composer. Karlheinz Stockhausen and Gottfried Michael Koenig used these kinds of techniques in the synthesis of *Kontakte* in 1960. (See also Kaegi 1967.)

Ultimately, all electronic signals, whether analog or digital, are bandlimited. An analog IG has an obvious advantage over a digital IG, which is strictly bandlimited by the Nyquist theorem. As is clear from the specifications in table 4.1, the bandwidth of analog IGs is much broader.

Theory and Practice of Trainlets

A *trainlet cloud* contains one or more trainlets. The main variables in the synthesis of a trainlet cloud are:

- Density per second of trainlets
- Trainlet pulse period
- Trainlet harmonic structure
- Trainlet spectral energy profile

An *amplitude envelope* shapes each trainlet, and the generator projects each trainlet to a specific location in space. The parameters vary for each trainlet, so each trainlet may be unique.

Since trainlet synthesis operates in the digital domain, it is important that the impulses be bandlimited to avoid aliasing. A standard bandlimited impulse generator for modern synthesis languages is Godfrey Winham's buzz unit generator and its variant, gbuzz (Winham 1966; Winham and Steiglitz 1970; Howe 1975; Steiglitz 1996). Buzz emits a set of N harmonic partials of equal amplitude. As the number of harmonics N increases, the pulse width becomes narrower (figure 4.6). Gbuzz emits a set of N harmonic partials with more flexible control of the spectrum.

In the 1998 prototype implementation of trainlet synthesis, I wrote the synthesis engine in the Csound language (Boulanger 2000). Csound's gbuzz unit generator emits harmonic cosines aligned to a fundamental frequency. It scales the amplitudes of the cosines so their sum equals a stipulated peak value. One

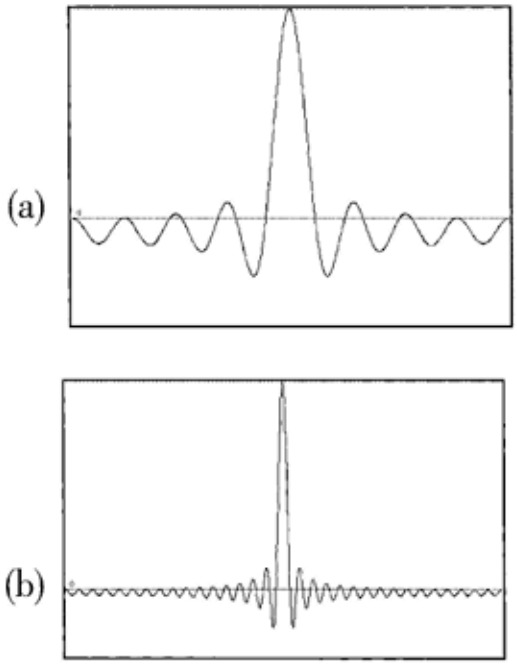


Figure 4.6 Bandlimited pulses. (a) Sum of eight harmonics. (b) Sum of thirty-two harmonics. Notice the narrowing of the pulse.

can stipulate the lowest harmonic, the total number of harmonics starting from the lowest, and the chroma of the harmonic series (see below).

Chroma is a spectral brightness factor. Figure 4.7 shows the relationship between chroma and spectra. Chroma determines the relative strength of the harmonic series. If the lowest harmonic partial has a strength coefficient of A, the lowest harmonic + nth partial will have a coefficient of $A \times (\text{chroma}^n)$, an exponential curve. The chroma may be positive, zero, or negative, and is not restricted to integers. If chroma = 1, the harmonics are of equal strength. If chroma < 1, the higher harmonics are attenuated, as though the signal had been sent through a lowpass filter. As the value of chroma tends toward 0, they attenuate more rapidly. If chroma > 1, the highest harmonic has the greatest amplitude (as though a highpass filter had processed it), while each lower harmonic stepping down from the highest has a progressively lower amplitude. As the chroma value increases, the signal is brighter in timbre.

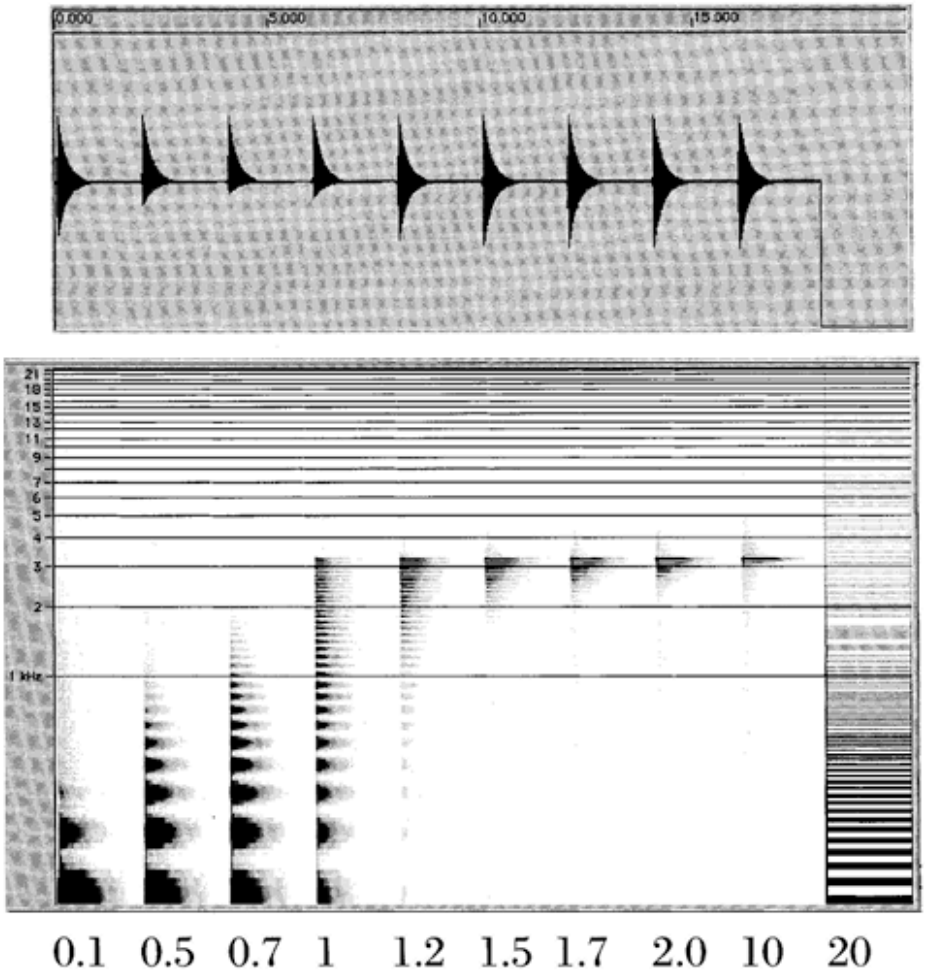


Figure 4.7 Relationship between chroma and spectra. We see time-domain waveforms and sonogram spectra (on a logarithmic frequency scale) of trainlets with increasing chroma. All trainlets have a pulse frequency of 100 Hz, with thirty-two harmonics, and the lowest harmonic is always 1. The chroma values are indicated at the bottom of the figure. In the last example, chroma = 20, the algorithm explodes.

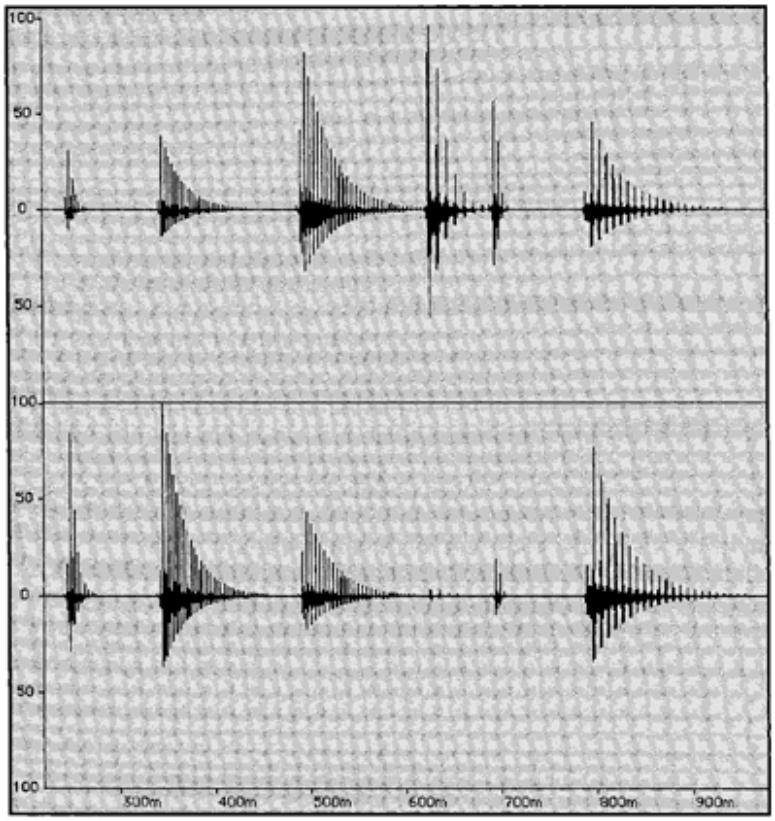


Figure 4.8 Time-domain view of scattering of trainlets in two channels. The trainlets are at various frequencies and have various durations. The amplitude of each trainlet (the sum of both channels) is constant.

Trainlet clouds are musical units on the sound object time scale. As in granular synthesis, synchronous clouds spawn a regular series of trainlets. A linear accelerando or rallentando can also be realized in this mode. Asynchronous clouds spawn trainlets at random intervals according to the stipulated density. Figure 4.8 provides a time-domain view of a trainlet cloud, while figure 4.9 shows the sonogram of two trainlet clouds, one with long trainlets, the other with short trainlets. Notice their characteric spectral pattern.

Table 4.2 enumerates the rest of the cloud parameters. The values in brackets are typical ranges.

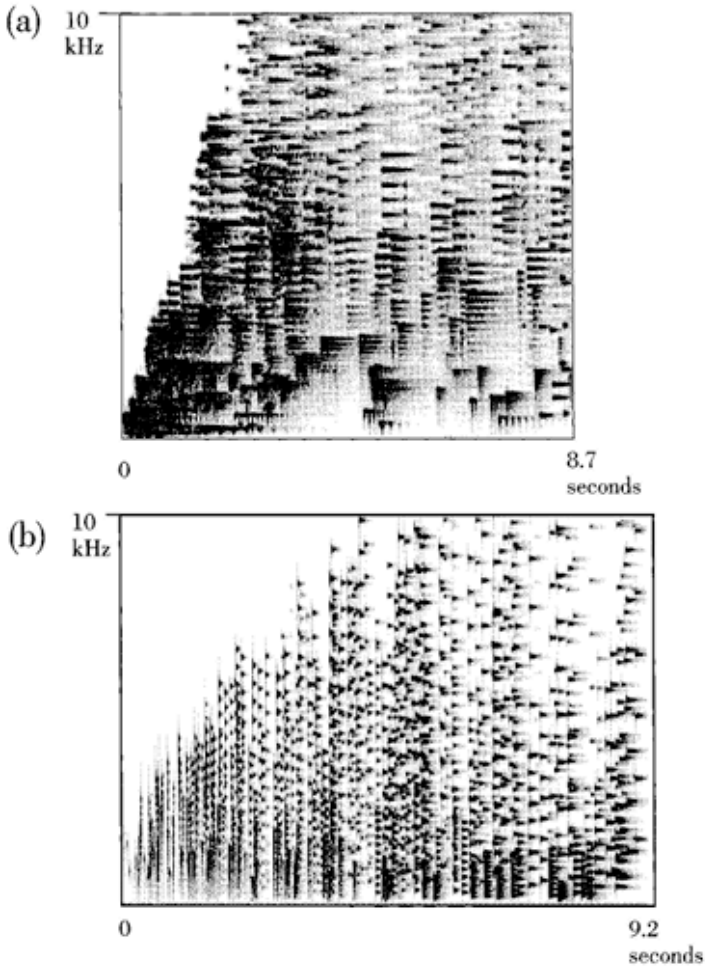


Figure 4.9 Sonogram of trainlet clouds. (a) Long trainlets. (b) Short trainlets.

Table 4.2 Trainlet cloud parameters

1. Cloud start time, in seconds
2. Cloud duration, in seconds
3. Random duration flag. If set, interpret the initial and final trainlet durations as maximum and minimum trainlet durations, respectively. The actual durations are generated randomly between these limits.
4. Trainlet durations at start of cloud (0.001–0.8 sec)
5. Trainlet durations at end of cloud
6. Density of trainlets per second at start of cloud (1–300)
7. Density of trainlets per second at end of cloud
8. Upper frequency bandlimit of the cloud at start (20 Hz–20 KHz)
9. Lower frequency bandlimit of the cloud at start
10. Upper frequency bandlimit of the cloud at end
11. Lower frequency bandlimit of the cloud at end
12. Amplitude at start of cloud (1–96 dB)
13. Amplitude at end of cloud
14. Number of harmonics per trainlet at start of cloud (1–64)
15. Number of harmonics per trainlet at end of cloud
16. Lowest sounding harmonic at start of cloud
17. Lowest sounding harmonic at end of cloud
18. Chroma at start of cloud. If chroma < 0 then the effect is lowpass. If chroma $= 1$ all harmonics are equal in strength. If chroma > 1 then the effect is highpass.
19. Chroma at end of cloud,
20. Initial waveform, usually sine
21. Final waveform, usually sine
22. Spatial position of trainlets in the stereo field at the start of the cloud, either fixed ($0 =$ L, $0.5 =$ middle, $1 =$ right) or random
23. Spatial position of trainlets in the stereo field at the end of the cloud
24. Initial attack time (5–50 ms)
25. Final attack time

Note: Initial and final values refer to the beginning of the cloud and the end of the cloud, respectively.

We have also implemented a version of trainlet synthesis in the real-time Creatovox synthesizer, described in chapter 5. One hand can play the trainlet clouds on the keyboard of the instrument, while the other manipulates the cloud parameters with joysticks and other MIDI controllers.

Assessment of Trainlet Cloud Synthesis

I conducted numerous tests with the trainlet cloud technique focussing primarily on trainlets with exponential attacks, linear sustains, and linearly decaying envelopes. The gbuzz unit generator, stamps the trainlet sonority with its "brittle" signature (Schindler 1998). This brittleness derives from gbuzz's fixed waveform and unnatural spectrum. Long trainlets in sparse clouds revisit a sound quality that was characteristic of early computer music studies.

My best results involved trainlets with durations of 40 ms to 100 ms, where the characteristic timbre of the trainlets had just enough time to manifest. Fundamental frequencies greater than 500 Hz tend to sound metallic, fundamentals under 500 Hz are darker and more effective. In any case, with trainlets having up to thirty-two harmonics, the fundamental must remain under 600 Hz in order to avoid aliasing. A 32-harmonic trainlet at 600 Hz with a chroma of 1.0 is very bright.

With over twenty parameters to vary, a wide range of sound material can be produced by trainlet cloud synthesis, which seems to have considerable musical potential. More research is needed, however. As with any technique, the output of trainlet synthesis can be taken as a starting point for further signal processing. I have, for example, tested configurations in which a trainlet generator connects to a constant-Q bandpass filter. The filter lets one accentuate a particular formant in the partial spectrum.

Pulsar Synthesis

Pulsar synthesis (PS), named after the spinning neutron stars that emit periodic signals in the range of 0.25 Hz to 642 Hz, is a powerful method of digital sound synthesis with links to past analog techniques. Coincidentally, this range of frequencies—between rhythm and tone—is of central importance in pulsar synthesis.

PS melds established principles within a new paradigm. In its basic form, it generates electronic pulses and pitched tones similar to those produced by

analog instruments such as the Ondioline (Jenny 1958; Fourier 1994) and the Hohner Elektronium (1950), which were designed around the principle of filtered pulse trains. Pioneering electronic music composers including Stockhausen (1955, 1957, 1961, 1963) and Koenig (1959, 1962) used filtered impulse generation as a staple in their studio craft. Pulsar synthesis is a digital technique, however, and so it accrues the advantages of precise programmable control, waveform flexibility, graphical interface, and extensibility. In its advanced form, pulsar synthesis generates a world of rhythmically structured crossbred sampled sounds.

This section first presents the basic theory of pulsars and pulsar graphs. We then move on to the more advanced technique of using pulsars to transform sampled sounds through cross-synthesis, presenting musical applications of pulsar synthesis in compositions by the author. At the end of this section, we describe the features of a new interactive program called PulsarGenerator (Roads 2001).

Basic Pulsar Synthesis

Basic pulsar synthesis generates a family of classic electronic music timbres akin to those produced by an impulse generator connected to a bandpass filter. Unlike this classic technique, however, there is no filter in the basic PS circuit.

A single pulsar is a particle of sound. It consists of an arbitrary *pulsaret* waveform w with a period d followed by a silent time interval s (figure 4.10a). The total duration of a pulsar is $p = d + s$, where p is the *pulsar period*, d is the *duty cycle*, and s is silent. Repetitions of the pulsar signal form a *pulsar train*. Let us define the frequency corresponding to the repetition period as $f_p = 1/p$ and the frequency corresponding to the duty cycle as $f_d = 1/d$. Typical ranges of f_p are between 1 Hz and 5 kHz, the typical range of f_d is from 80 Hz to 10 kHz.

In PS, both f_p and f_d are continuously variable quantities. They are controlled by separate envelope curves that span a train of pulsars. The train is the unit of musical organization on the time scale of notes and phrases, and can last anywhere from a few hundred milliseconds to a minute or more.

Notice in figure 4.10b that the *duty ratio* or d:s *ratio* varies while p remains constant. In effect, one can simultaneously manipulate both fundamental frequency (the rate of pulsar emission) and what we could call a *formant frequency* (corresponding to the duty cycle), each according to separate envelopes. A

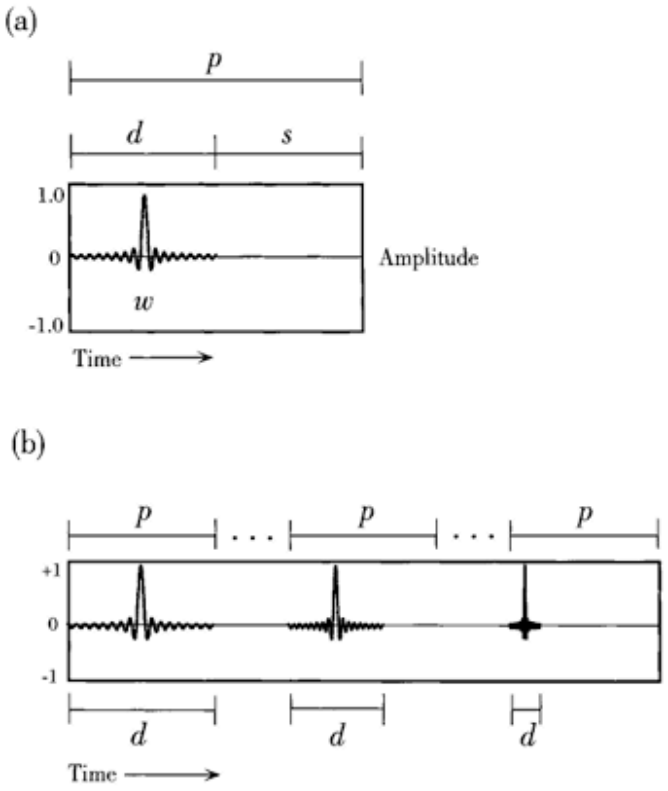


Figure 4.10 Anatomy of a pulsar. (a) A pulsar consists of a brief burst of energy called a *pulsaret* w of a duration d followed by a silent interval s. The waveform of the pulsaret, here shown as a band-limited pulse, is arbitrary. It could also be a sine wave or a period of a sampled sound. The total duration $p = d + s$, where p is the fundamental period of the pulsar. (b) Evolution of a pulsar train, time-domain view. Over time, the pulsar period p remains constant while the pulsaret period d shrinks. The ellipses indicate a gradual transition period containing many pulsars between the three shown.

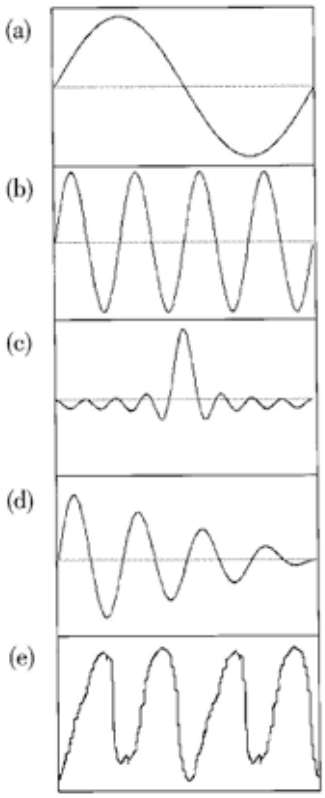


Figure 4.11 Typical pulsaret waveforms. In practice, any waveform can be used. (a) Sine. (b) Multicycle sine. (c) Band-limited pulse. (d) Decaying multicycle sinusoid. (e) Cosmic pulsar waveform emitted by the neutron star Vela X-1.

formant is a peak region in a spectrum. Lowering the fundamental means increasing s, and raising the fundamental means decreasing s.

So far, the structure we have described is similar to that of a standard impulse generator. Pulsar synthesis generalizes this configuration in several ways. First, it allows the pulsaret w to be any waveform. Figure 4.11 shows some typical pulsaret waveforms, including those with multiple subperiods within their duty cycle (figure 4.11b, d, and e).

Let us assume that w is a single cycle of a sine wave. From a signal processing point of view, this can be seen as a sine wave that has been limited in time by a rectangular function v, which we call the *pulsaret envelope*. An important

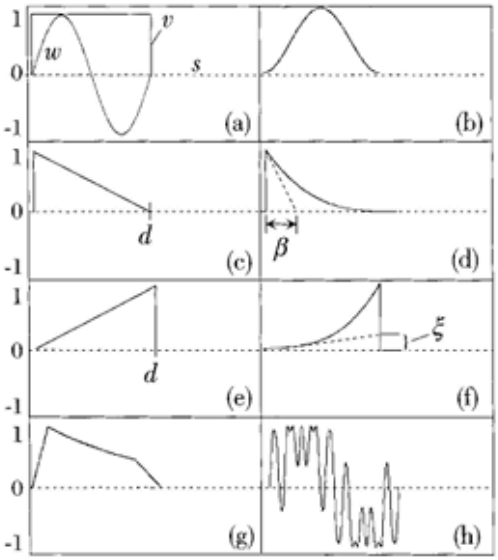


Figure 4.12 Typical pulsaret envelopes. (a) Rectangular. (b) Gaussian. (c) Linear decay. (d) Exponential decay. The term β determines the steepness of the exponential curve. (e) Linear attack, with duty cycle d. (f) Exponential attack. The term ξ determines the steepness of the exponential curve. (g) FOF envelope. (h) Bipolar modulator.

generalization is that v can also be any shape. As we show later, the envelope v strongly affects the spectrum of the pulsar train.

Figure 4.12 shows some typical pulsaret envelopes. A rectangular envelope (figure 4.12a) produces a broad spectrum with strong peaks and nulls for any pulsaret. Figure 4.12g depicts a well-known configuration for formant synthesis, an envelope with a sharp attack followed by an exponential decay. This corresponds to the FOF and Vosim techniques described later in this chapter. Such a configuration can be seen as a special case of pulsar synthesis. As figure 4.12h shows, the envelope can also be a bipolar ring modulator.

Keeping p and w constant and varying d on a continuous basis creates the effect of a resonant filter swept across a tone. There is, of course, no filter in this circuit. Rather, the frequency corresponding to the duty cycle d appears in the spectrum as a formant peak. By sweeping the frequency of this peak over time, we obtain the sonic equivalent of a time-varying bandpass filter applied to a basic impulse train.

Pulsaret-Width Modulation

Pulse-width modulation (PWM) is a well-known analog synthesis effect occuring when the duty cycle of a rectangular pulse varies while the fundamental frequency remains constant (figure 4.13a). This produces an edgy "sawing" quality as the upper odd harmonics increase and decrease over the course of the modulation. At the extremes of PWM, the signal is silent. For example, when $d = 0$, PWM results in a signal of zero amplitude (figure 4.13b). When $d = p$, PWM produces a signal of a constant amplitude of 1 (figure 4.13c).

Pulsaret-width modulation (PulWM) extends and improves this model. First, the pulsaret waveform can be any arbitrary waveform. Second, it allows the duty cycle frequency to pass through and below the fundamental frequency. Here $f_d \leq f_p$. Notice in figure 4.13 how the duty cycle of the sinusoid increases from (d) to (e). In (f), $p = d$. Finally, in (g) $p < d$. That is, the duty cycle is longer than the fundamental period. Only the first quadrant of the sine wave repeats. The fundamental period cuts off the duty cycle of the pulsaret in midwaveform. In our implementation, we apply a user-controlled crossfade time around this cutoff point, which we call the *edge* factor. When there is no crossfade, the edge factor is high.

We have also tested an alternative approach to pulsar-width modulation, designed by Alberto de Campo, which produces a different sound. In this method, *overlapped pulsaret-width modulation* or OPulWM, the fundamental frequency corresponds to the rate of pulsar emission, independent of the pulsaret duty cycle. That is, the duty cycle of an individual pulsar always completes, even when it crosses below the fundamental frequency. Whenever the fundamental period expires, our algorithm spawns a new pulsar. Thus, when $d > p$ several pulsars overlap with others whose duty cycle has not yet completed. As d increases, the generator spawns more and more overlapping pulsars. For practical reasons, then, we stipulate an arbitrary overlap limit. OPulWM results in a great deal of phase cancellation and so tends to be a more subtle effect than regular PulWM.

Synthesis across Time Scales

PS operates within and between musical time scales. It generates a stream of microsonic particles at a variable rate across the continuum spanning infrasonic pulsations and audio frequencies. When the distance between successive

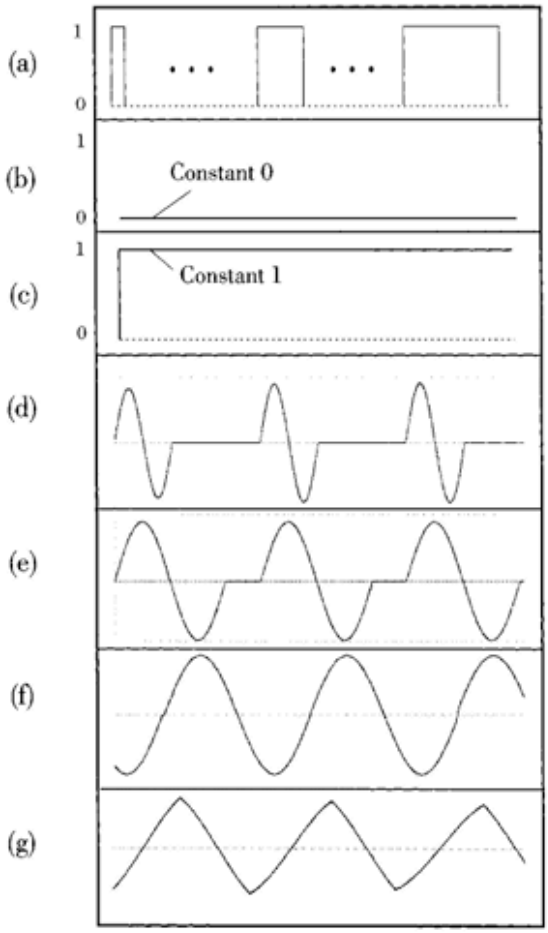


Figure 4.13 Pulsaret-width modulation. (a) Classical PWM with a rectangular pulse shape. The ellipses indicate a gradual transition between the pulses. (b) PWM when the duty cycle $d = 0$ results in a signal of zero amplitude. (c) PWM when the duty cycle $d = p$ (the fundamental period), the result is a signal with a constant amplitude of 1. (d) Pulsar train with a sinusoidal pulsaret. (e) Same period as (d), but the duty cycle is increasing. (f) The duty cycle and the period are equal, resulting in a sinusoid. (g) The duty cycle is greater than the fundamental period, which cuts off the final part of the sine waveform.

impulses is less than about one twentieth of a second, the human hearing mechanism causes them to fuse into a continuous tone. This is the *forward masking effect* (Buser and Imbert 1992). As Helmholtz (1885) observed, in the range between 20 and 35 Hz, it is difficult to distinguish the precise pitch of a sustained tone; reliable pitch perception takes hold at about 40 Hz, depending on the waveform. So listeners hear pitch in a periodic sustained tone for p between approximately 25 ms (corresponding to $f_p = 40$ Hz) and 200 μsec (corresponding to $f_p = 5$ kHz).

As the rate of pulsar emission slows down and crosses through the threshold of the infrasonic frequencies ($f_p < 20$ Hz), the sensation of continuous tone evaporates, and we can perceive each pulsar separately. When the fundamental f_p falls between 62.5 ms (corresponding to the time span of a thirty-second note at quarter note = 60 MM) and 8 sec (corresponding to the time span of two tied whole notes at quarter note = 60 MM), we hear rhythm. The fundamental frequency envelope becomes a graph of rhythm. This takes the form of a function of time that a user draws on the screen of a computer. Such a *pulsar graph* can serve as an alternative form of notation for one dimension of rhythmic structure, namely the onset time of events. The correspondence between the musical units of rhythmic structure (note values, tuplets, rests, etc.) can be made clear by plotting note values on the vertical or frequency scale. For example, assuming a tempo of 60 MM, a frequency of 5 Hz corresponds to a quintuplet figure. Note that the two-dimensional pulsar graph does not indicate the duration of the events. This could be represented by adding a third dimension to the plot.

To interpret the rhythm generated by a function inscribed on a pulse graph, one has to calculate the duration of the grain emission curve at a given fixed frequency rate. For example, a grain emission at 4 Hz lasting 0.75 seconds emits three grains. When grain emission switches from one value to the next, the pulsar corresponding to the new duration plays immediately followed by a silence equal to the period of grain emission. Figure 4.14 plots a rhythm that alternates between fixed-rate pulses, accelerandi, and silence.

Spectra of Basic Pulsar Synthesis

Many time-varying parameters interact to produce the pulsar timbre, including the pulsaret, the pulsaret envelope, the fundamental frequency, multiple formant frequencies, and the burst masking ratio. The spectrum of a single pulsar stream is the convolution product of w and v, biased in frequency by f_d and f_p.

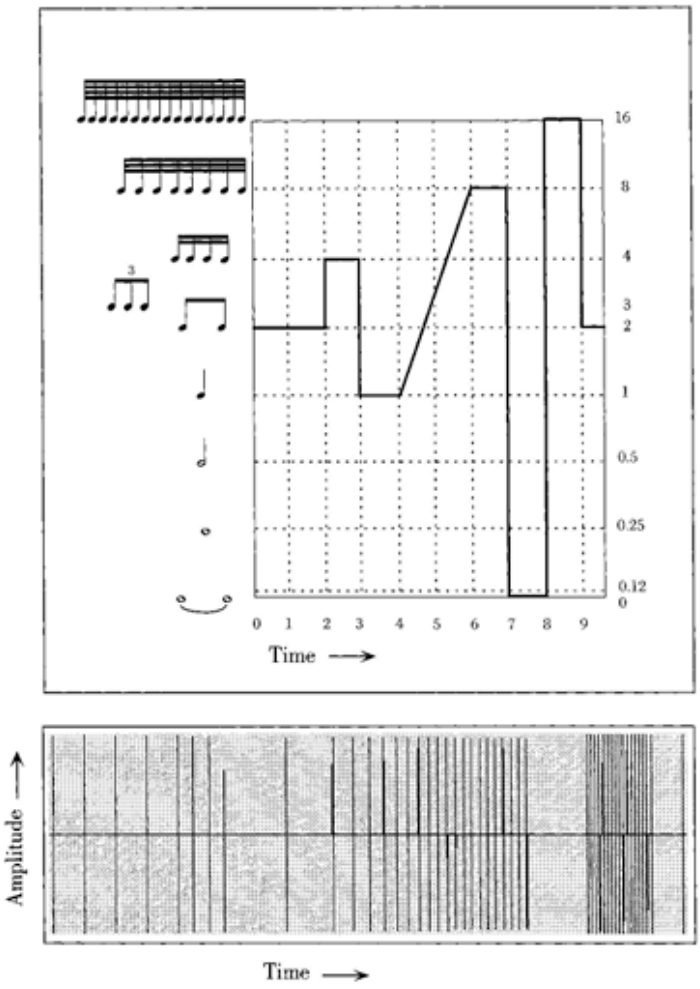


Figure 4.14 Pulsar rhythms. (Top) Pulse graph of rhythm showing rate of pulsar emission (vertical scale) plotted against time (horizontal scale). The left-hand scale measures traditional note values, while the right-hand scale measures frequencies. (Bottom) Time-domain image of generated pulsar train corresponding to the plot above.

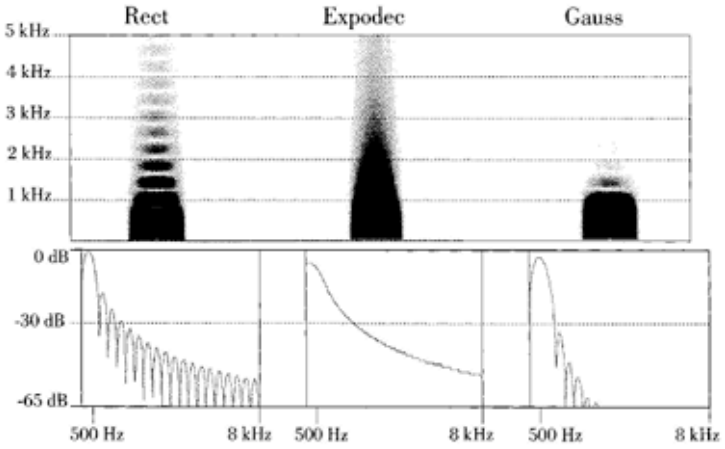


Figure 4.15 Effect of the pulsaret envelope on the spectrum. The top row presents frequency-versus-time sonograms of an individual pulsar with a sinusoidal pulsaret, a fundamental frequency of 12 Hz, and a formant frequency of 500 Hz. The sonograms use 1024-point fast Fourier transform plots with a Von Hann window. They are plotted on a linear frequency scale. From left to right, we see the sonogram produced by a rectangular envelope, an expodec envelope, and a Gaussian envelope. The lower row plots the spectra of these pulsars on a dB scale.

Since w and v can be arbitrary waveforms, and f_d and f_p can vary continuously, the range of spectra produced by PS is quite large.

When the formant frequency is set at a specific frequency energy spreads in that region of the spectrum. Precisely how the energy spreads depends on w and v. The pulsaret waveform w can be considered a template of spectrum shape that repeats at the stipulated fundamental frequency f_p and is time-scaled by the duty cycle or formant frequency f_d. If, for example, the ratio of the amplitudes of the first five harmonics of w is 5:4:3:2:1, this ratio prevails, independent of the specific values of p and d, when $f_p \leq f_d$.

The pulsaret envelope's contribution to the spectrum is significant. Figure 4.15 shows the spectra of individual pulsars where the waveform w is a sinusoid, and the pulsaret envelope v varies among three basic shapes. In the case of figure 4.15a, v is rectangular. Consequently, the formant spectrum takes the form of a broad sinc $(\sin[x]/x)$ function in the frequency domain. The spectrum shows strong peaks at factors of $1.5f_d$, $2.5f_d$, etc., and nulls at harmonics of f_d. This is characteristic of the sinc function. An exponential decay or *expodec* envelope (such as in figure 4.15d) tends to smooth the peaks and valleys in

the spectrum (figure 4.15b). The bell-shaped Gaussian envelope compresses the spectral energy, centering it around the formant frequency (figure 4.15c).

Thus by modifying the pulsaret envelope, one can alter the profile of the pulsar spectrum. The appendix presents a mathematical analysis of the spectra of simple pulsaret envelopes.

Advanced Pulsar Synthesis

Advanced pulsar synthesis builds upon basic pulsar synthesis by adding several features that take it beyond the realm of vintage electronic sonorities. Three methods are of particular importance:

1. Multiple pulsar generators sharing a common fundamental frequency but with individual formant and spatial trajectories
2. Pulse-masking to shape the rhythm of the pulsar train
3. Convolution of pulsar trains with sampled sounds

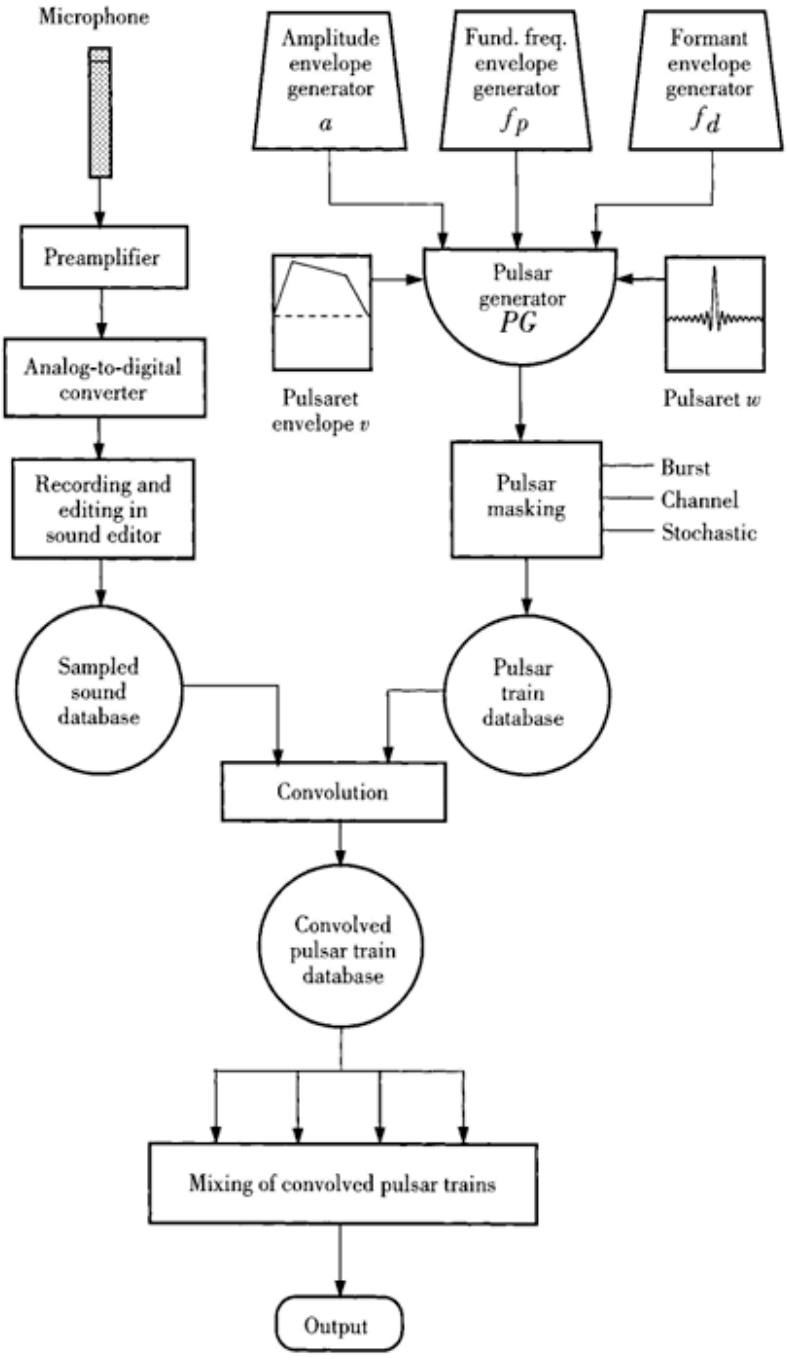
Figure 4.16 outlines the schema of advanced pulsar synthesis. The following sections explain the different parts of this schema.

Multiple Pulsar Generators

A pulsar generator has seven parameters:

- Pulsar train duration
- Pulsar train fundamental frequency envelope f_p
- Pulsaret formant frequency envelope f_d
- Pulsaret waveform w
- Pulsaret envelope v
- Pulsar train amplitude envelope a
- Pulsar train spatial path s

The individual pulsar train is the simplest case. To synthesize a complex sound with several resonance peaks, we can add several pulsar trains with the same fundamental frequency but with different time-varying formant frequencies f_d. One envelope controls their common fundamental frequency, while two or more separate envelopes control their formant trajectories f_{d1}, f_{d2}, etc.


Microphone
Amplitude envelope generator a
Fund. freq. envelope generator f_p
Formant envelope generator f_d
Preamplifier
Pulsar generator PG
Analog-to-digital converter
Pulsaret envelope v
Pulsaret w
Recording and editing in sound editor
Pulsar masking
Burst
Channel
Stochastic
Sampled sound database
Pulsar train database
Convolution
Convolved pulsar train database
Mixing of convolved pulsar trains
Output

A unique feature of pulsar synthesis is that each formant can follow its own spatial path. This leads to complex spatial interplay within a single tone or rhythmic phrase.

Pulsar Masking, Subharmonics, and Long Tonepulses

A pulsar generator emits a metronomic sequence of pulsars, where the rate of emission can vary over time according to the fundamental frequency envelope function f_p. *Pulsar masking* breaks up the stream by introducing intermittancies (regular or irregular) into the metronomic sequence. It deletes individual pulsarets, leaving an interval of silence in their place. This takes three forms: *burst, channel*, and *stochastic masking*.

Burst masking (figure 4.17a) models the burst generators of the classic electronic music studios. It produces a regular pattern of pulsarets that are interrupted at regular intervals. The on/off pattern can be stipulated as the *burst ratio* $b{:}r$, where b is the burst length in pulsaret periods and r is the rest length in pulsaret periods. For example, a $b{:}r$ ratio of $4{:}2$ produces an alternating sequence of four pulsarets and two silent periods: `111100111100111100111100`, etc. If the fundamental frequency is infrasonic, the effect is rhythmic.

When the fundamental is in the audio frequency range, burst masking imposes an amplitude modulation effect on the timbre (figure 4.18), dividing the fundamental frequency by a subharmonic factor $b + r$. With the PulsarGenerator program (described later), we can alter the burst ratio in real time, producing a gamut of subharmonic permutations.

When $b + r$ is large, the subharmonic crosses through the threshold separating tone and rhythm. The result is a series of alternating long tonepulses (at the fundamental pitch) and silent intervals.

Channel masking (figure 4.17b) selectively masks pulsars in two channels, creating a dialog within a phrase by articulating each channel in turn. Figure 4.17b shows two channels only, but we can generalize this scheme to N channels.

Figure 4.16 Schema of pulsar synthesis. A pulsar generator with separate envelope controls for fundamental frequency, formant frequency, amplitude, stochastic masking, and spatial position. In advanced pulsar synthesis, several generators may be linked with separate formant and spatial envelopes. A pulsar stream may be convolved with a sampled sound.

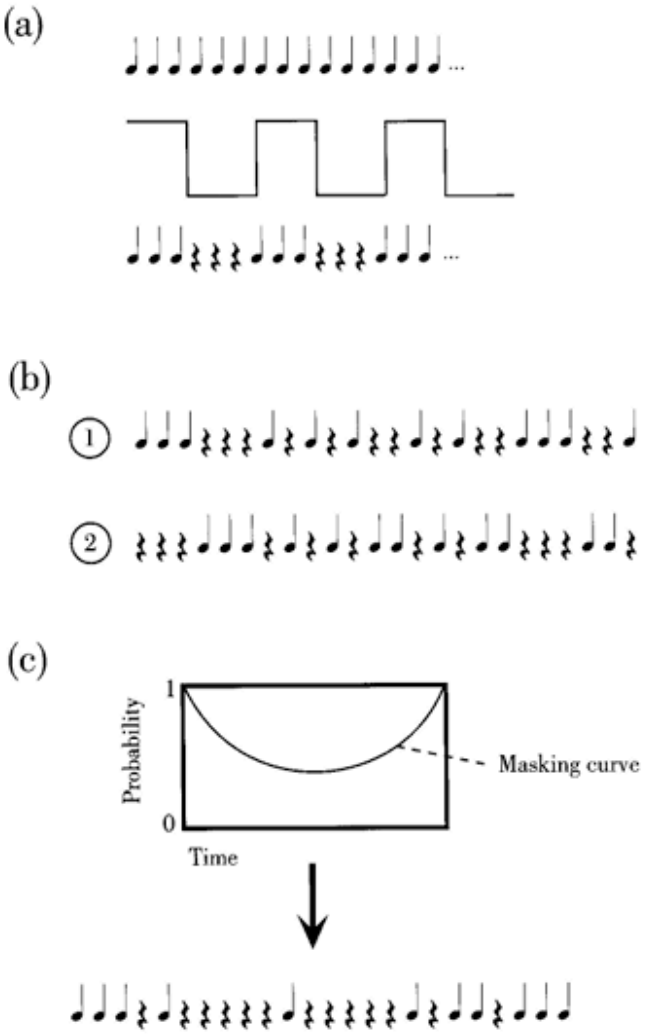


Figure 4.17 Pulsar masking turns a regular train into an irregular train. Pulsars are illustrated as quarter notes, and masked pulsars are indicated as quarter rests. (a) Burst masking. The burst ratio here is 3:3. (b) Channel masking. (c) Stochastic masking according to a probability table. When the probability is 1, there is no masking. When the probability is 0, there are no pulsars. In the middle, the pulsar train is intermittent. Notice the thinning out of the texture as the probability curve dips in the center.

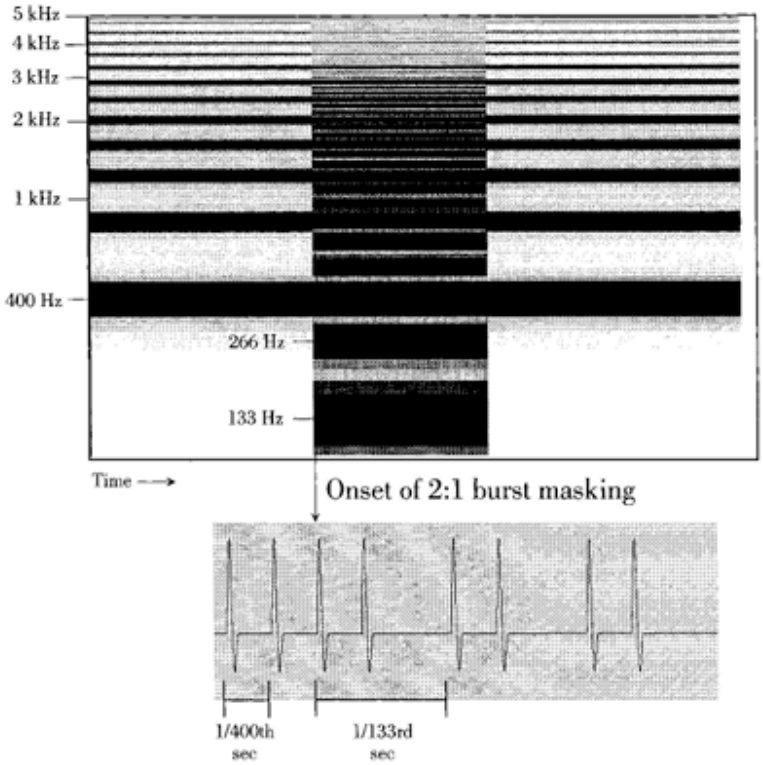


Figure 4.18 Sonogram depicting the effect of burst masking in the audio frequency range. The pulsaret is one cycle of a sinusoid, and the pulsaret envelope is rectangular. The *b:r* ratio is 2:1. The fundamental frequency is 100 Hz and the formant frequency is 400 Hz. Notice the subharmonics at 133 Hz and 266 Hz caused by the extended periodicity of the pulse masking interval (400 Hz/3).

Stochastic masking introduces random intermittancy into the regular stream of pulsars. We have implemented stochastic masking as a weighted probability that a pulsar will be emitted at a particular point in a pulsar train. The probability is expressed as an envelope over the duration of the pulsar train. When the value of the envelope is 1, a pulsar is emitted. If the value is less than 1, it has less possibility. A value of 0 results in no pulsar emissions. Values between 0.9 and 0.8 produces an interesting analog-like intermittancy, as if there were an erratic contact in the synthesis circuit (figure 4.17c).

Convolution of Pulsars with Samples

Pulsar synthesis can be harnessed as a method of sound transformation through convolution. Convolution is fundamental to the physics of waves (Rabiner and Gold 1975). It "crosses" two signals, creating a new signal that combines the time structures and spectra of both inputs. Many transformations emerge from convolution, including exotic filters, spatializers, models of excitation/resonance, and a gamut of temporal transformations (echoes, reverberation, attack-smoothing, rhythm-mapping). Pure convolution, however, has no control parameters, that is, the type of effect achieved depends entirely on the nature of the input signals. See Roads (1992b, 1993a, 1997) for applications of convolution in musical sound transformation.

Sophisticated transformations involving rhythm- and spatial-mapping can be achieved through convolution. It is well known that any series of impulses convolved with a brief sound maps that sound into the time pattern of the impulses. These impulses can be emitted by a pulsar generator. If the pulsar train frequency is in the infrasonic range, then each pulsar is replaced by a copy of the sampled sound object, creating a rhythmic pattern. The convolution of a rhythmic pattern with a sound object causes each impulse to be replaced by a filtered copy of the sound object. Each instance of the sampled object is projected in space according to the spatial location of a specific pulsar's position.

In convolution, each pulsar represents the impulse response of a filter. Thus timbral variations can derive from two factors: (1) filtering effects imposed by the time-varying pulsar train, and (2) overlapping effects caused by convolution with pulsar trains whose fundamental period is shorter than the duration of the sampled sound.

Figure 4.19 shows the temporal and filtering effects of convolution in the form of sonograms. The input signal (b) is the Italian word *qui* (pronounced "kwee"). It convolves with the pulsar train (a) with a variable infrasonic fundamental frequency and a variable audio formant frequency. The resulting convolution (c) combines the time structure and the spectra of the two signals.

The composer can stockpile a database of sampled sound objects for crossing with trains selected from the pulsar database. If the goal of the synthesis is to retain the time structure of the pulsar train (e.g., to maintain a specific rhythm), the sampled sound objects should be of short duration (less than the fundamental period of the pulsar train) and have a sharp attack (a rise time of less than 100 ms). These constraints minimize the time-smearing effects of convolution (Roads 1992b, 1993a, 1997). A good starting point for a sound database

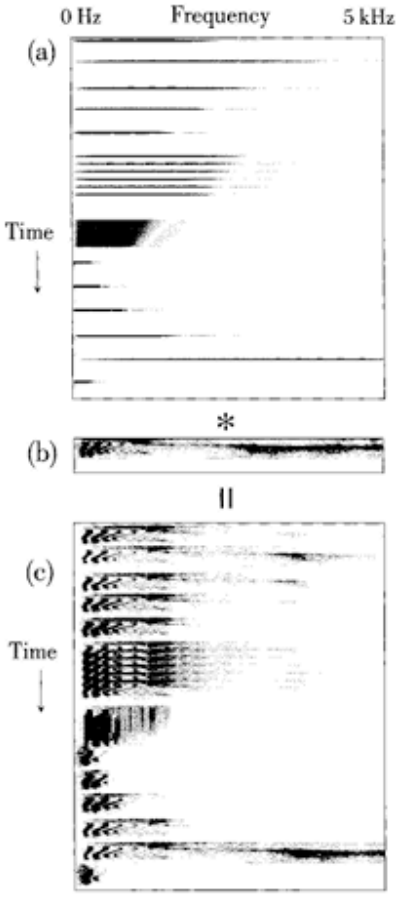


Figure 4.19 Effect of convolution with pulsar train. (a) Infrasonic pulsar train with a variable fundamental and formant frequency. (b) Sampled sound, the Italian word *qui* (pronounced "kwee"). (c) Convolution of (a) and (b).

is a collection of percussion samples. If one seeks a smoother and more continuous texture the constraints can be relaxed. Samples with long durations superimpose multiple copies of the sampled object, creating a rippling sound stream. Samples with slow attacks blur the onset of each sample copy, smearing the stream into a continuum. Thus by controlling the attack shape of the sample one can affect the sonic texture.

Implementations of Pulsar Synthesis

My original implementation of PS dates to 1991, using James McCartney's Synth-O-Matic, a programmable sound synthesis environment for Apple Macintosh computers (McCartney 1990, 1994). In 1996, Mr. McCartney replaced Synth-O-Matic with SuperCollider 1—an object-oriented programming language with a Power Macintosh runtime system (McCartney 1996). Using SuperCollider 1, Stephen T. Pope and I created a new implementation of basic PS in 1997.

With the improved SuperCollider 2 (McCartney 1998), Alberto de Campo and I developed a new realization of pulsar synthesis, presented in a 1999 summer course at the Center for New Music and Audio Technology, University of California, Berkeley. Further refinement of this prototype has led to the PulsarGenerator application, distributed by CREATE. Figure 4.20 shows the graphical interface of PulsarGenerator, version 1. Notice the control envelopes for the synthesis variables. Users can design these envelopes in advance of synthesis, or manipulate them in real time as the instrument plays. We have implemented a scheme for saving and loading these envelopes in groups called *settings*. The program lets one crossfade at a variable rate between settings, which takes performance with PulsarGenerator to another level of musical complexity.

In wave-oriented synthesis techniques, an algorithm loops through a wavetable and varies the signal according to relatively slowly-updated control functions. Thus the efficiency of synthesis corresponds to the number of simultaneous unit generators (oscillators, filters, etc.). In contrast, particle synthesis is more demanding, since the synthesis algorithm must also handle the task of scheduling possibly thousands of events per second, each of which may be unique. The efficiency of pulsar synthesis is therefore related to the rate of particle emission. At infrasonic rates (< 20 pulsars per second), the PulsarGenerator application uses less than 3.6% of the processor on a single-processor Apple G4 running at a 500 MHz clock speed. At high audio rates (such as a three-formant instrument emitting six thousand pulsars per second, corre-

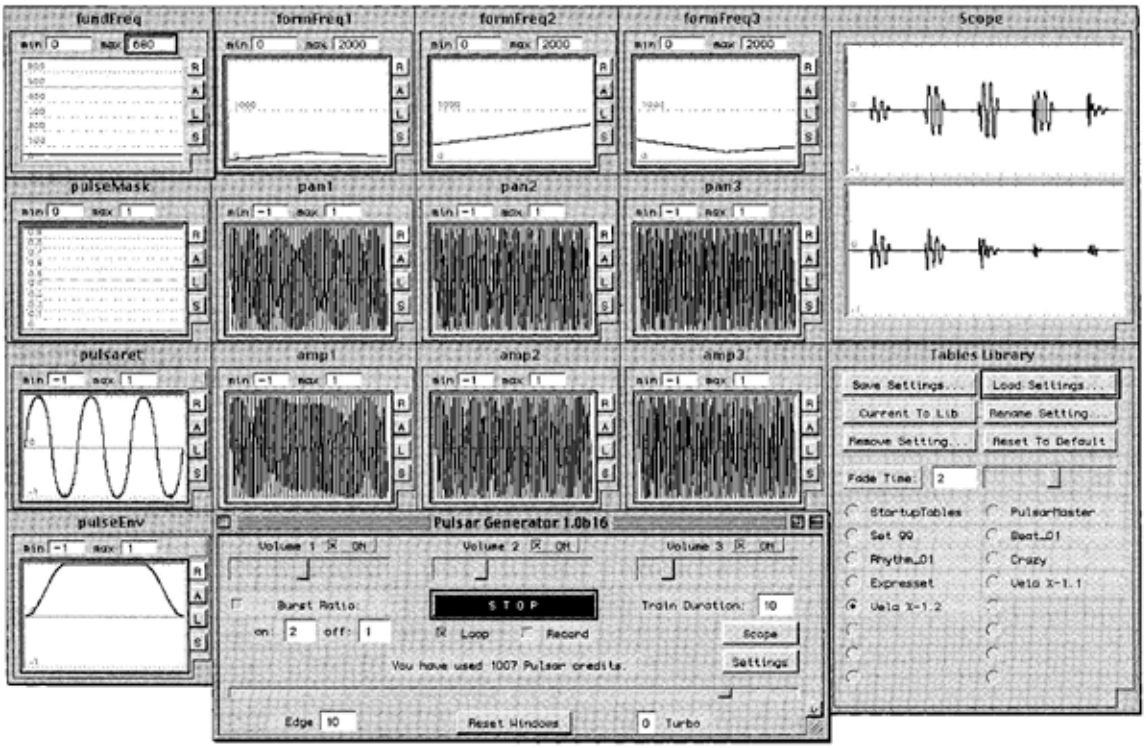


Figure 4.20 Control panel of the PulsarGenerator application by Alberto de Campo and Curtis Roads. Copyright Alberto de Campo, Curtis Roads, and the Regents of the University of California 2000.

sponding to the fundamental frequency of 2 kHz), the application requires approximately 45% of the processor. It is a testimony to SuperCollider 2 that the entire implementation, including the graphical interface, needed less than one thousand and five hundred lines of code and comments. Our code builds the interface, defines the synthesis algorithm, schedules the pulsars, and handles file input and output. McCartney's SCPlay, an efficient real-time sound engine, calculates the samples.

Composing with Pulsars

To interact with PulsarGenerator in real time is to experiment directly with sonic ideas. While experimenting, a composer can save settings and plan how these will be used within a composition. The PulsarGenerator program can also

record the sounds produced in a real-time session. The composer can then edit the session or convolve and mix it with other material.

A final stage of pulsar composition is to merge multiple trains to form a composite texture. This is a question of montage, and is best handled by editing and mixing software designed for this purpose. Each layer of the texture may have its own rhythmic pattern, formant frequency envelope, choice of convolved objects, and spatial path. Working on a variety of time scales, a composer can apply signal processing transformations such as mixing with other sounds, filtering, modulations, and reverberation to individual pulsars, pulsar trains, and pulsar textures.

Musical Applications of Pulsar Synthesis

I developed pulsar synthesis while realizing *Clang-Tint* (Roads 1993b), an electronic music composition commissioned by the Japanese Ministry of Culture (Bunka-cho) and the Kunitachi College of Music, Tokyo. The second movement of this work, entitled *Organic*, focuses on expressive phrasing. It combines bursts of insect noise and animal and bird calls with electronic pulse-tones. The electronic sound palette utilizes pulsar synthesis in many forms: pulsating blips, elongated formant tones, and clouds of asynchronous pulsars. For the latter, I first generated multiple infrasonic pulsar trains, each one beating at a different frequency in the range of 6 to 18 Hz. I then mixed these together to obtain the asynchronous pulsar cloud.

The raw material of my electronic music composition *Half-life*, composed in 1998 and 1999, is a one-minute pulsar train that is wildly varied. Most of the sounds in the rest of the work are derived from this source. *Half-life* extends the pulsar material through processes of granulation, microfiltration, granular pitch-shifting, recirculating feedback echo, individual pulsar amplitude shaping, and selective reverberation. *Tenth vortex* (2000) and *Eleventh vortex* (2001) continue in this direction.

Assessment of Pulsar Synthesis

Music unfolds on multiple time scales, from high-level macrostructure down to a myriad of individual sound objects or notes. Below this level is another hierarchy of time scales. Here are the microsonic particles such as the classical rectangular impulses, grains, wavelets, and pulsars (Roads 1999). Musicians proved the effectiveness of analog impulse generation decades ago. In com-

parison, digital pulsar synthesis offers a flexible choice of waveforms and envelopes, increased precision, and graphical programmable control.

Unlike wave-oriented synthesis techniques, the notion of rhythm is built into techniques based on particles. Rhythm, pitch, and timbre are all interrelated but can be separately controlled. Pulsar synthesis offers a seamless link between the time scales of individual particle rhythms, periodic pitches, and the meso or phrase level of composition. A novel feature of this technique is the generation of multiple independent formant trajectories, each following its own spatial path.

As we have shown, basic pulsar technique can create a broad family of musical structures: singular impulses, rhythmic sequences, continuous tones, time-varying phrases, and beating textures. Pulsar microevents produce rhythmic sequences or, when the density of events is sufficiently high, sustained tones, allowing composition to pass directly from microstructure to mesostructure.

Graphic and Sonographic Synthesis of Microsound

Graphic and sonographic sound synthesis employ visual means to translate images into sound. Here we look at two approaches to interpreting graphical images as sound waveforms: time-domain or graphic, and frequency-domain or sonographic.

Graphic Synthesis in the Time-Domain and Frequency-Domain

Electro-optical synthesizers read waveforms inscribed on rotating tone wheels as in a recent implementation of *photosonic synthesis* using a rotating optical disc with its inscribed time-domain waveform (Duden and Arfib 1990; Arfib, Duden, and Sanchez 1996). In this approach to graphic synthesis, a short waveform repeats many times per second to form a tone.

Another approach is to draw a waveform over an extended period of time. A sound editor presents an image of sound waveforms in a time domain, some programs providing an onscreen pencil for drawing waveforms or smoothing over discontinuities. The time-domain approach to graphic synthesis can be effective on the micro time scale, but requires a great deal of labor. In the era of optical film soundtracks, a few dedicated souls made short pieces using a similar technique. A prime example is Norman McLaren's 1948 *Dots Points*, a 2-minute 47-second film in which both sound and visuals are drawn directly on the film with pen and ink. (See also the transient drawing technique presented

later in this chapter, which creates individual particles by sculpting waveforms in a sound editor.)

Time-domain graphic synthesis lets one design the microstructure of each sound event in detail. However, this approach does not scale well to higher time spans. Acoustical microstructure is delicate. The ear is extremely sensitive to even minor alterations in transient morphology and envelope shape, such as the slope or duration of an attack. Even a slight amplitude discontinuity may cause an audible click or thump. At the same time, the ear tends to ignore permutations in the phase of partials that cause dramatic changes in the shape of the waveform.

Waveform drawing loses usefulness as one zooms out from a waveform display to take in a larger time view. One gains a global view of the sound object envelope, but can no longer see the inner structure of the sound (waveshape, number of iterations, microvariations, etc.), which largely determine its identity.

Another approach to graphic synthesis, which is more powerful on larger time scales, is based on a sonogram frequency-domain representation. To produce a sonogram, a sound is passed through a bank of filters and the output energy from each filter is traced as a function of frequency and time. The sonogram projects frequency on the vertical axis and time on the horizontal axis. The amplitude appears as darkness on a white background, where a darker trace means more energy.

Any visual representation that captures the basic idea of a frequency-versus-time plane is a sonographical notation. Now available are a number of systems, where graphical traces which one can inscribe on a frequency-time grid can be converted directly into sound. In effect one can paint the sound, erase it, or touch it up with all the flexibility of a software paint program. Indeed, several systems let one import images that have been prepared with visual software. The next sections survey three sonographic systems, with emphasis on their microsonic capabilities.

Micro-Arc Synthesis with the UPIC

The UPIC (Unité Polyagogique Informatique de CEMAMu) is a sound synthesis system conceived by Iannis Xenakis and built by engineers at the Centre d'Études de Mathématique et Automatique Musicales (CEMAMu) in Paris (Xenakis 1992). The UPIC system offers both graphical (time-domain) and sonographical (frequency-domain) interfaces for sound composition. In an initial version, dating from 1977, the user interacted by way of a large graphics

tablet, mounted vertically like a painter's easel. The first composition realized with the UPIC was Xenakis's *Mycenae-Alpha* (1980). A major breakthrough for the system was the development of a real-time version, based on a 64-oscillator synthesis engine (Raczinski and Marino 1988). By 1991, engineers had coupled this engine to a personal computer running the Windows operating system, permitting a sophisticated graphical interface (Marino, Raczinski, and Serra 1990; Raczinski, Marino, and Serra 1991). The program now runs stand-alone, with no additional hardware.

At the level of sound microstructure, waveforms and event envelopes can be drawn directly onto the tablet and displayed onscreen. At a higher level of organization, composers can draw the sonographical frequency/time structure of a score page. Lines—called *arcs*—appear on the display screen as one draws with the mouse. Individual arcs can then be moved, stretched or shrunk, cut, copied, or pasted. The arcs on the page can also represent sampled sounds.

In the 1991 version of the UPIC, a page can have sixty-four simultaneous arcs, with four thousand arcs per page. Most importantly, the duration of each page can last from 6 ms to more than two hours. This temporal flexibility lets the user zoom in to the micro time scale. When a page lasts only a second, say, any arcs written onto it will be microsounds. These *micro-arcs* can also be cut, copied, and pasted, as well as stretched or compressed in time and frequency. Moreover, the rate at and direction in which the score is read can be controlled in real time with a mouse. This allows discontinuous jumps from one region of the score to another, for example. The sequence of control motions as it plays a score can be recorded and later the same performance can be replayed or edited.

The UPIC system is an especially pliable musical tool since it integrates many levels of composition within a common user interface. Graphic functions created onscreen can function equally as envelopes, waveforms, pitch-time scores, tempo curves, or performance trajectories. This uniform treatment of composition data at every level should be extended to more computer music systems.

Synthesis of Microsound in Phonogramme

Vincent Lebros developed Phonogramme in 1993 at the Université de Paris VIII (Lesbros 1995, 1996). Phonogramme offers an approach to graphical synthesis with some similarities to the UPIC, but offering a number of extensions. First, the program can generate sound directly from a MacOS computer, or it can generate MIDI data to be sent to a bank of synthesizers. Second, the pro-

gram offers a *harmonic pen*, which draws a number of harmonically related partials above the arc inscribed by the user. Users can adjust the number and strength of these partials before drawing. The harmonic pen solves a common problem of sonographic synthesis: a single arc is not a complete description of a sound object. As a sonogram of a concrète sound clearly indicates, it takes many traces on the sonographic plane to create a sound object with a fundamental frequency, formant structure, harmonic/inharmonic partials, and inner variations. Finally, unlike the 1991 UPIC system, Phonogramme can analyze a sound and generate an editable sonographic representation.

Phonogramme's score page can be interpreted on any time scale chosen by the user. A line or arc inscribed can be interpreted as a microsonic particle. It is also possible to turn the drawing pencil into a kind of dot spray where a fast gesture made with the mouse leaves behind not a continuous line but a series of microsonic dots (figure 4.21).

Synthesis of Microsound in MetaSynth

The MetaSynth program by (Wenger and Spiegel 1999) is a recent implementation of sonographical synthesis. It operates on a MacOS computer with real-time playback (or delayed playback for complex scores). Although it does not incorporate every feature of predecessors such as UPIC and Phonogramme, it does offer unique features that take graphical synthesis to a new level of power and expressivity.

Among the drawing implements included with MetaSynth are some tools for microsound synthesis, including a granular spray brush (figure 4.22). The spray brush determines the time and frequency span of the grains. The grain envelope is fixed as a triangle. In another mode, it can play back a line drawing with a granular instrument. The waveform of the grains is variable on a per-page basis, from a synthetic waveform to a sampled waveform.

MetaSynth is a major contribution to sonographic synthesis in that it greatly expands the palette of brushes and other implements to paint sound color. It also permits nongraphical granulation of sound files in its Effects window. (See chapter 5 for more on granulation of sound files.)

Assessment of Graphic and Sonographic Synthesis of Microsound

Graphic and sonographic synthesis share a common interface: drawing. What their drawings represent are quite different, however. Graphic synthesis depicts

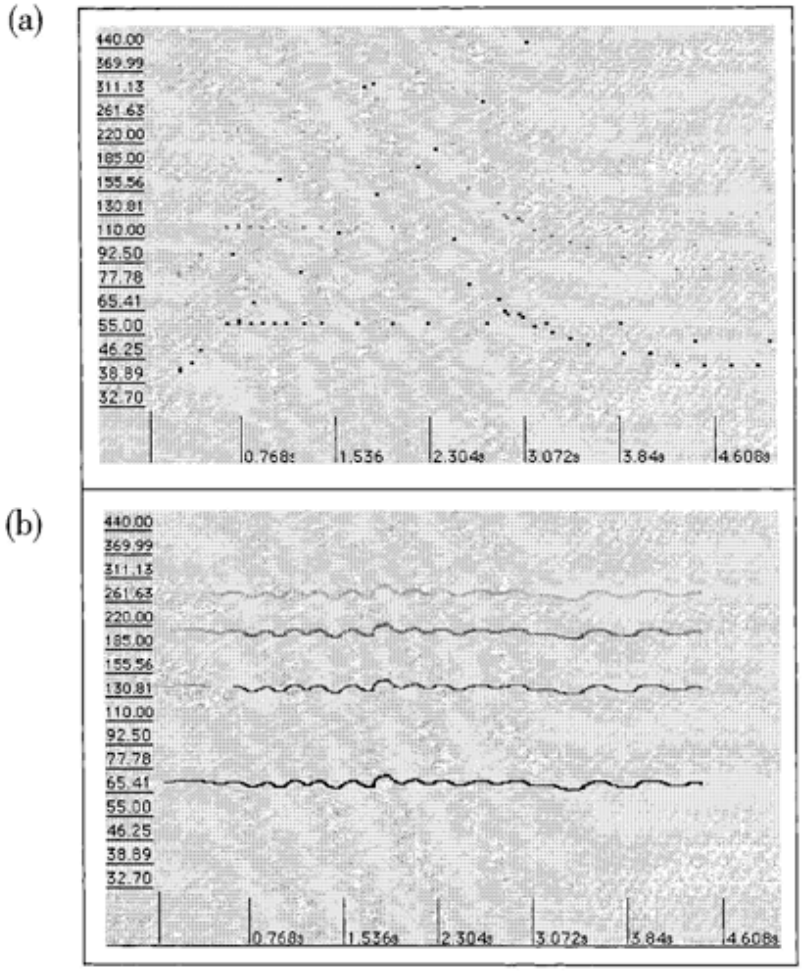


Figure 4.21 Phonogramme scores. Both scores are just over 4.6 seconds in length. (a) Fast horizontal gestures leave behind a stream of micro-arcs. Notice the four harmonics superimposed over the original low-frequency gestures by the harmonic pencil. (b) Slow hand movements create continuous tones.

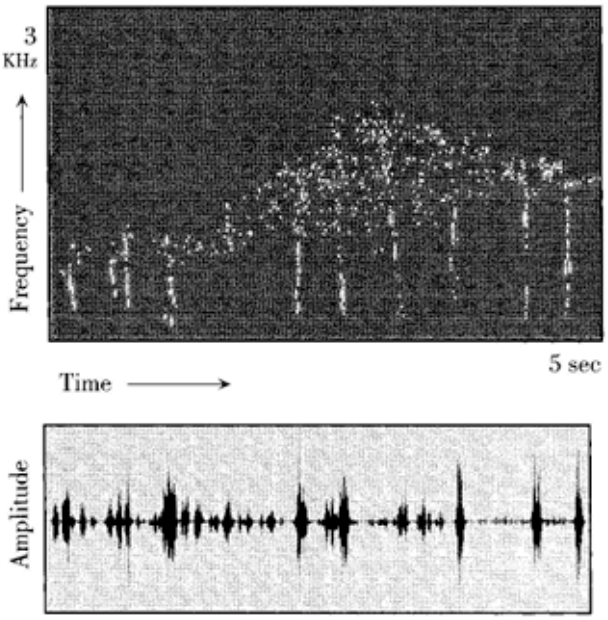


Figure 4.22 Sonographical synthesis of particles in MetaSynth. (a) The particles were drawn by hand with a spray brush tool. This particle score can be played back with many different waveforms, including sampled sounds. (b) Time-domain view of the waveform.

time-domain envelopes and waveforms. To draw pitch curves and envelope shapes with graphic tools is a supple and efficient procedure. It is easy to create shapes and phrases that would be difficult to achieve by other means.

To draw time-domain waveforms, however, is problematic. It is difficult to infer a sound directly from a view of its waveform. This is further complicated by the fact that the same sound can be represented graphically by innumerable ways whose only differences are in the phase relationships among the frequency components. Furthermore, any waveform repeated without variation takes on a static quality. So, as in other approaches to synthesis, waveform generation in graphic synthesis systems has shifted from individual fixed waveforms to evolving sources such as sampled sounds or groups of time-varying waveforms.

Sonographic sound synthesis is a direct and intuitive approach to sound sculpture. Interaction with sonographical synthesis can be either precise or imprecise, depending on how the user treats the process. A composer who plans

each line and its mapping into sound can obtain precise results. It is just as worthwhile to treat the medium as a sketch pad, where initial drawings are later refined into a finished design.

Unlike traditional notation, which requires serious study for a long period of time, a child can learn the relationship between drawn gestures and sound in minutes. This initial simplicity hides a deeper complexity, however. As with any technique, the best results demand a long period of study and experimentation.

As mentioned earlier, a single arc is not a description of a complete sound object. An arc is only one component of a complex time-varying sound. Such sounds require many arcs. We can see this complex nature in the sonograms of relatively simple instrumental tones, such as a cello. Noiser timbres, such as a cymbal or gong, display enormous complexity. They seem to be composed of globules of energy that sometimes connect and at other times break apart.

As such representations become more commonplace, a long process of codification—from complex sonographic patterns into abstract iconic notation—seems to be inevitable. The Acousmagraphe system, developed at the Groupe de Recherches Musicale (Paris), points in this direction (Desantos 1997).

Particle-Based Formant Synthesis

A *formant* is a peak of energy in a spectrum, which can include both harmonic and inharmonic partials as well as noise. Formant peaks are a characteristic of the spoken vowel sounds and the tone color of many musical instruments. Within the range of 0 to 5000 Hz, the vocal tract has five formant regions. Formant regions act as a kind of "spectral signature" or "timbral cue" to the source of many sounds. The formants of a voice or an instrument are not fixed, however, they drift according to the frequency of the fundamental (Luce 1963). Furthermore, formants are only one clue the ear uses to identify the source of a tone.

Fully understanding the nature of formants in human speech has long been a goal of scientific research. Ingenious methods for synthesizing the formants of vowel-like tones have been developed (Tyndall 1875; Miller 1916), and it is not surprising that this research has served as a wellspring of ideas for musical formant synthesis.

The rest of this section presents concise descriptions of three particle synthesis techniques: formant wave-function or FOF synthesis, Vosim, and window-function (WF) synthesis. FOF and Vosim evolved from attempts to simulate

human speech, whereas WF was developed to emulate the formants of traditional musical instruments. For more detailed descriptions, see the references and Roads (1996).

FOF Synthesis

Formant wave-function synthesis (*fonction d'onde formantique* or FOF) generates a stream of grains, each separated by a quantum of time, corresponding to the period of the fundamental frequency. So a single note produced by this technique contains hundreds of FOF grains. The definitive FOF grain is a sine wave with either a steep or smooth attack and a quasi-exponential decay.

The envelope of a FOF grain is local, that of the entire note is global. The local envelope of a FOF grain is defined as follows. For the attack portion of the FOF grain, $0 \leq t \leq tex$, the envelope is:

$$env_t = 1/2 \times [1 - \cos(\pi_t / tex)] \times \exp(-atten_t)$$

For the decay portion, $t \geq tex$, the envelope is:

$$env_t = \exp(-atten_t)$$

π is the initial phase of the FOF signal, *tex* is the attack time of the local envelope, and *atten* is the decay time (D'Allessandro and Rodet 1989). The effect is that of a damped sinusoidal burst, each FOF grain lasting just a few milliseconds. The convolution of the brief FOF envelope with the sinusoid contributes audible sidebands around the sine wave, creating a formant spectrum. The spectrum of the damped sine generator is equivalent to the frequency response curve of one of the bandpass filters and the result of summing several FOF generators is a spectrum with several formant peaks.

Each FOF generator is controlled by many parameters. Among these are the formant parameters $p1$ through $p4$:

$p1$ is the center frequency of the formant.

$p2$ is the formant bandwidth, defined as the width between the points that are -6 dB from the peak of the formant.

$p3$ is the peak amplitude of the formant.

$p4$ is the width of the *formant skirt*. The formant skirt is the lower part of the formant peak, about 40 dB below the peak, akin to the foothills of a mountain. The skirt parameter is independent of the formant bandwidth, which specifies the breadth at the peak of the mountain.

The inherent connection between time-domain and frequency-domain operations is exemplified in the way FOF parameters are specified. Two of the main formant parameters are specified in the time domain as properties of the envelope of the FOF grain. First, the duration of the FOF attack controls parameter $p4$, the width of the formant skirt, that is, as the duration of the attack lengthens, the skirtwidth narrows. Second, the duration of the FOF decay determines $p2$, the formant bandwidth. Hence a long decay length translates into a sharp resonance peak, while a short decay widens the bandwidth of the signal.

The basic sound production model embedded in FOF synthesis is the voice. However, users can tune many parameters to move beyond vocal synthesis toward synthetic effects and emulations of instruments (Bennett and Rodet 1989).

Typical applications of FOF synthesis configure several FOF generators in parallel. Some implementations are very complicated, with over 60 parameters to be specified for each sound event. The CHANT program, developed in the 1980s, was proposed as a response to this complexity, providing a collection of rules for controlling multiple FOF streams in parallel (Rodet, Potard, and Barrière 1984).

Vosim

Like FOF synthesis, the Vosim technique generates a series of short-duration particles in order to produce a formant effect. Vosim synthesis was developed by Werner Kaegi and Stan Tempelaars at the Institute of Sonology in Utrecht during the early 1970s (Kaegi 1973, 1974; Tempelaars 1976, 1977, 1996). Vosim generates a series of tonebursts, producing a strong formant component. Like FOF, it was originally designed for vowel sounds, and later extended to model vocal fricatives—consonants such as [sh]—and quasi-instrumental tones (Kaegi and Tempelaars 1978).

The Vosim waveform approximates the signal generated by the human voice in the form of a series of pulsetrains, where each pulse is the square of a sine function. The parameter A sets the amplitude of the highest pulse. Each of the pulsetrains contains N $\sin^2$ pulses in series decreasing in amplitude by a decay factor b. The width (duration) of each pulse T determines the position of the formant spectrum. A variable-length delay M follows each pulse train, which contributes to the pulsetrain's overall period, and thus determines the fundamental frequency period. The period is $(N \times T) + M$, so for seven pulses of

Table 4.3 VOSIM parameters

Name	Description
T	Pulsewidth
δT	Increment or decrement of T
M	Delay following a series of pulses
δM	Increment or decrement of M
D	Maximum deviation of M
A	Amplitude of the first pulse
δA	Increment or decrement of A
b	Attentuation constant for the series of pulses
N	Number of pulses per period
S	Type of modulation (sine or random)
NM	Modulation rate
NP	Number of periods

200 μsec duration and a delay equal to 900 μsec, the total period is 3 ms and the fundamental frequency is 333.33 Hz. The formant centers at 5000 Hz.

Two strong percepts emerge from the typical Vosim signal: a fundamental corresponding to the repetition frequency of the entire signal, and a formant peak in the spectrum corresponding to the pulsewidth of the $\sin^2$ pulses. A Vosim oscillator produces one formant. In order to create a sound with several formants, it is necessary to mix the outputs of several Vosim oscillators.

Table 4.3 lists the set of parameters that control the Vosim oscillator. T, M, N, A, and b are the primary parameters. By modulating the delay period M, one can produce vibrato, frequency modulation, and noise sounds. Kaegi and Tempelaars introduced three additional variables: S, D, and NM, corresponding respectively to the type of modulation (sine or random), the maximum frequency deviation, and the modulation rate. They wanted also to be able to provide for "transitional" sounds, which led to the introduction of the variables NP, δT, δM, and δA. These are the positive and negative increments of T, M, and A, respectively, within the number of periods NP.

By changing the value of the pulsewidth T, the formant changes in time. This is *formant shifting*, a different effect than the progressive spectral enrichment which occurs in, for example, frequency modulation synthesis. The Vosim signal is not bandlimited, but spectral components are greater than 60 dB down at six times the fundamental frequency (Tempelaars 1976).

Window Function Synthesis

Window function (WF) synthesis generates pulsestreams that result in formants of purely harmonic content (Bass and Goeddel 1981). It begins with the creation of a broadband harmonic signal. Then a weighting stage emphasizes or attenuates different harmonics in this signal to create time-varying formant regions which can emulate the harmonic spectra of traditional instruments.

The building block of WF synthesis is a particle known as a *window function pulse*. A typical window function is a smooth pulse with a bell-shaped envelope. Window spectra exhibit a characteristic center lobe and side lobes. The center lobe is typically much higher in amplitude than the side lobes, meaning that the signal is bandlimited. In the Blackman-Harris window function chosen by Bass and Goeddel, the frequencies in the side lobes are down at least 60 dB.

Bass and Goeddel created a broadband signal by linking a periodic series of WF pulses separated by a period of zero amplitude. For different fundamental frequencies, the duration of the WF pulse stays the same; only the interpulse silence varies. In this use of a pulse followed by a period of deadtime, the WF technique is similar to pulsar synthesis, Vosim, and the FOF method explained earlier. In other respects, however, the techniques are not similar.

In WF synthesis, the number of harmonics increases as the fundamental frequency decreases. This is because the higher harmonics fall outside the center lobe of the WF pulse's spectrum. Thus, low tones are timbrally rich, while high tones are less so. This is characteristic of some traditional instruments such as pipe organs and pianos, which Bass and Goeddel wanted to simulate. Note that not all instruments—for example, the harpsichord—exhibit this behavior. Furthermore, some instruments do not have a purely harmonic spectrum and so make poor models for the WF technique.

To create formant regions in the spectrum requires a further processing called *slot weighting*. A *time slot* is defined as the duration of a single WF pulse plus a portion of its deadtime. The timbre of the sound can be altered by weighting the slots (i.e., multiplying a slot by a value) with a periodic sequence of N slot weights. This feeds a stream of WF pulses to a multiplier as an input signal, along with a periodic stream of slot weights. The multiplier computes the product of each input pulse with a specific weight, making an output stream of WF pulses at different amplitudes. The spectrum of such a stream exhibits peaks and valleys at various frequencies. For time-varying timbres, each slot weight can be specified as a time-varying function.

WF synthesis requires an amplitude compensation scheme, because low frequencies contain few pulses and much zero-amplitude deadtime or silence, while high frequencies contain many pulses and almost no deadtime. A quasi-linear scaling function adjusts the amplitude as an inverse function of frequency. That is, low tones are emphasized and high tones are attentuated for equal balance throughout the frequency range.

Assessment of Particle-Based Formant Synthesis Techniques

Particle-based formant synthesis models a class of natural mechanisms which resonate when excited, and are quickly damped by physical forces. A typical example would be a stroke on a woodblock—a sound which cuts off almost immediately. The result is a grain-like "pop." Another example is the glottal pulse, which the vocal tract filters. Continuous tones string together a series of such particles.

FOF synthesis has been available within the widely distributed Csound language for some time (Boulanger 2000). The Common Lisp Music language (Schottstaedt 2000) includes a wavetrain object able to realize both FOF and Vosim synthesis. Window Function synthesis technique was experimental, and since its original realization, has not continued.

Vocal-like tones can be simulated by mimicking the fast impulses that continuously excite resonance in the vocal tract. Realistic simulation from a particle technique, however, requires an enormous investment of time. In the 1980s, vocal synthesis using the FOF technique performed the "Queen of the Night" aria from Mozart's *Magic Flute*. The realization of this 30-second fragment took months of effort.

In the simulation of vocal and instrumental tone, the particle representation should be invisible. If we divorce these techniques from their original uses, we can see that particle-based formant synthesis remains a rich resource for synthetic timbres, both at the infrasonic frequency level—where it produces a wide variety of rhythmic pops—and also at the audio frequency level, where it generates expressive resonant tones.

Synthesis by Transient Drawing

Synthesis by *transient drawing* is a direct method. It takes two forms: *synthetic* and *transformational*. Synthetic transient drawing requires a graphical sound

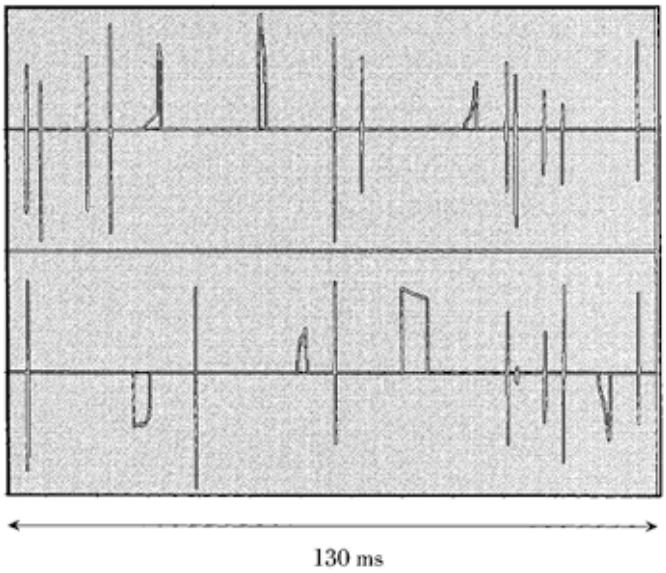


Figure 4.23 Transient wave writing in a sound editor. Notice the hand-drawn transients interspersed with narrow computer-generated impulses.

editor which provides a waveform pencil that can be used to manually inscribe the waveform of a transient. These transients can be interspersed with waveforms produced by other means (figure 4.23).

Transformational transient drawing also uses a sound editor, after beginning with an existing low-level audio signal, such as background noise or the tail of a reverberant envelope. A brief extract of this nonstationary signal is selected and rescaled to a much higher amplitude, creating a transient particle (figure 4.24). This particle can be reshaped using the implements of the editor, such as narrowband filtering, envelope reshaping, phase inversion, and spatialisation. The following frequency bands are especially important in filtering the transient particles.

1. Direct current (DC) cut—removal of frequencies below 20 Hz
2. Deep bass cut or boost—80 Hz
3. Mid-bass boominess cut—200 Hz
4. Low-mid boost—500–700 Hz resonances
5. Mid harshness cut—1–2 kHz

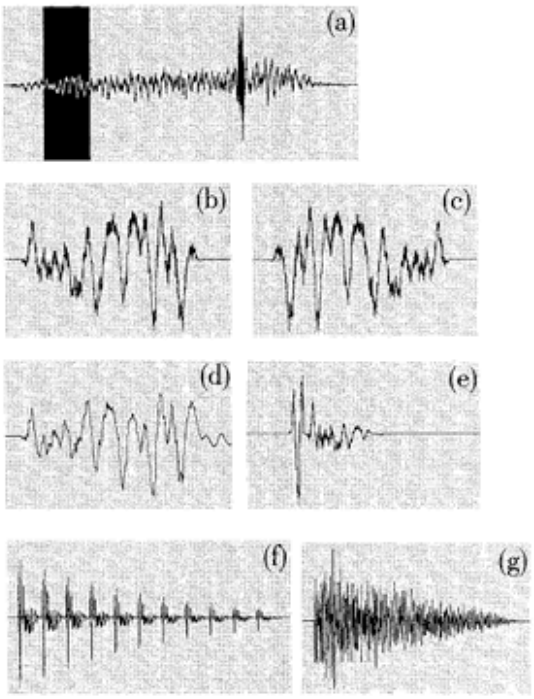


Figure 4.24 Manipulations of a transient wave extracted from an ambient texture. (a) Ambient noise texture. The darkened part is a 120-ms transient that we extract as a transient wave. (b) Detail of the extracted transient. (c) Reversal of (b). (d) Lowpass filtered (smoothed) version of (b). (e) Re-enveloped and pitch-shifted copy of (b), now lasting less than 50 ms. (f) Ten-fold replication (cloning) of (e) to form a decaying sound object. (g) Copy of (b) that has been passed through a bank of narrow filters to form a pitched tone. The attack was edited to create a sharper onset.

6. Crackle boost—notches at 3 kHz and 6 kHz
7. High harshness removal—lowpass at 5 kHz
8. Brilliance boost—12 kHz

Filters 2, 4, 6, and 8 accentuate the particle, while filters 1, 3, 5, and 7 attenuate it in the harsh frequency zones. Successive micro operations can radically transform the original signal. For example, a sharp narrow filter applied to a brief noise particle transforms it into a resonating pure tone (figure 4.24g). If we apply advanced techniques such as time-stretching, there comes a point at which we can transform any transient into a myriad of sounds on any time scale.

Assessment of Transient Drawing

Transient drawing takes compositional decision-making down to a microscopic level, starting from almost nothing and obliging the composer to handcraft each individual particle in a phrase. The extreme differentiation resulting from this approach leads to a high probability of *singularities* or unique sound events. In this sense, transient drawing maximizes the information content of the musical signal.

Both the synthetic and the transformational approaches to transient drawing are labor-intensive, relying on manual editing. In effect, the composer works in a manner similar to a Pointillist painter such as Georges Seurat, building an image from thousands of tiny brush strokes. In Seurat's later paintings (for example, *La Luzerne, Saint-Denis*) the particles can be scattered within dense masses. When one of these particles appears isolated by silence, its singularity conveys a striking impact.

As we saw in the section on graphic synthesis, waveform drawing makes sense only on a micro time scale. It does not translate well to higher time scales. This is because zooming out on a waveform display provides a global view of the sound object envelope at the expense of the inner microstructure (waveform shape, number of iterations, etc.).

Transient drawing has become an integral part of my compositional practice. Many examples of it appear in *Clang-Tint*, especially in the movement *Organic*. I also made extensive use of the technique in the first movement of *Half-life*, in which a number of singularities appear. Perhaps the most practical use of transient drawing is for fabricating such singularities to be scattered over sparse textures. In order to build continuous tones from a single transient, I used transient drawing in conjunction with particle cloning (described next).

Particle Cloning Synthesis

In particle cloning synthesis, a brief tone pip, or a longer continuous or fluttering tone, is made by repeatedly cloning a single sound particle. This particle may derive from any synthesis technique, including transient drawing. Figures 4.24f and 4.25 illustrate particle cloning. Here is a recipe for particle cloning:

1. Generate a single particle by any means.
2. Sculpt the particle to the desired shape and duration in a waveform editor.

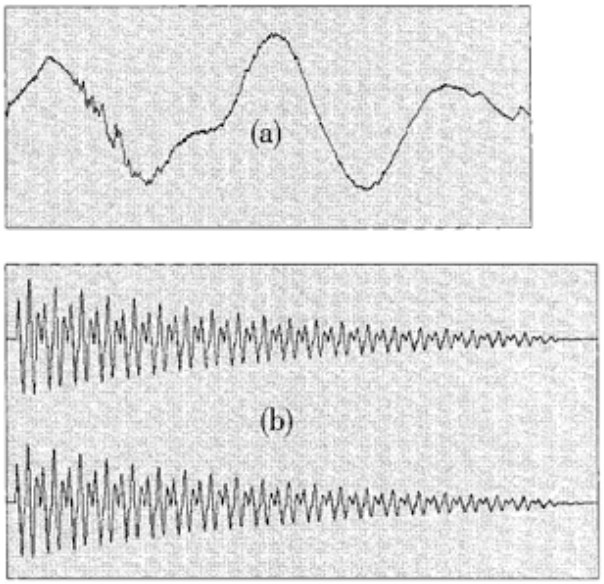


Figure 4.25 Particle cloning synthesis. (a) Solo particle, lasting 35 ms, extracted from an acoustic sample of a drum sound. (b) A 200 ms sound object formed by cloning (a) 50 times, pitch-shifting it up two octaves, creating another channel slightly delayed, and applying an exponentially decaying amplitude curve.

3. Clone the particle and repeat it over a specified duration to form a tone pip. The number of particles cloned corresponds to the total duration divided by the duration of the particle. The resulting pitch depends on the period of the particle.
4. Shape the amplitude envelope of the resulting tone pip.
5. Pitch-shift the tone pip to the desired pitch, with or without time correction.
6. Apply one or more bandpass filters. The important bands to control are the low (50–150 Hz), low-mid (150–300 Hz, narrowband), mid (500–900 Hz), mid-high (3–4 kHz, wide bandwidth), and high (9–12 kHz, wide bandwidth) ranges.

In stage 3, we used a Replicate function in a sound editor to automate the process. Replicate fills an arbitrary duration with the contents of the clipboard. Thus it is possible to copy a particle, select a 3-second region, and fill the region with a sequence of cloned particles. Obviously, the frequency of the tone pip depends on the period of the particle, and so a 10-ms particle produces a tone

pip at a pitch of 100 Hz. By selecting some of the silent samples around a particle, one can shift the fundamental frequency downward. The resulting tone can then be transposed to any desired pitch. If the period of the particle is greater than about 50 ms, the texture is no longer continuous, but flutters.

Stages 4, 5, and 6 foster heterogeneity among tones cloned from the same particle. Each tone can have a unique duration, amplitude envelope, pitch, and spectrum weighting.

Assessment of Particle Cloning Synthesis

The construction of tones by particle cloning elevates the time scale of a particle from the micro level to the sound object level, and so fosters the emergence of singularities on the sound object time scale. As we have implemented it, it is a manual technique, carried out with a sound editor. I developed the particle cloning method in the course of composing *Half-life* (1998–99), in which the initial particles derive from pulsar synthesis or transient drawing. These particles stand out prominently in part 1 of the piece, in the melodic section at time 2:04–2:11.

Physical Models of Particles

Computer sound synthesis rests on the premise that a software program can model a sound-generating process. *Physical modeling* (PhM) synthesis starts from a mathematical description of acoustic sound production (Roads 1996). That is, the equations of PhM describe the mechanical and acoustic behavior of an instrument as it is played. The goals of physical modeling synthesis are both scientific and artistic. PhM investigates the extent to which equations and algorithms can simulate the sound-producing mechanisms of existing instruments. In this sense, a physical model embodies the Newtonian ideal of a mathematical model of a complicated mechanico-acoustic process. PhM can also create sounds of fancifully imaginary instruments, otherwise impossible to realize.

PhM synthesis can be traced back to the mathematical descriptions of John William Strutt (Lord Rayleigh). His insightful volume *The Theory of Sound* (1894/1945) laid the foundation for a century of research. For many years, efforts in physical modeling focused on synthesis of continuous tones. Only

```
/************************ MARACA ****************************/

#define MARA_SOUND_DECAY 0.95
#define MARA_SYSTEM_DECAY 0.999
#define MARA_NUM_BEANS 25

void maraca_setup() {
  num_objects = MARA_NUM_BEANS;
  gain = log(num_objects) / log(4.0) * 40.0 / (MY_FLOAT)
num_objects;
  coeffs[0] = -0.96 * 2.0 * cos(3200.0 * TWO_PI / SRATE);
  coeffs[1] = 0.96*0.96;
  soundDecay = MARA_SOUND_DECAY;
  systemDecay = MARA_SYSTEM_DECAY;
}

MY_FLOAT maraca_tick() {
  MY_FLOAT data;
  shakeEnergy *= systemDecay;          // Exponential system decay
  if (my_random(1024) < num_objects)   // If collision
    sndLevel += gain * shakeEnergy;    //   add energy
  input = sndLevel * noise_tick();     // Actual Sound is Random
  sndLevel *= soundDecay;              // Exponential Sound decay
  input -= output[0]*coeffs[0];        // Do gourd
  input -= output[1]*coeffs[1];        //   resonance
  output[1] = output[0];               //     filter
  output[0] = input;                   //       calculations
  data = output[0] - output[1];        // Extra zero for shape
  return data;
}
```

Figure 4.26 Perry Cook's model of a maraca, coded using the Synthesis Toolkit in the C++ language. Notice the declaration at the top indicating the number of beans in the shaker. A statement in a score file (not shown) triggers the maraca model.

more recently has attention turned to synthesis using physical models of the particulated sounds of certain percussion instruments and environmental microsounds. Perry Cook's Physically Informed Stochastic Event Modeling (PhISEM) exemplifies this approach. PhISEM is a suite of programs to simulate the sounds of shaken and scraped percussion such as maracas (figure 4.26), sekere, cabasa, bamboo windchime, tambourine, sleighbells, and guiro (Cook 1996, 1997). Cook also developed a model to simulate the sound of water drops based on the same principles, and suggested that this technique could also synthesize the sound of feet crunching on gravel, or ice cubes in a shaken glass. (See also Keller and Truax (1998) and the discussion of physical models for granular synthesis in chapter 3).

The common thread among instruments modeled by PhISEM is that sound results from discrete microevents. At the core of PhISEM are particle models.

Basic Newtonian equations governing the motion and collision of point masses produce the sounds. For shaken percussion instruments, the algorithm assumes multiple individual sound sources, for example, that with maracas each of them contains many beans. It calculates the probability of bean collisions; very high after a shake, and rapidly decaying. If a bean collision occurs, it is simulated by a burst of exponentially decaying noise. All collision noises pass through a sharply tuned bandpass filter, which simulates the resonance of the gourd.

Assessment of Physical Models of Particles

Many percussion instruments create their distinctive timbre through the accumulation of microsounds. In the case of shaken instruments, this can be modeled as a stochastic process. Physical models of such instruments produce granular sounds, either sparse or dense.

My experiences with this technique were based on the MacOS synthesis program Syd (Bumgardner 1997), which realized Cook's maracas model. It allowed control of the resonance frequency, the resonance pole, the probability of bean collisions, the system decay, and the sound decay. Under certain conditions, these sounds can evoke acoustic instruments recorded in an anechoic environment. When the range of parameter settings is extended, sounds that are not particularly realistic, but still interesting are created.

The physical modelling of particles could go much further. As Chapter 3 points out, a vast scientific literature devoted to models of granular processes has yet to be harnessed for sound synthesis.

Abstract Models of Particles

One can take PhM a step further, and bypass simulations of the acoustic instrument world. Software allows one to develop models of abstract systems which can only exist within the memory of the machine. This is a well-established approach to digital synthesis. Several common synthesis techniques arise out of efficient computational models only indirectly related to acoustical tone production. The technique of frequency modulation (FM) synthesis, for example, can readily be seen as an abstract model (Chowning 1973). FM synthesis is based on a familiar equation in the theory of radio communications. After much effort, John Chowning tested it in the audio frequency range and then tuned the synthesis equation to make the results sound somewhat like tradi-

tional instruments. Only when the equation is heavily constrained, however, does it resemble a recognizable instrument, otherwise it produces abstract "radiosonic" timbres.

Abstract models can also control particle synthesis. Chapter 3 described control schemes for granular synthesis based on recursive and chaotic algorithms. The next section discusses an even more abstract approach, where the particles are part of the model, and the sound is a side-effect of their interactions.

Abstract Particle Synthesis

To *sonify* is to translate data into sound. Abstract models sonify particle interactions. Thus the abstract model links particle interactions to sound generation. The sound produced by an abstract model may not sound particulated, it may instead be long and continuous, as in the work of Sturm (1999). Sturm programmed a model of atomic particles which acted according to the laws of physics. When the particles interacted in their environment, they created a unique solution which evolved over time. By using de Broglie's equation—stipulating a relationship between particles and waves—this simulation produced long sine waves. In theory, any signal can be represented by a unique set of particles with specific initial conditions and a particular time-variant potential. Sturm hypothesized that granular sounds could be modeled by the motions of a particle within a box with its interior lined with sound samples.

Xenakis's GENDYN system is another example of the abstract approach (Xenakis 1992; Serra 1992). Instead of using a model imported from atomic physics, however, Xenakis invented a nonlinear algorithm for simulating the behavior of a particle bouncing off elastic barriers. The path traced by the particle is interpreted directly as a soundwave. The waveform is the trace of the particle's chaotic path as it bounces back and forth between the barriers while the model's parameters are changing constantly.

We should also mention the Cosmophone project (Calvet, et al. 2000). The Cosmophone is a device for detection and sonification of cosmic ray bombardments which works by coupling particle detectors to a real-time data acquisition and sound synthesis system. The sonic output of the system goes to a multichannel loudspeaker configuration. The information received by the detectors triggers the emission of sounds. The sounds emitted by the system are programmable. Clearly, various methods of particle synthesis would be well worth testing with such a system.

Table 4.4 Sound particles

Name	Envelope type	Waveform	Characteristics
Grains	Gaussian or arbitrary	Arbitrary, including sampled	Can be scattered irregularly or metrically; each grain is potentially unique; Gaussian grains are compatible with analysis systems like the Gabor transform and the phase vocoder
Glissons	Gaussian or arbitrary	Arbitrary	Variable frequency trajectory; synthesizes glissandi or noise textures
Pulsars	(1) Rectangular around pulsaret, then null; (2) Gaussian (3) Expodec (4) Arbitrary	Arbitrary	Independent control of fundamental and formant spectrum; can also be applied in the infrasonic frequencies as a rhythm generator; synchronous distribution of pulsars
Trainlets	Expodec or arbitrary	Impulse	Used to synthesize tones; offers independent control of fundamental and formant spectrum
Wavelets	Gaussian or other, subject to mathematical constraints	Sinusoidal	Particle duration varies with frequency; starts from an analysis of an existing source sound
Grainlets	Gaussian or arbitrary	Sine	Interdependent synthesis parameters
Micro-arcs	Arbitrary	Arbitrary, including sampled	Flexible graphic design; subject to graphical transformations
FOF grains	Attack, sustain, release	Sine	Envelope controls formant spectrum
Vosim grains	Linear attack, exponential decay	Sine^2 pulses	Used for pitched tones; flexible control of spectrum
Window-function pulses	Rectangular around pulse then nil	Blackman-Harris pulse	Synthesizes formants with purely harmonic content
Transient drawing	Arbitrary	Hand-drawn	Generally sharp and percussive; each transwave is unique
Particle cloning	Arbitrary	Arbitrary, including sampled	Repeats a particle so that it becomes a continuous tone

Assessment of Abstract Models of Particles

In most implementations, abstract models of particles have been used to produce a pressure wave representing a continuous sound. Most of Sturm's sound examples sound like the additive synthesis of sine waves, where each sine is warping or sweeping in frequency, usually in synchrony with the other sines. Xenakis's compositions using the GENDYN system are quite different in sonic quality from the sound of Sturm's examples: the waveforms are jagged in shape and rich in spectrum, their behavior is obsessive and erratic, always wavering and warbling, often explosive. Xenakis's *S.709* (1992, EMF CD 003) is a classic example, meandering wildly like some atomic insect.

In abstract models, the mapping between the particle model and the process of sound generation may be quite arbitrary. The history of digital sound synthesis contains numerous examples of all manner of algorithmic schemes applied to the control of sound synthesis. Thus one could easily use an abstract model of particles to control the parameters of an arbitrary synthesis technique; frequency modulation or granular synthesis, for example.

Summary

All forms of music composition—from the freely improvised to the formally organized—are constrained by their sound materials. The urge to expand the field of sound comes from a desire to enrich compositional possibilities, and much can be gained from the harvest of synthetic waveforms produced by particle synthesis. In chapter 3 and in this chapter, we have looked at a variety of sound particles. Chapter 6 describes additional particles derived from windowed spectrum analysis. Table 4.4 summarizes the variety of sound particles studied in this book.

Artists are frequently recalled to the belief that the splendors of nature surpass anything that human beings can create. The natural world, however, did not inhibit the first painters, on the contrary, it inspired them. Similarly, for composers, the omnipresence of natural sound does nothing to quash the need to create a virtual sound world. With the sound particles, we cultivate a new strain of culture within the natural order.