WEATHER ORGAN: A SPARSE STOCHASTIC SYNTHESIZER IN FAUST

Mykle Hansen

MSL
Portland, Oregon, USA
mykle@mykle.com

ABSTRACT

Weather Organ is a Faust instrument for manipulating sparse noise to synthesize non-tonal soundscapes, with a focus on reproducing certain stochastic natural sound sources. Its structure, algorithms, and various interesting uses are described.

1. INTRODUCTION

This paper describes Weather Organ, a Faust instrument for manipulating sparse noise to imitate slowly-changing natural sound sources such as rain, wind, surf, fire, Geiger counters and volcanic activity, and to create interesting new textures of sound. It is the product of various discoveries made while using Faust to explore the definition, synthesis and experience of acoustic noise while, simultaneously using the exploration of noise as as focus for learning Faust.

The use of synthetic noise to simulate noisy natural sounds is one of the oldest tools in sound design. In the 1920s, sound effects pioneers Ora and Arthur Nichols constructed a machine that used the hiss of pressurized air to create the sound of ocean waves, gunshots and birdsong to accompany silent films; later, at CBS Radio's first-ever sound effects department, they used the white noise produced by a modified air conditioner to imitate the sound of a jet engine for Orson Welles' 1939 Mercury Theater broadcast of "War Of The Worlds"[1]. In 1961, Harald Bode[2] described noise generators as one of the fundamental components of music synthesis, and the filtration of noise as one of the fundamental strategies for producing un-pitched synthetic tones in the percussion units of early home organs. In 1965, J. Pierce [3] described how Bell Labs researchers were already exploring the use of noise on a slower time-scale, using pseudorandom numbers to organize the pitches and durations of notes in computer-generated musical compositions along lines first proposed by Iannis Xenakis in [4] .

Modern digital synthesis gives this approach the utmost precision; Farnell [5] gives detailed analyses of the physical processes that produce the sounds of fire, rain, running water, and surf (among many others) and describes PureData recipes to simulate those sounds, making heavy use of pseudorandom noise at various time resolutions. Weather Organ is not the product of a depth of knowledge or research comparable to Farnell's work; the design process was an impressionistic effort to bend an evolving noise experiment toward natural sounds. Nevertheless, it interestingly demonstrates that the human ear does not require such complete detail in order to recognize such sounds. In the same way that low-fidelity audio recordings can provide a convincing illusion of musicians present in the room, it may be that for natural sound sources caused by many small impulses, the character of time distribution of those impulses is the fingerprint by which they are recognized.

When the density of noise is increased to the point that individual clicks aren't audible to the ear, Weather Organ follows a long tradition of the use of constant noise to synthesize sounds of heavy wind, rain or surf. The use of sparse noise to generate sparse environmental sounds, such as a light rain or a mild breeze, is less common, but quite powerful. Sparse noise manipulation offers an algorithmically simpler and computationally less expensive tool than Iannis Xenakis' dynamic stochastic synthesis process or modern digital granular synthesis; it replaces complex waveform manipulation and sample processing with simple statistical manipulations and the rolling of dice. But it still offers an expensive means to explore Xenakis' concepts of sonic granularity, particularly the perceptual threshold between discrete and continuous sound.

Throughout this paper I will mention various ways in which features of the instrument can evoke natural acoustic phenomena. These are purely subjective associations, no more scientific than a wine review, but I hope they are inspiring to those who might choose to explore the instrument.

In any case, Weather Organ demonstrates some models for using sparse noise to synthesize stochastic sounds which may be sufficiently convincing for some uses, while being less complex, and likely less compute-intensive, than the more sophisticated models of [4] and [5]. It also allows for the freeform exploration of other interesting textures of stochastic sound which have no natural analog known to this author at this time.

2. OVERVIEW

Figure 1 shows a simplified flow diagram of the components of Weather Organ: a noise generator is fed to a filter bank, while various sources of randomness with different time resolutions and distributions (*flutter* and *drift*) are used to modulate the parameters of the generator and the filters. A final output section provides gain control.

3. NOISE GENERATION

Weather Engine's noise generator emits samples of *noise*, either generated internally in white or brown spectral *colors*, or taken from an *external* source, at a *sparseness* from 0.1Hz to the sample rate, with *periodicity* continuously variable from predictable to random, and with *width* from 1 sample to the period of *sparseness*. The result can be enhanced with *grit*. All of these italic terms are defined below. In these descriptions, a +1 impulse emitted by a trigger function is called a *trigger event*, while a short-duration noise generated by multiplying such a trigger event with a noise signal is called a *noise event*.

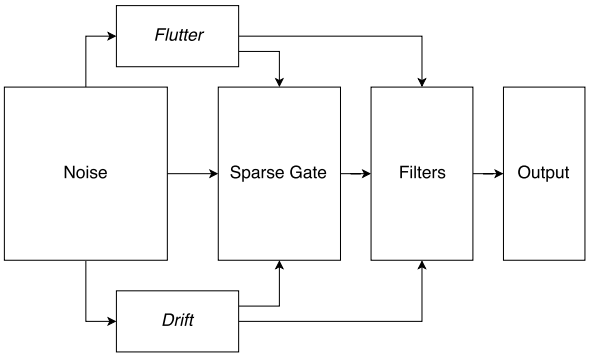

Figure 1: *Top-level organization of Weather Organ.*

3.1. *Noise*

The definition of *noise* is fascinatingly complex and surprisingly subjective across all the domains where the word is used. Rather than unpack all that, I use the word generally in this paper to mean *unpitched sounds*, but specifically in this section to mean *digital pure white noise*, a stream of randomly distributed floating-point values with a running average approaching zero, with flat frequency spectrum (except when noted as brown) and flat amplitude distribution (except when noted as Gaussian). `multinoise` from Faust's `noises.lib` is the main random generator and ultimate source of all noise in Weather Organ. Because it uses multiple streams of random values in various ways simultaneously, the de-correlation of multiple random streams provided by `multinoise` is audibly important for achieving a wider range of sounds.

3.2. *Sparseness*

Sparseness is the inverse of Density, when Density refers to the mean density of noise events produced by the noise generator; *sparse noise* is noise in which that density is relatively low. Weather Organ uses the same types of *sparse noise* as produced by the `sparse_noise` function of `noises.lib`: a constant stream of zeroes interrupted at random intervals by single samples of random value, i.e. single-sample noise events. The mean density of noise events, `f0`, measured in average-events-per-second, is the only argument taken by `sparse_noise`. The algorithm used by that function guarantees that one noise event will occur at some moment during every time interval of `f0`.

Weather Organ's noise generator is similar to `sparse_noise`, but with three enhancements.

- The triggering algorithm for sparse noise generation is in a seperate function, `sparse_periodic_trigger`, which emits +1 impulses. This function's output signal is combined with one of several noise sources to produce different flavors of sparse noise.
- The random number source used by `sparse_periodic_trigger` to randomize generated events is taken as an argument, so that multiple instances may be de-correlated.
- The third enhancement, *periodicity*, is described below.

3.3. *Periodicity*

`sparse_periodic_trigger` also takes a *coefficient of periodicity* argument (c), a value between 0 and 1 which affects the distribution of random events. When set to 0, the position of the event within the `f0` period is entirely random, and the event distribution is the same as of `sparse_noise`. When set to 1, events are emitted exactly at the start of each period. Values between 0 and 1 produce intermediate degrees of periodicity. This is useful for imitating semi-rhythmic noise sources such as boiling water or dripping taps.

3.4. *Color*

Weather Organ generates *sparse periodic white noise* as described above, by multiplying a Gaussian noise stream with a stream of events from `sparse_periodic_trigger`. To produce *sparse periodic brown noise*, `sparse_periodic_trigger` is used with Faust's `sAndH` to sample and hold a noise signal. The result is a wave that changes instantaneously from one random value to another at each noise event. Between events, the wave holds a constant DC offset. In Figure 2, FFT analysis in Gnu Octave shows that the while the harmonic spectrum of non-zero impulses is *white* (in the common sense of being flat across the frequency range from sub-audible to Nyquist), the spectrum of each shift between DC offsets over the timeframe of a single sample is *brown*, i.e. it contains a $1/f^2$ distribution of frequencies across the audible spectrum. This is not a characteristic of the noise input signal; the individual input samples, surrounded by silence, have no spectral tilt of their own. It is the return to zero after an impulse that causes the impulse to be white, or the absence of a return to zero that causes it to be brown. However, as the mean event density `f0` rises, and the gaps between noise events get smaller, the output takes on more of the spectrum of the input. Therefore brown noise input would be ideal to produce a brown-tinted output across all values of `f0`. However, for various reasons[1], Weather Organ uses the `pink_noise` function of `noises.lib` here instead of a proper brown noise generator.

3.5. External Noise Source

As a third option, Weather Organ can take any input signal and convert it to sparse periodic brown noise, by running it through the

[1]Mainly: While first developing Weather Organ I mis-identified the spectrum of sampled & held noise as $1/f$ (pink) instead of $1/f^2$ (brown); thankfully, an astute reviewer caught this error in an earlier draft of this paper. While revising the paper I experimented with using brown noise here, generated with Faust's `spectral_tilt` function, to better match the spectrum of sparse periodic brown noise. But after some listening in which I applied drift to the density of brown noise, I found that the resulting all-brown sound, although closer to what I had intended, did not sound as interesting as what I had mistakenly produced by sampling & holding pink noise. In the accidental implementation, as the density of brown noise increases, the spectrum undergoes a gradual shift from brown to pink. To my ear this seems to evoke a phenomenon of the sound of waves breaking on a shoreline, in which the higher frequencies are reduced when the wave is farther away, but become brighter as the water gets closer. This suggests that flux-animated spectral tilt would be an interesting feature; I hope to explore that soon.

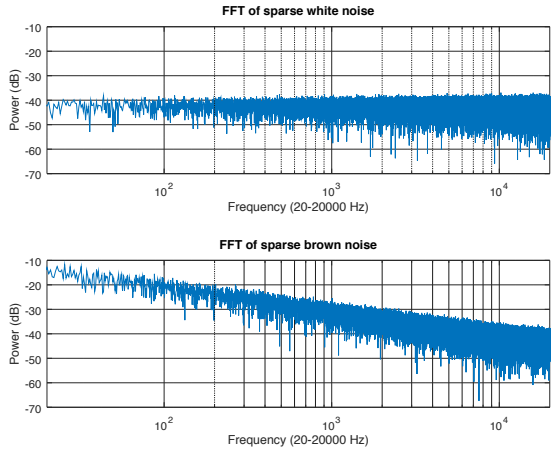

Figure 2: *Harmonic analyses of ten seconds of sparse noise.* The upper graph was produced with `sparse_noise()` from noises.lib. The lower graph was generated with the sparse periodic brown noise algorithm with zero periodicity. The mean event frequency `f0` was 10Hz in both cases.

same sample and hold signal pathway described above for brown noise. As the gate width and event frequency are adjusted, this feature can give the input signal some of the sonic characteristics of an imperfectly-tuned radio.

Interesting physical interactions with Weather Organ can be had when it is configured in a feedback loop between a microphone and a loudspeaker. Below a certain noise density, the sparseness of the passed signal prevents out-of-control feedback, yet room acoustics, speaker placement and other audible site phenomena still produce a recognizable coloration. If a handheld microphone is used, certain settings of Weather Organ can produce not only the sound of a Geiger counter but also the physical experience of using a handheld instrument to seek out concentrations of invisible energy in a three dimensional space. (This technique works best in an enclosed space; in effect, Weather Organ is used to inspect the modes of a room coupled with an audio system.)

3.6. *Width*

The noise generator also takes a *width* parameter in seconds, to increase the number of noise samples in each noise event from a minimum of 1 sample. This is accomplished by applying `sAndH` with a countdown timer to the output of `sparse_periodic_trigger`, then using the resulting signal to control either the white or brown algorithm described above. The resulting bursts of sparse noise, though still too brief to be perceived as continuous, contain an audible texture that can evoke the momentarily-complex sound of a raindrop striking a puddle.

3.7. *Grit*

The output of the noise generator is normalized internally between the values of +1 and -1. It can be driven louder, without exceeding those bounds, by pushing all output samples closer to the bounds of that interval. This is done by multiplying the sign of each sample with the absolute value of that value raised to an exponent between 0 and 1. An exponent of 1 does nothing; an exponent of 0 drives each sample to either +1 or -1. Fractional exponents have an intermediate effect. The *grit* parameter is that exponent. Its effect is similar to that of an audio compressor.

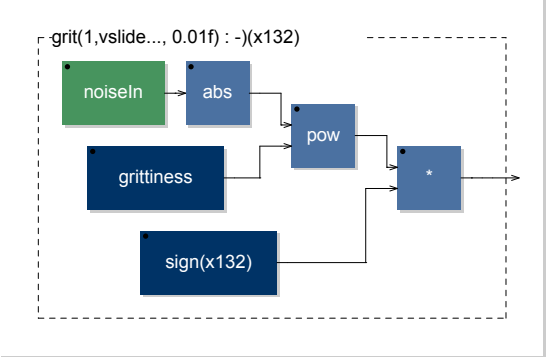

Figure 3: Grit algorithm.

4. FILTERS

Weather Organ feeds noise events through a bank of three synchronized filters. This is only one of many possible ways to add timbre to noise events; other approaches will be explored in the future. Faust's `va.moog_vcf` lowpass-filter model (described by Stilson and Smith in [6]), while CPU-expensive compared to the rest of Noise Organ, is straightforward to use and, at higher Q values, exhibits a not-unpleasant ringing tone analogous to the sound of a resonant object struck by a momentary force.

4.1. Fundamental Filter

The fundamental filter has an adjustable Q and cutoff frequency. The fundamental cutoff can be changed by MIDI or OSC note messages; combined with sufficiently high values of Q, this allows Weather Organ to be played as a polyphonic tuned musical instrument.

4.2. Overtone Filters

The two overtone filters share a base Q value with the fundamental filter. Their cutoff frequencies are adjustable multiples of the fundamental cutoff. As the fundamental Q and cutoff change, either by direct manipulation or through its Flutter and Drift controls (described below), the overtone filters change in sync. In this way, the three filters can give each noise event a complex harmonic signature that remains constant as the fundamental filter's cutoff changes. Each overtone filter also has its own Flutter and Drift controls (described below), that can fluctuate independently of the fundamental if desired. This can create liquid gurgling tones, or the sound of tumbling shards of brittle resonant materials such as glass or earthenware.

5. FLUX SOURCES

Weather Organ uses two sources of flux, *flutter* and *drift*, to produce random adjustments to most of its parameters. Most parameters have individual flutter and drift controls, allowing the degree

of flux to be adjusted for each parameter. The fluctuations are applied, either through addition to or exponentiation as appropriate, of the base value of the parameter in question.

5.1. *Flutter*

Flutter is a straightforward random coefficient that is synchronized with the noise engine; a parameter with flutter applied will change, discontinuously, by a random degree within an adjustable range, in sync with each noise event produced by the noise engine. Flutter is held constant between noise events, because any discontinuous changes during those periods would inject extra noise in the output signal.

Although all flutter changes are correlated in time, they are uncorrelated in value. Each parameter's flutter is calculated using randomness from an independent, uncorrelated Gaussian noise stream.

5.2. *Drift*

Drift is a slow-changing, continuous variation applied to a parameter, intended to imitate the fluctuations in slow-shifting noise phenomena such as wind or surf. To accomplish this, a *drift wave* is created by applying a constant-slope attack-release envelope to a very-low-frequency sparse brown noise signal, using the same sparse brown noise algorithm described in section 3.2 above (i.e. having a random constant DC offset that changes discontinuously and randomly) at a mean event density between 0.1 seconds and ten minutes. The envelope's attack and release slopes are made proportional to that density, such that they would have just enough time to open and close on a constant +1/-1 square wave of the same frequency as the density. When these envelopes are applied to the sparse brown noise signal, the resulting drift wave alternates between motion and stillness. Applying this value as an offset to the sparseness of the noise generator, and/or the Q with of the filter bank, can produces a recognizable simulation of the random turbulence in wind.

In contrast with flutter, all the drift-enabled parameters of Weather Organ share a single drift wave. This allows, for instance, the same drift effect to be applied to both noise density and filter Q, increasing the verisimilitude of the synthetic wind. For additional flexibility, all drift knobs allow either a positive or negative offset, so that some parameters may be driven low as others are driven high.

6. CONTROLS

Figure 4 shows the Weather Organ user interface, implemented with Faust's UI primitives. It is divided into four sections: Weather, Noise, Filter and Output.

6.1. Weather

The Weather section contains two parameters which together affect all flutter and drift settings in the system. It also meters the realtime activity of the flutter and drift generators at current settings.

Flux is a percentage between 0 and 200, by which all flutter sources and the main drift wave are multiplied. This one slider can reduce or increase all flutter and drift in the instrument in one place. Impressionistically speaking, it adjusts a coefficient of anxiety, from calm to panic.

Turbulence controls the average frequency of the drift wave, which in turn controls the rate with which drift settings change. It is named for the effect it has on Weather Organ's simulations of wind.

6.2. Noise

The Noise section controls the noise generator. A musician or meteorologist may choose the noise source, adjust the sparse noise density, width and rhythm, and apply grit to the result. Density can have drift. Width can have both drift and flutter.

6.3. Filter

The Filter section contains controls for the frequency and Q of the fundamental filter, the overtone ratio and levels of the two overtone filters, and drift and flutter controls. The drift and flutter controls of the overtone filters adjust two parameters: the overtone ratio, and overtone Q relative to the fundamental Q. There is also a simple low shelf filter in this section, useful for softening the impact of sparse brown noise on subwoofers.

6.4. Output

The output section provides a global gate and gain control, and an optional limiter (specifically, `limiter_1176_R4_mono` from `compressors.lib`), a comforting feature for owners of fragile loudspeakers.

7. CONCLUSIONS

I wandered into this project while researching an article about the modern uses of acoustic noise. Today, high-fidelity recordings of noise, both natural and synthetic, are a quantifiable commodity of growing popularity, available for download or on DVD from a variety of sources. Many people find that certain kinds of constant acoustic noise help them to sleep, to concentrate, or to escape more unpleasant sounds. However, there appears to be no single form of noise that works to everyone's benefit. It's my hope that Weather Organ might become a tool to help individual noise-users find the specific noise that comforts them.

Weather Organ is still evolving. The source code for this and future versions of Weather Organ is available online at `https://github.com/myklemykle/weather_organ`, for free use and modification under a Creative Commons license.

8. ACKNOWLEDGMENTS

Many thanks to the members of the faudiostream-users and faudiostream-devel mailing lists, who have been most gracious with their time and advice while I've been learning and using Faust. Thanks also to the early reviewers of this paper.

9. REFERENCES

[1] Robert L Mott, *Radio Sound Effects: Who Did It, and How, in the Era of Live Broadcasting*, McFarland, 2008.

[2] Harald Bode, "Sound synthesizer creates new musical effects," *Electronics*, 1961.

[3] John Pierce, "Portrait of the computer as a young artist," *Playboy*, 1965.

[4] Iannis Xenakis, *Musiques formelles*, Richard-Masse, 1963.

[5] Andy Farnell, *Designing sound*, MIT Press, 2010.

[6] Tim Stilson and Julius O. Smith, "Analyzing the Moog VCF with considerations for digital implementation," 1996.

Figure 4: *The controls of Weather Organ.*