Obiwannabe

Use the source...

Sponsored by the number 7

Hoorah!

Our task is to build a crowd, a cheery room full of people clapping and making a noise. This is an example that is about microstructure, about creating shaping functions on very small and big scales. We will also go into a little more depth exploring some different types/classes of parametric relationships. I'm also going to introduce more of the concept of a sounds *texture* which leads us into granular synthesis.

Granular synthesis is sound synthesis using lots of little pieces of sound to make another larger, longer and different one. We could make a trumpet by scattering thousands of tiny grains of trumpet-like pieces over a period of time so that the mixture of them becomes a great trumpet sound. Working out the type, flavour and sequence of all the grains can become complicated hard work, so we like to use analysis tools to help sift and prepare the grains for more advanced compositions. But you do not need to appreciate the real theory and often complicated practices behind granular synthesis to start using it practically in simple ways. Merely understanding the methodology of creating sounds from many tiny fragments is enough to get started. Using this attitude to sound creation a designer can produce sounds using quite basic grain-like generators and see what happens when layers of different textures are blended and faded. A cool thing about a granular approach is the possibilty of making interesting shifts and changes that depart vastly from a physical basis. In other words it's great for unworldly noises, morphologies and imaginitive sounds.

Applause

A microstructure is a small detail or feature that comes up in the big overall pattern. Our unit of scale is a single person who can clap their hands or whistle or yell. Some sounds are fractal in nature because their small scale structure resembles their structure on a bigger scale, but other sounds are composed of distint layers of detail and this is more the case here. Structure can be on so many levels each with their own variables, but in a sound like a room full of clapping people its a phenomenon with a lot of (statistically speaking) constants. The room doesn't change size, the people stand still, they clap at an individual rate not in unison (unless they're being

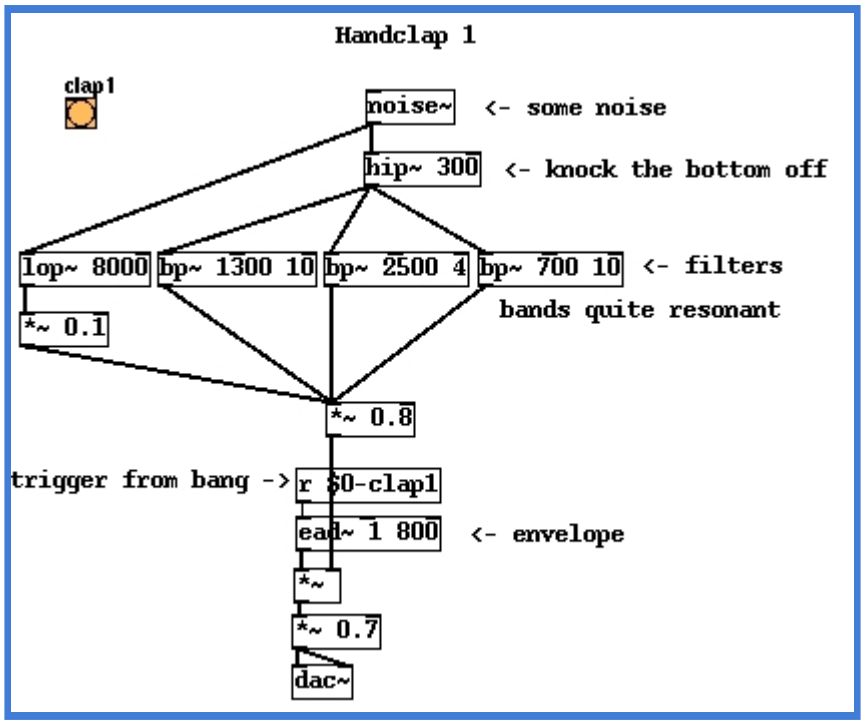
rude in some culture, but it's not universal by any means because it's okay to do that in Spain or Mexico or to clap in unison to build up an invitation.) Let us invent a hypothetical parameter called intensity which captures everything needed, the vigour and loudness of each person clapping in a grand ensemble.

Each clap itself is shory burst of energy around a narrow resonance, determined in several ways including, the size of hands, force with which they are brought together and shape of the hands (flat, cupped, palm to palm). At some point we cross from hearing individual claps in the first seconds to perceiving a solid wall of noise. Where does that point lie? It comes party at a point where the envelopes of each event overlap one another by some critical degree, and is partly determined by two absolute numbers of about 20ms and 100ms. Events smaller than 20ms are atoms, we can't distinguish any time detail smaller than that with our ears, we just hear clicks of various colours. At around 100ms we start to stream events automatically into the same channel, ascribing them to one common source or extent and losing focus on the parts in favour of a bigger picture. This is the first barrier of granular/texture synthesis we are going to cross. In the chapter on parameterisation and level of detail we will revisit this very important theory. For now let's stick with making single events and trying to put them in a random ensemble.

At this point we might naively think, let's build a single clapper, give it a rate that speeds up and then down and instantiate a room full of them. Before we got anything much good we would have a hundred or so instances. That might be fine, we might have built a very efficient clap unit, but sooner or later this approach is going to just run us out of CPU cycles. Let's do it anyway to prove a point.

One hand clapping

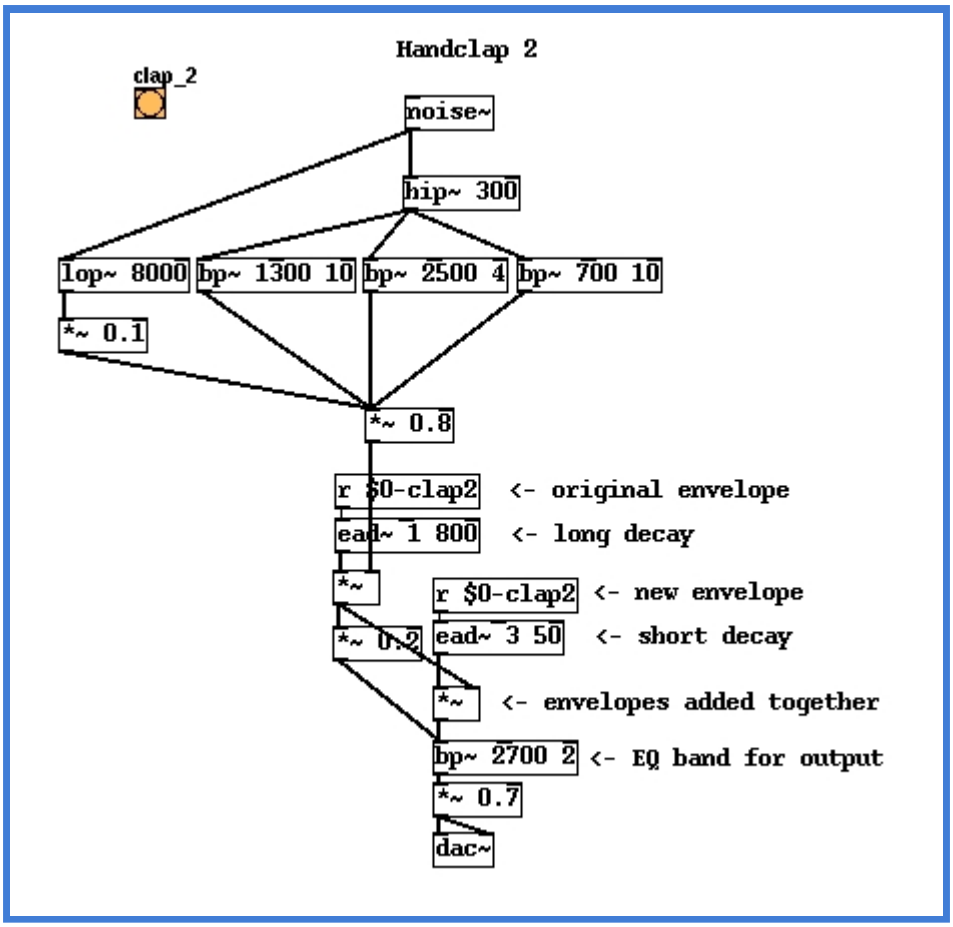
Let's start with a broadband noise source and take just a little of it through a lowpass filter to get rid of the very brightest part. That will give us a little snappyness. The rest we pass through three parallel bandpass units with medium resonance to pick out a strong peak at 2500Hz a few hundred Hz wide and a couple of narrower sub-bands at 1300Hz and 700Hz. That gives about the right texture for a handclap, but all the frequencies are happening at once, at random, so we don't have a way of spacing them in time. That doesn't seem to matter much because a handclap short and we can get a good first try using only a single envelope generator.



Puredata file .pd

Audio file .mp3

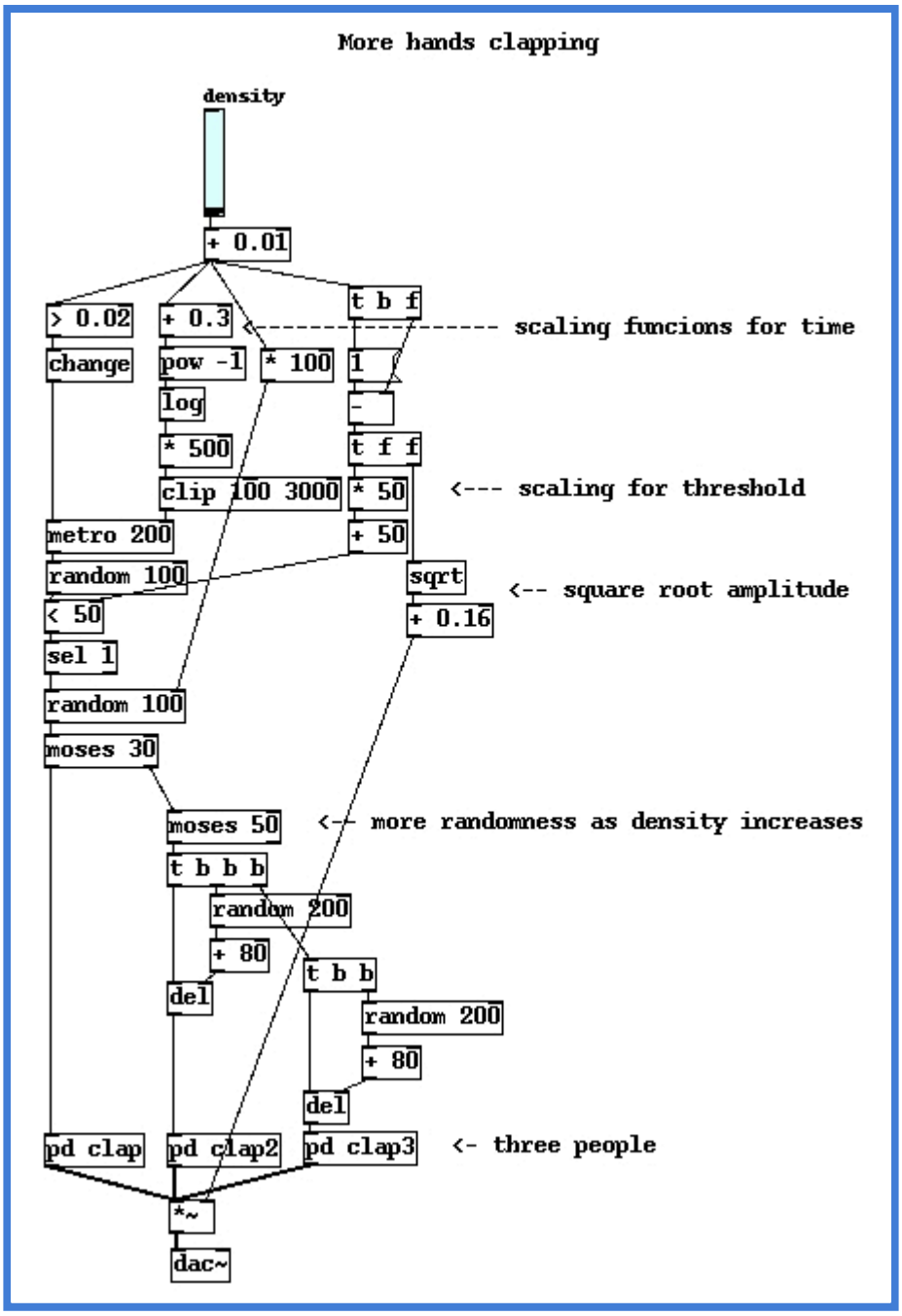
It sounds too loud as it stands. The real part of a clap is only about 50ms and the rest is reverb. If we add another envelope generator with its signal added onto the original slower decaying envelope we get an effect like a short snap with reverb. The first use of this method I know of is the Roland/Boss Dr110 drum machine.



Puredata file .pd

Audio file .mp3

To get a larger ensemble lets take it a step at a time and increase the number of clapping hands to three people. The next patch just copies our last handclap twice and provides some stochastic (meaning to use an element of randomness) control over all three. We control this with a single parameter we denote by *density*. When density is zero the metronome is switched off so no events arise. As density increases the other two clappers join in, and at the same time the pattern moves from a regular periodic beat to a scattered one.



Puredata file .pd

Audio file .mp3

What happens when we try to scale this up is unexpected. It's obvious there isn't enough variation in amplitude and tone to get the right effect. We hit an interesting question too, for real-time applications we need to keep an upper limit on CPU usage, if the claps are derived from a synchronous event chain we can guarantee a limit on how much CPU is used at any point, but it sounds awful (listen to the mp3 example). If we allow each handclap to be its own master and clap at a random rate it sounds much better, but there is the possibility that all the generators might

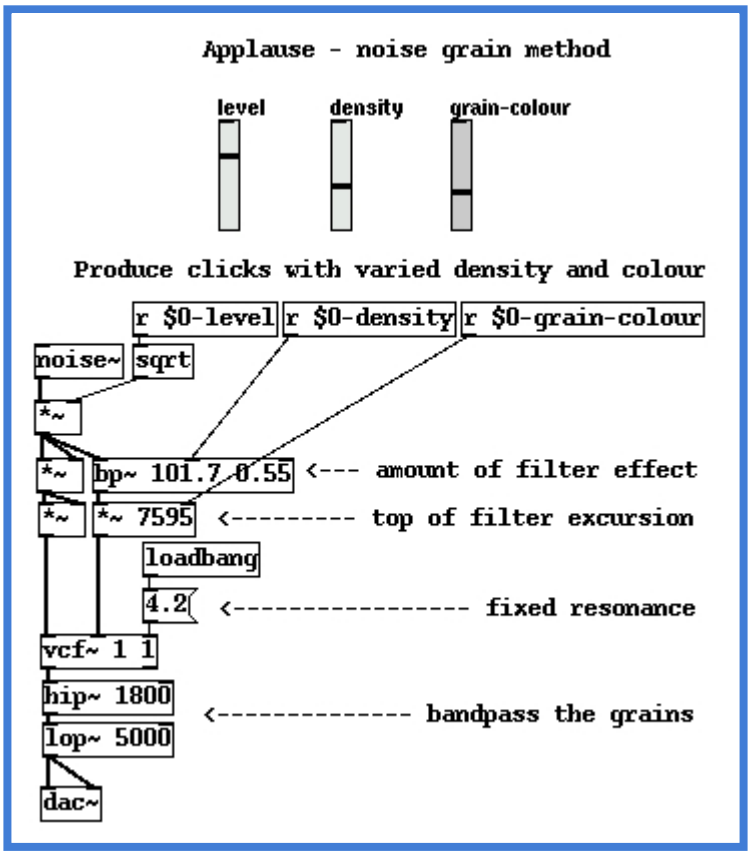
clap at once. Staying with the safer method we could increase the event rate processing to be much faster, and we could give each clap a random tone too, but that just consumes more CPU on processing messages, so we get stuck with scaling this quite early on. Let's move on to the trick of this exercise, granular texture production, and later we will crossfade the individual claps into the texture to see how it blends.

Audio file .mp3

Making a clapping texture

How can we obtain a single source signal that sounds like many of something else in ensemble? We cheat by using a statistical probablility. We want a kind of pattern that has short bursts of single events at the right density and distribution. As usual we start with a white noise generator.

Below is a special case of a grain generator that uses a [vcf~] unit. Because a [vcf~] can control its frequency with an audio signal we can make very short little points of coloured noise at a certain density. These grains are actually short sweeps up and down. Because the sweep envelope is so short we do not hear it, rather the grain sounds like a click with a certain duration, spectral width and center frequency that we have control over.



Puredata file .pd

Audio file .mp3

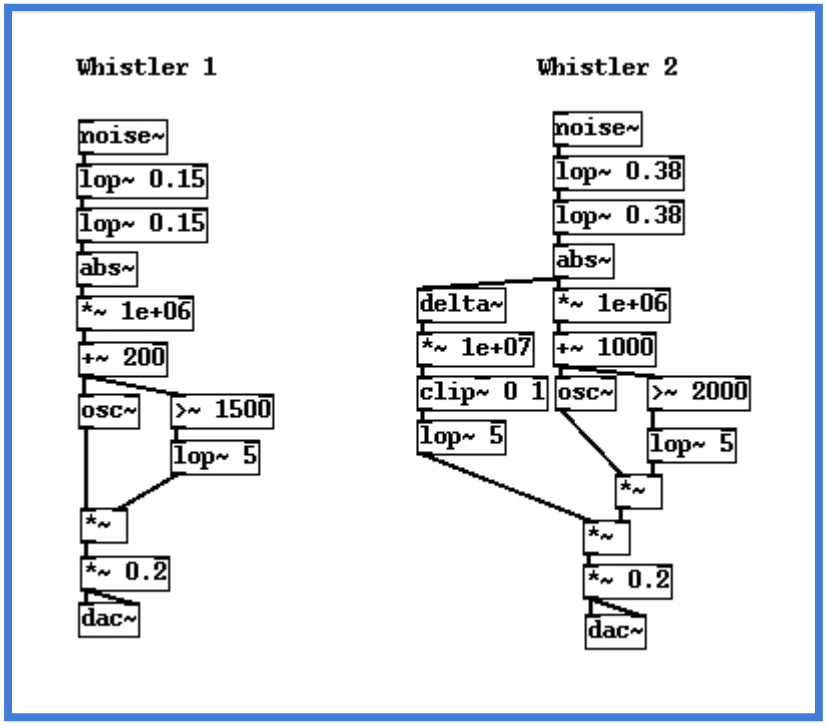
In the audio example you can hear me moving the density and colour around until something like an applause texture emerges. The first [bp~ 100 0.5] bandpass is there to set the density of the clicks, at a mean of about 100 per second. The multiplier block that follows it sets the frequency top for each grain (other higher frequencies get introduced by the modulation process, but that doesn't bother us, we just get an exagerated effect). By varying just density and peak value we obtain a good range of crispy crunchy textures from "paper being crumpled" through "ice and snow" to "wet gravel", "rainfall" and so on... The peculiar distribution of grains that seems to work for applause is centered on about 7kZ. The duration of each grain is determined by the power to which we raise the filter input, in this case a 4th power produces suitably short exitation clicks. Further equalisation is neccessary by non-resonant high and low pass filters at the output stage, but essentialy the clapping generator is five units, three multipliers, and two filters, one with fast frequency modulation.

This patch raises an interesting question. One which explains in a little more detail what I mean by sound (performance) parameter and synthesiser (functional) parameter and their difference. We can control the intensity of the patch amplitude with a multiplier after the [noise~] unit, or at the the patch output as usual. What's the difference? None at all. Not in this case, which is what makes it interesting. We have a parametric equivelence. Putting a amplitude control on the output would be exactly the same effect on the signal as controlling the noise level. Those two parameters share the same identity, they do the same thing, but prodedurally they are very different. If our patch included a nonlinearlity, like a [clip~] or waveshaping function of the noise amplitude the patch would make very different reponses to each when having them tweaked. Compare that to what happens in the next part.

Whistling

The crowd might really like the show, so we have a bit of whistling going on here too. What's happening here? Well each whistler is a little group of four together, two whistling in a slighty different way from the other two. We take some noise and get a very low frequency signal. It needs boosting up a lot after we filtered it so massively, but we get it back into the range of about 0 to a few thousand after taking just the positive magnitude with [abs~]. These countours are to control the whistle frequency and amplitude. A whistle is a near sinewave, so we just use one [osc~] and stay simple. A base offset of a hundered or two stops any silly low whistles and the amplitude is a low passed function of any peaks above 1500-2000Hz. This creates "whoops" of high whistles that rise and fall. The first two voices

basically do the same thing but in slightly different ranges.



Puredata file .pd

Audio file .mp3

Some whistles however just go up and then stop. To get ones that only rise we take the rate of change of the contour and using a [clip~] take only positive going sections. Using that signal as the amplitude control does the trick, we only hear sounds that rise. Lastly, some very excited audience members of the crowd might vibrato or warble their whistles. A quickish LFO on one of the whistles in the final scene makes this sound. Interestingly, here's a case where factoring out the noise base is really not the thing to do. Try it. The whistlers take on a demonic rudeness by all whistling in unison out of tune and slightly out of sync.

Notice now, in the above patch that there is a fundamental difference between what happens if you adjust the noise base level and what happens in the applause texture patch. Reducing it now alters the density of the whistles. As you move it towards zero you still get the odd loud and clear whistle but less of them.

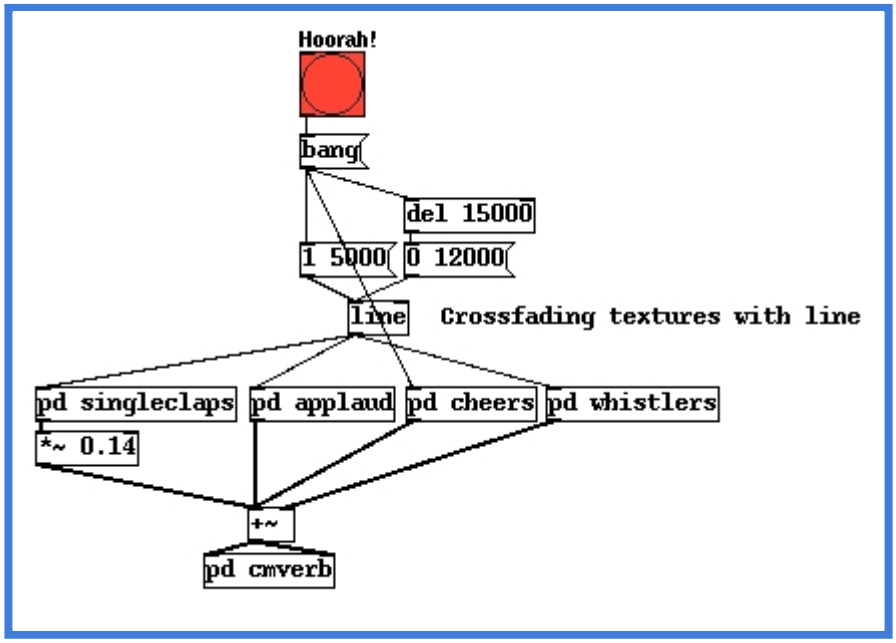
The inlet to this subpatch is in fact a single normalised value. As it happens we want the frequency and intensity of whistles to follow much the same curve. So they are combined. This illustrates the final point about parameterisation in this example, we now have two prodedurally different synthesis parameters which produce important but different effects on a sound, and we **deliberately** combined them into one performance

parameter.

Effects

Dry claps, and applause ensembles are quite hard to recognise without a context and ambience, so we've included one in this patch. In game parameterisation we would inject the sound into the world ambience layer at this point as a "diffuse" or "area" emitter in the appropriate direction(s) from the player. We can't do that here, but we can hear better how applause would really sound by having a reverb help us test the sound.

Putting it all together is simple. An envelope from a line comming in over two seconds and fading down over about fifteen secons makes the whole thing work. To add a bit of extra density I've combined a couple applause generators and gently changed the texture of one of them. The whole lot gets processed by the reverb to bind it together.



Puredata file .pd

Audio file .mp3

Conclusions

As we move from singular events into extents of sound we need to employ statistical methods and create efficient approximations of ensembles. Just adding together lots of individual sounds doesn't work well for a number of reasons. Granular synthesis is a cool way of approximating sounds built from an extent of many smaller sounds happening at once.

Exercise

The build and decay are not quite right in the example. Try to create new fade functions for the whistlers and clapping so that the clapping remains relatively loud until the last moments, the suddenly thins out in density. Can you create larger group sounds, shouting and rioting sounds that would be used for football crowds or battle scenes?

Links

Leevi Petolas thorough study of clapping texture

Next next tutorial

Top tutorials list