INSTITUTE OF SONOLOGY

Royal Conservatoire The Hague

Synthesis

Programming and Music
http://www.bjarni-gunnarsson.net

Synthesis

Synthesizing sound with signals and the transformation of those signals to create musical material.

Techniques discussed:

* *Additive Synthesis*
* *Noise-based Synthesis*
* *Subtractive Synthesis*

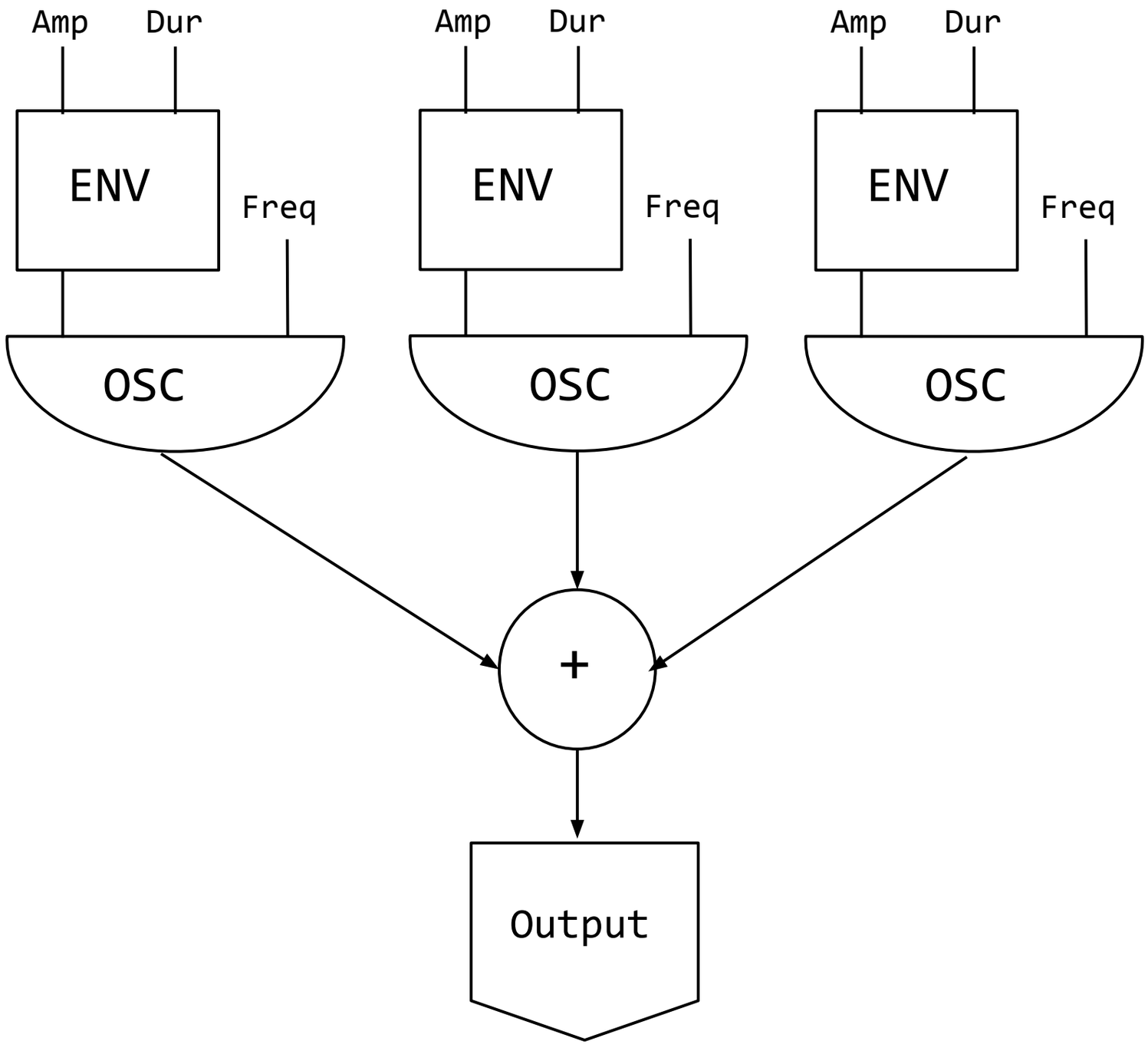
Additive Synthesis

Additive synthesis is a sound synthesis technique that creates timbre by ***adding sine waves*** together.

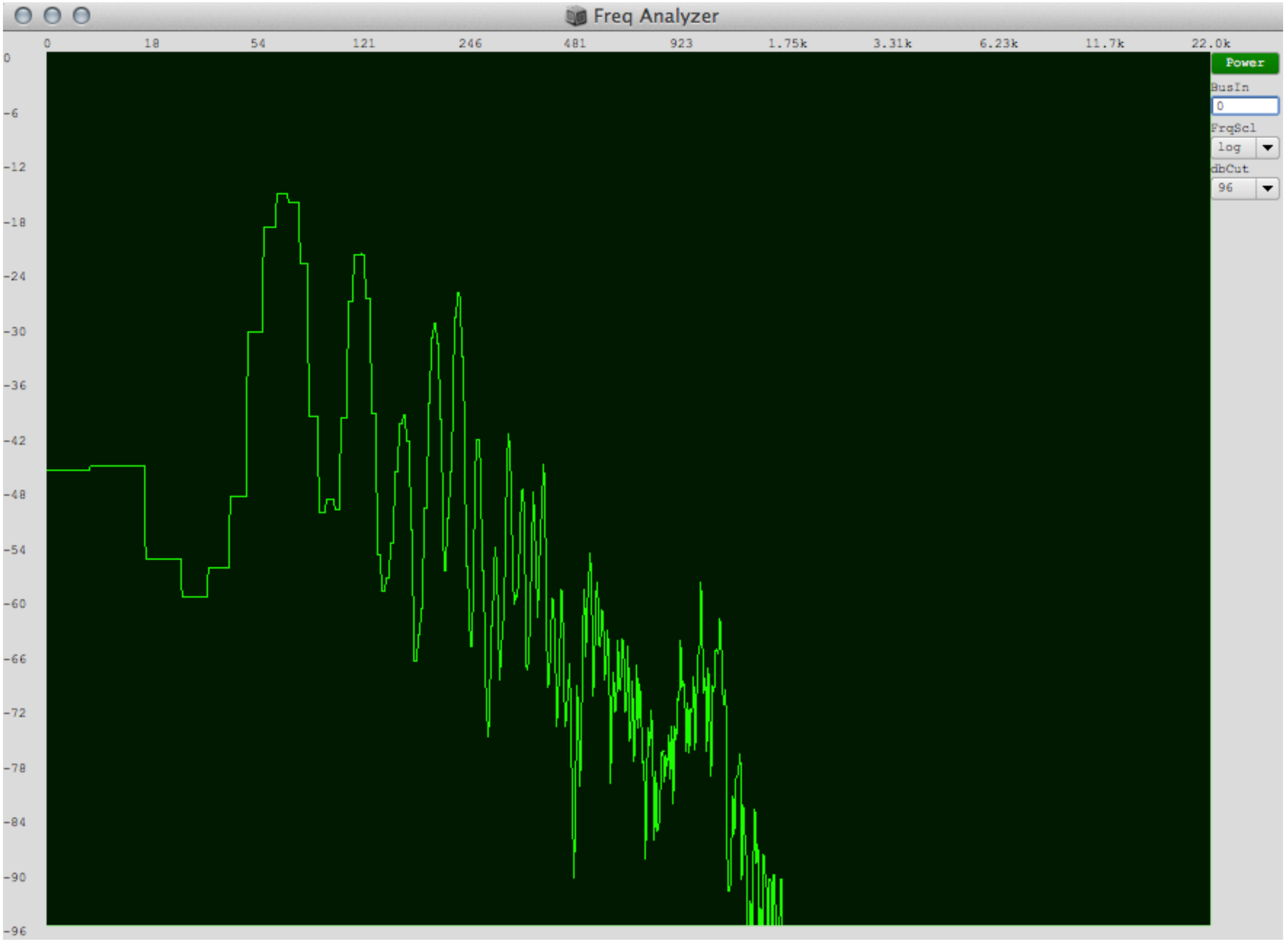
According to Fourier theory any periodic waveform can be broken down into a series of sine waves at different ***frequencies, amplitudes,*** and ***phases***.

The layering of sinusoidal waves and building sounds from individual partials requires many oscillators and large data specification in order to obtain ***complex sounds***.

Additive Synthesis



Frequency Spectrum



Cologne

The Cologne studio in the 1950s made much use of its sine wave oscillator. Additive synthesis was done by summing the partials on ***tape*** after ***recording*** them.

By using the sine wave as a starting point ***serial methods*** could be realized where "*everything to the last element of the note is subjected to serial permutation*" (Eimert, 1955)

This extremely laborious and time-consuming activity resulted in various developments in electronic music and for composing sound.

Noises

The ***color*** of a noise signal (a signal produced by a stochastic process) is generally understood to be some broad characteristic of its ***power spectrum.***

This sense of "color" for noise signals is similar to the concept of ***timbre*** in music.

Noises

Noises are usually named after colors. This started with "***white noise***", a signal whose spectrum has equal power within any equal interval of frequencies. That name was given by analogy with "white light", which was (incorrectly) assumed to have such a "flat" power spectrum over the visible range.

Other color names, like ***"pink", "red", and "blue"*** were then given to noise with other spectral profiles; often (but not always) in reference to the color of light with similar spectra.

Filters

It is common to use filters during synthesis to shape a signal. This works best when the source being filtered has a rich spectrum.

Low Pass – allows all frequencies below the cutoff point to pass through.

High Pass – the opposite of the low pass filter, allows all frequencies above the cutoff point to pass through.

Band Pass – allows a band on frequencies to pass through in the centre, but stops all frequencies outside of this band.

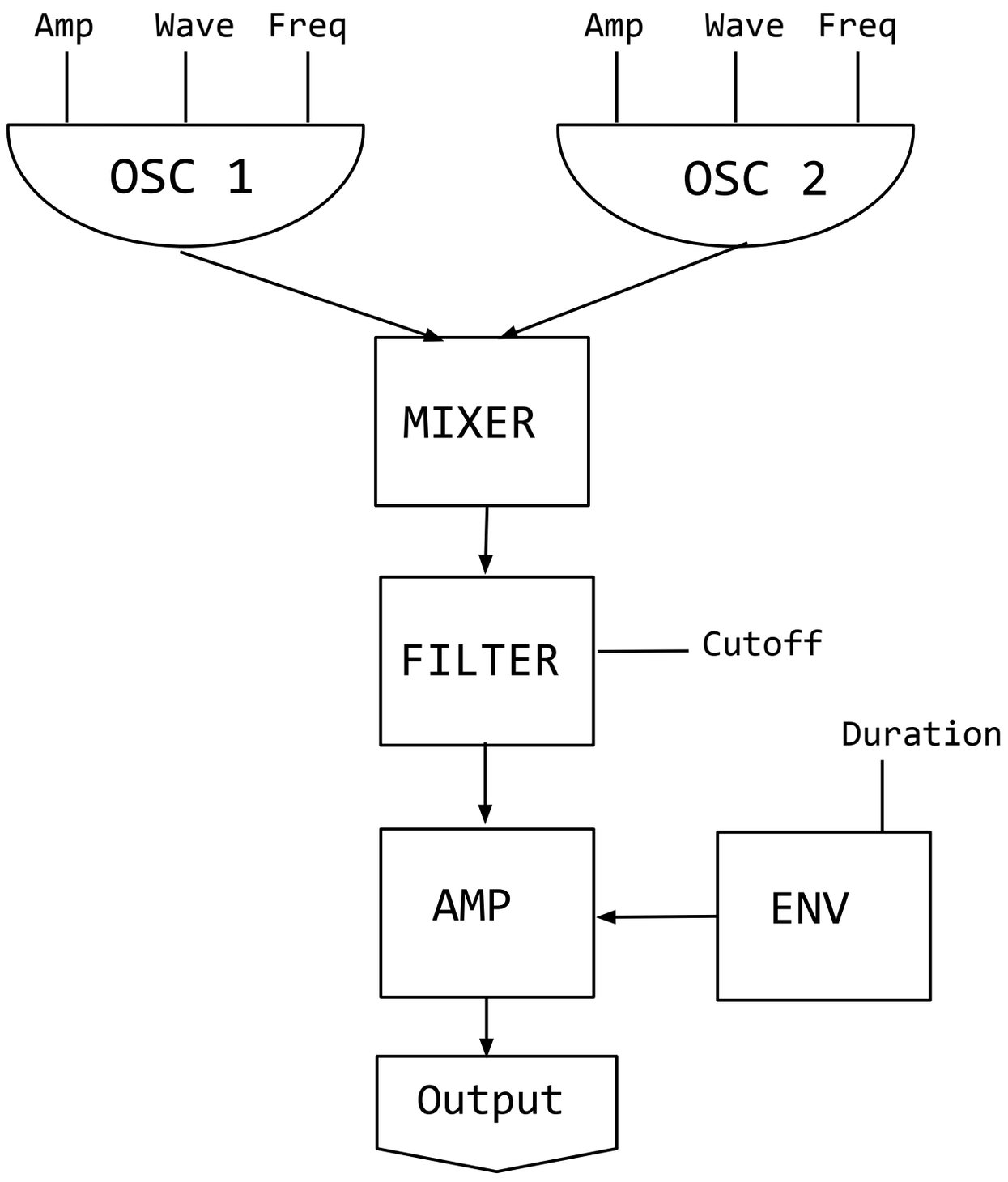
Band Notch/Reject – stops a band of frequencies in the centre from passing through.

Subtractive Synthesis

A method of sound synthesis where ***partials*** of an audio signal (often one rich in harmonics) are ***attenuated*** by a ***filter*** to alter the timbre of the sound.

Can be applied to any source audio signal, the sound most commonly associated with the technique is that of analog synthesizers of the 1960s and 1970s, in which the ***harmonics*** of simple waveforms such as ***sawtooth, pulse*** or ***square*** waves are attenuated with a voltage-controlled ***resonant low-pass filter***.

Subtractive Synthesis



Exercises

```
(
NF(\iop, {|freq=78, mul=1.0, add=0.0|
    var noise = LFNoise1.ar(0.001).range(freq, freq + (freq * 0.1));
    var osc = SinOsc.ar([noise, noise * 1.04, noise * 1.02, noise * 1.08],0,0.2);
    var out = DFM1.ar(osc,freq*4,SinOsc.kr(0.01).range(0.92,1.05),1,0,0.005,0.7);
    HPF.ar(out, 40)
}).play;
)

(
NF(\dsc, {|freq = 1080|
    HPF.ar(
        BBandStop.ar(Saw.ar(LFNoise1.ar([19,12]).range(freq,freq*2), 0.2).excess(
        SinOsc.ar( [freq + 6, freq + 4, freq + 2, freq + 8])),
        LFNoise1.ar([12,14,10]).range(100,900),
        SinOsc.ar(20).range(9,11)
    ), 80)
    ;
}).play;
)
```

```
var <>pindex, <>cindex;

initialize {
    if(pindex.isNil, { pindex = 1000 });
    if(cindex.isNil, { cindex = 2000 });
}

clearProcessSlots {
    pindex = 1000;
    (this.pindex - 1000).do{|i| this[this.pindex+i] = nil; }
}

clearOrInit {|clear=true|
    if(clear == true, { this.clearProcessSlots() }, { this.initialize() });
}

transform {|process, index|
    if(index.isNil && pindex.isNil, {
        this.initialize();
    });

    pindex = pindex + 1;
    this[pindex] = \filter -> process;
}

control {|process, index|
    var i = index;

    if(i.isNil, {
        this.initialize();
        cindex = cindex + 1;
        i = cindex;
    });

    this[i] = \pset -> process;
}
```

```
(
NF(\depfm, {|freqMin=5, freqMax=20, mul=20, add=80, rate=0.5, modFreq=2100, index=0.3, amp=0.2|
    var trig, seq, freq;
    trig = Dust.kr(rate);
    seq = Diwhite(freqMin, freqMax, inf).midicps;
    freq = Demand.kr(trig, 0, seq);
    HPF.ar(PMOsc.ar(LFCub.kr([freq, freq/2, freq/3, freq/4], 0, mul, add),
    LFNoise1.ar(0.3).range(modFreq,modFreq*2), index) * amp, 50)
}).play;
)
```

Exercises

1. Create a sound movement that starts with a sine wave and ends with a noise.

2. Create a cluster of sine wave (10 or more) going from low frequencies to high ones.

3. Create a slow sound movement lasting 30 seconds or more using at least 20 sine waves.

4. Combine different moments of noise types where they can be perceived at the same time during some moments but also separate in others.

5. Add at least two filtered Saw waves together and use a sine wave to control the balance between the two.