Noise Engineering

Loquelic Iteritas

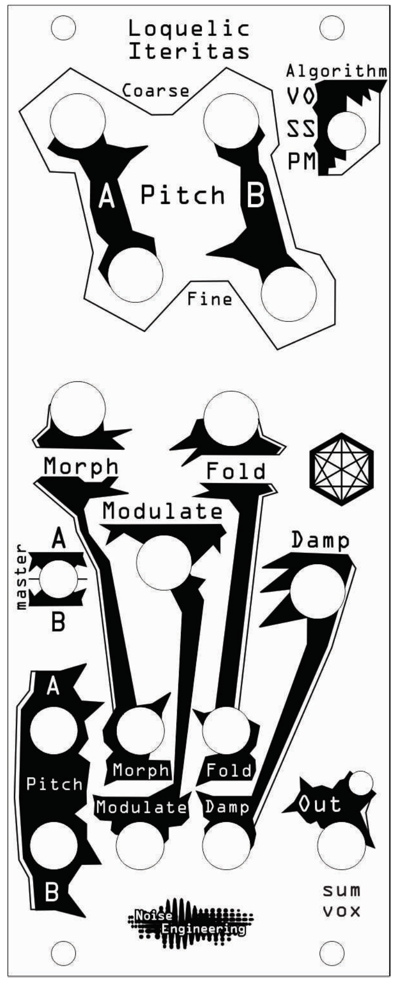
Complex Digital Oscilator

Overview

Type	VCO
Size	10HP Eurorack
Depth	1 Inch
Power	2x8 Eurorack
+12 mA	150 / 80
-12 mA	5
+5 mA	0 / 90 (optional)

"I could kill someone with that"
-- DJ Surgeon

"This thing sounds fucking amazing
lots of stuff I've never heard before"
-- Surachai



Loquelic Iteritas is a digital VCO with interpretations of three classic synthesis algorithms involving dual pitch control. It creates a huge variety of sounds parameterized by four tone and two pitch controls.

Noise Engineering

Loquelic Iteritas

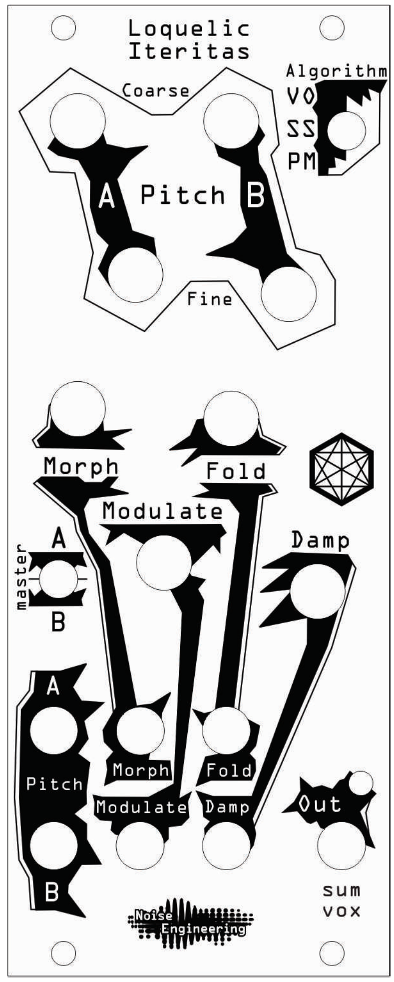
Complex Digital Oscilator

Patch Tutorial

The easiest way to get to know Loquelic Iteritas is to turn the knobs and listen. Connect the output to your mixer and start twiddling.

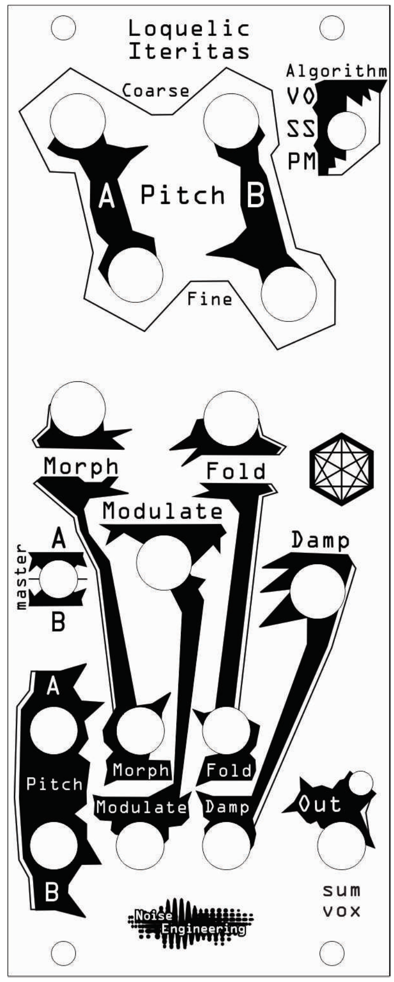
Loquelic Iteritas is about continuous tone control. Hook any LFO up to any of the four tone control inputs (Morph, Fold, Modulate, Damp).

Other interesting effects can be created by controlling the pitches independently (by default the 1v/8va inputs are normaled to each other). For instance, using a Tonnetz Sequent to produce musical intervals produces interesting results.



Noise Engineering
Loquelic Iteritas
Complex Digital Oscilator

Interface



Pitch A
The pitch of oscillator A can be controlled by the 1v/8va input and offset by it's coarse and fine knobs. The pitch inputs are cross normaled.

Pitch B
The pitch of oscillator B can be controlled by the 1v/8va input and offset by it's coarse and fine knobs. The pitch inputs are cross normaled.

Damp
is a tone control. Consult the following pages detailing each mode to find the behavior of this knob in the specific mode.

Mod
is a tone control. In all modes it controls phase modulation between the two pitch oscillators.

Fold
is a tone control. In all modes it controls the threshold of the wavefolding.

Morph
is a tone control. In all modes it controls the waveform of the oscillator continually varying between sin, triangle and saw.

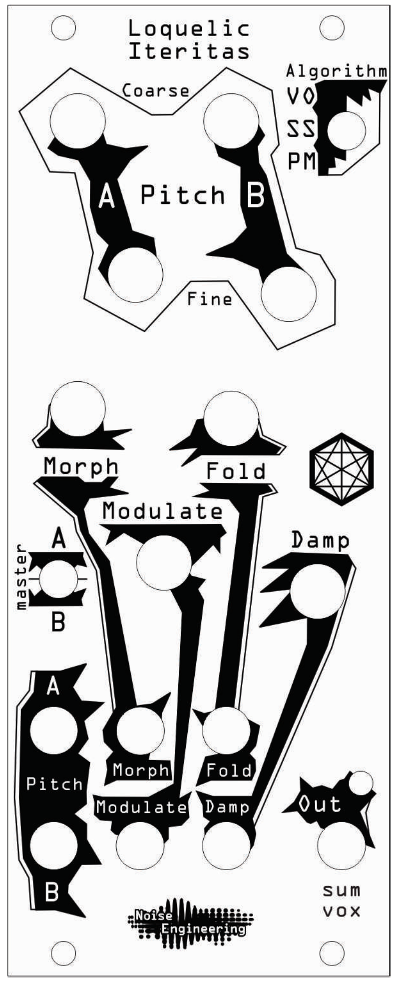
Algorithm
selects which algorithm is used. These are detailed on the following pages.

Master
controls the sync of the oscillators. When in the middle position both oscillators are free running. When A is selected oscillator B will sync to oscillator A. when B is selected A will sync to B.

Out
Out is the AC coupled audio output.

Noise Engineering
Loquelic Iteritas
Complex Digital Oscilator

Algorithm: VO



The VO algorithm is roughly based off of the VOSIM algorithm which I discovered while reading Curtis Roads's epic *Microsounds*. This algorithm amplitude modulates a carrier by an exponential to create a more complex harmonic structure. The simplest carrier is a sinusoid which produces a spectrum with a Gaussian distribution centered on the carrier. More complicated waveforms produce Gaussians around each harmonic, producing spectra similar to comb filtered noise.

Pitch A is the fundamental frequency of the carrier. Pitch B is the retrigger frequency of the exponential decay.

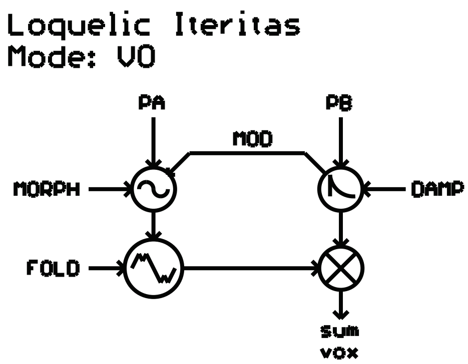
Interface

MORPH - changes the waveform of oscillator A
DAMP - sets the decay constant on oscillator B relative to its period
MOD - phase modulates oscillator A by oscillator B
FOLD - sets the wave fold threshold on the final wave folder

References

Kaegi, Werner, and Stan Tempelaars. "Vosim-a new sound synthesis system." Journal of the Audio Engineering Society 26.6 (1978): 418-425.

Roads, Curtis. Microsound. MIT press, 2004.

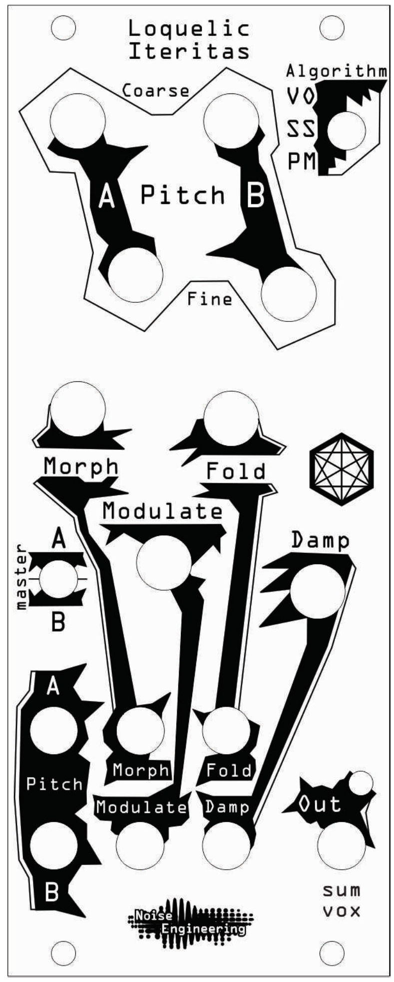


Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Algorithm: SS



Algorithm SS is a highly modified version of summation synthesis originally developed by James Moorer. The premise comes from a simple mathematical equality between an infinite harmonic series and a relatively easy to compute expression.

Original equation:

$$\frac{\sin(\Theta) - a\sin(\Theta - \beta)}{1 + a^2 - 2a\cos(\beta)} = \sum_{x=0}^{\infty} a^x \sin(\theta + x\beta)$$

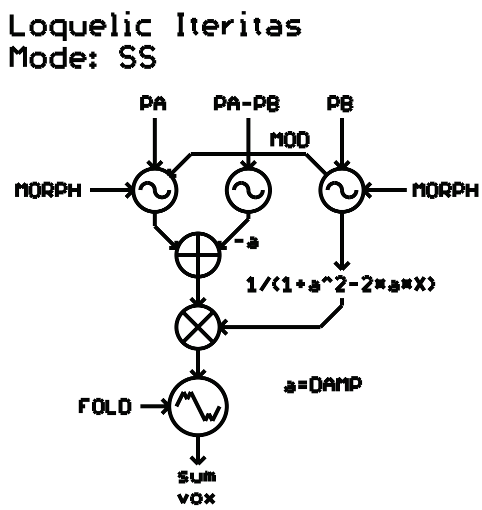
This equation allows a wide variety of musical spectra to be produced by only two parameters. Loquelic Iteritas generalizes the sinusoidal terms into multi-waveform oscillators: two of these track the two input pitches while the third tracks the difference of the two pitches and adds a wave folder for more harmonics. In the equation oscillator A is the left sinusoidal term in the numerator. Oscillator B is the sinusoidal term in the denominator.

Modified Equation:

$$\frac{\sin(w_A t) - a\sin(w_A t - w_B t)}{1 + a^2 - 2a\cos(w_B t)} = \sum_{x=0}^{\infty} a^x \sin(w_A t + x w_B t)$$

Interface

MORPH - changes the waveform of all oscillators
DAMP - sets the a parameter in the equality. This controls the generated spectra with higher values producing higher power harmonics.
MOD - phase modulates oscillator A by oscillator B
FOLD - sets the wave-fold threshold on the final wave folder



References

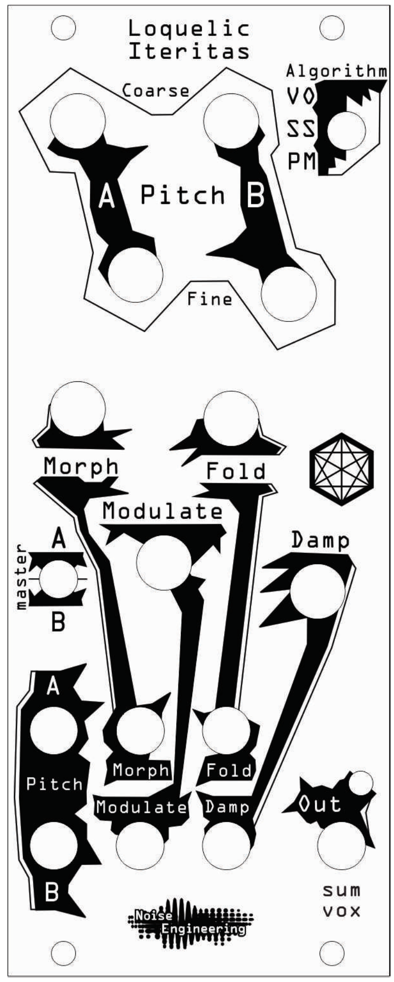
Moorer, James A. "The synthesis of complex audio spectra by means of discrete summation formulas." Journal of the Audio Engineering Society 24.9 (1976): 717-727.

Jolley, Leonard Benjamin William, ed. Summation of series. Courier Corporation, 2012.

Noise Engineering
Loquelic Iteritas
Complex Digital Oscilator

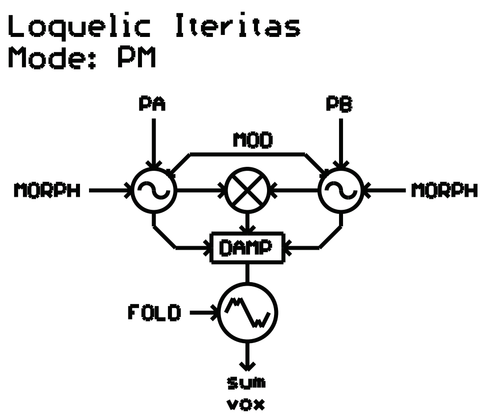
Algorithm: PM

The PM algorithm is a naive time-domain two-oscillator phase-modulation implementation that combines both oscillators with amplitude modulation.



Interface

MORPH - changes the waveform of both oscillators
DAMP - blends between oscillator A and B through their product (AM)
MOD - phase modulates the oscillators by each other
FOLD - sets the wave-fold threshold on the final wave folder



Noise Engineering

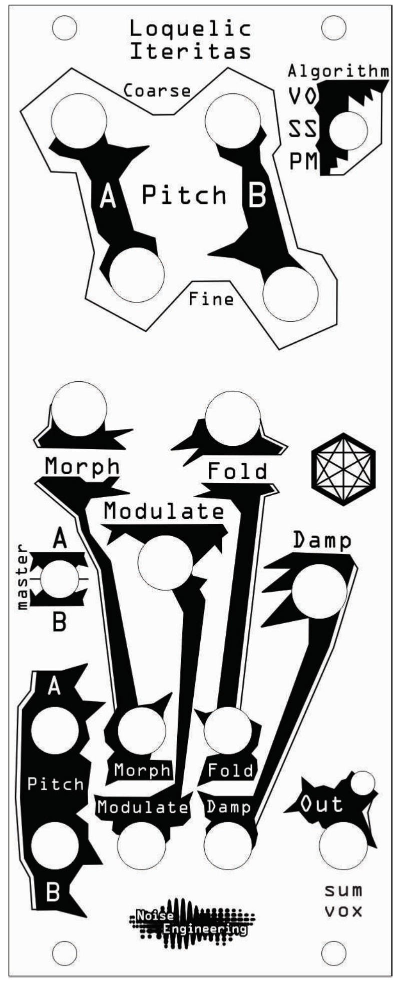
Loquelic Iteritas

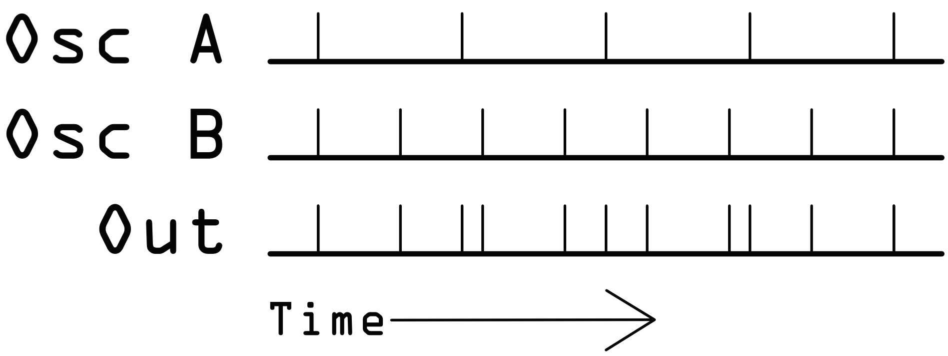
Complex Digital Oscilator

Sample Rate

Loquelic Iteritas uses a unique multisampling technique to make aliasing more musical. By choosing a particular sample rate for a waveform that has a harmonic structure (all overtones are integer multiples of the fundamental) the alias power can be moved into frequencies that are also multiples of the fundamental and therefore more musical.

This gets complicated when synthesizing two oscillators at different pitches but using the same DAC. The compromise that Loquelic Iteritas makes is to give up the notion of a fixed sample rate and compute a time delay between samples based on both oscillators. For the single oscillator case, this delay is based entirely on pitch. If this delay is computed based on each oscillator's pitch, both sample rates can be interleaved by checking which oscillator's delay will be up first. This oscillator is then updated to its next timestep and an output value is computed based on both oscillator's output state. This makes no guarantees about exactly where the aliasing goes. It is an attempt to make the aliasing related in some way to the fundamental pitch.





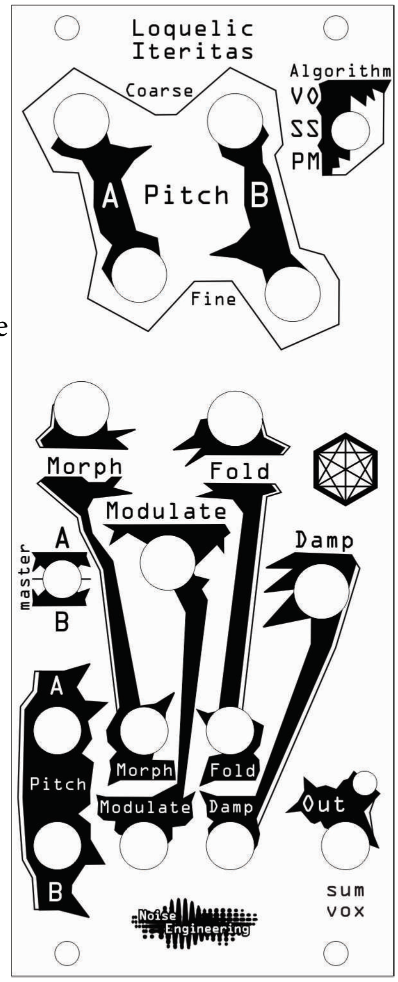
Two independent sample rates combine to form one irregular sample rate. Sample rate is not a constant.

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Design Notes



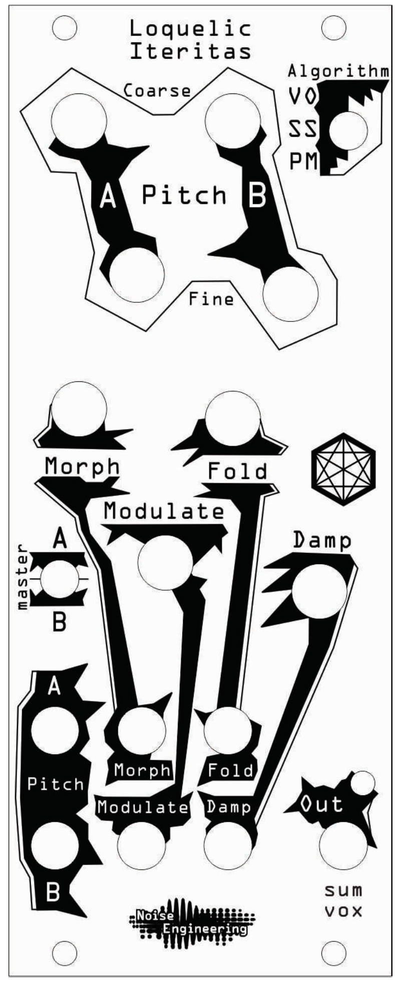
Loquelic Iteritas has been in development for over two years. It was started at the same time as Basimilus Iteritas but has taken much longer to mature. Originally it was just a simple implementation based on VOSIM but I soon realized I could pack a lot more punch in this form factor and found two additional algorithms. Loquelic Iteritas was designed to be a functional oscillator for sound designers as well as for musicians. I wanted to maximize the possible sound space given the input controls going from simple calm sounds to extreme, even broken, sounds. The priority of tonal variance led to some sacrifices on the musical side such as the total pitch range.

The algorithms used are quite simple and are intentionally left naive as they often include interesting rough spots. For example, PM mode has a nasty half-sample-rate self oscillation under high modulation indexes that, when combined with the irregular sample rate, produces interesting, if quite harsh, results.

Noise Engineering
Loquelic Iteritas
Complex Digital Oscilator

Code

For reference I have included the core synthesis code for each algorithm. I am constantly amazed at how much sound variety such simple algorithms can produce and hope that others will appreciate their simplistic beauty. Note: code superfluous to the core algorithm has been removed.



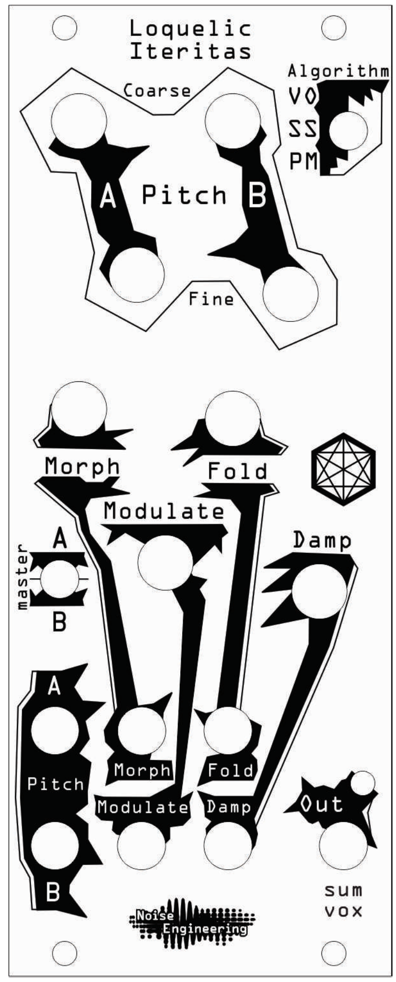
Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Code: VO

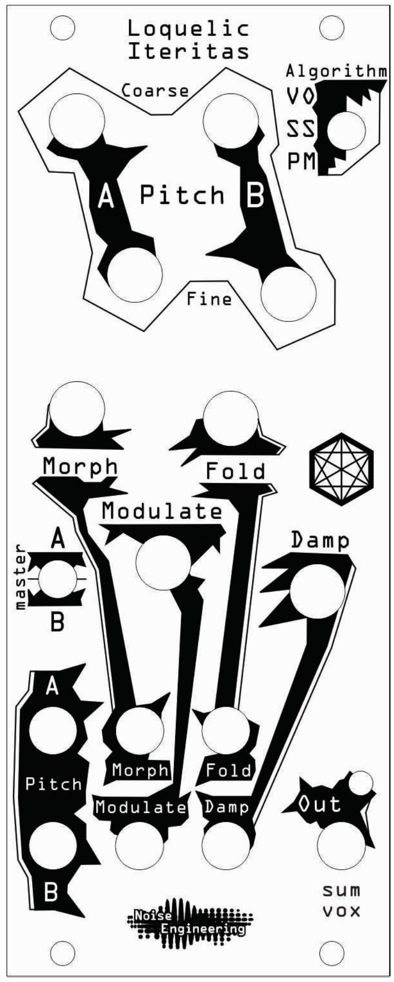
```
unsigned LI_FrameVO()
{
    int delay;

    if((state.voOsc.delay - state.voR1) < (state.voEnv.delay - state.voR2))
    {
        if(state.voOsc.sync && state.current.syncSw == LI_SYNC_B)
        {
            NeAttackDecayReset(state.voEnv);
        }

        state.voOutC = NeMoscSample(state.voOsc, state.morph, state.voMod);
        delay = state.voOsc.delay - state.voR1;
        if(delay < 0) delay = 0;
        state.voR1 = 0;
        state.voR2 += delay;
    }
    else
    {
        state.voOutE = NeAttackDecayOscSample(state.voEnv);
        state.voMod = fix24_mul(state.voModAmt, 2 * (state.voOutE - FIX24_HALF));

        if(state.voEnv.reset && state.current.syncSw == LI_SYNC_A)
        {
            NeMoscReset(state.voOsc);
        }

        delay = state.voEnv.delay -  state.voR2;
        if(delay < 0) delay = 0;
        state.voR2 = 0;
        state.voR1 += delay;
    }

    fix24 out = 0;
    out = NeFoldSample(state.fold, state.voOutC);
    out = fix24_mul(state.voOutE, out);
    out = fix24_mul(state.voMComp,out);
    out = fix24_soft_clip_poly(out);
    return fix24_to_u16_audio_delay(out, delay);
}
```



Noise Engineering
Loquelic Iteritas
Complex Digital Oscilator

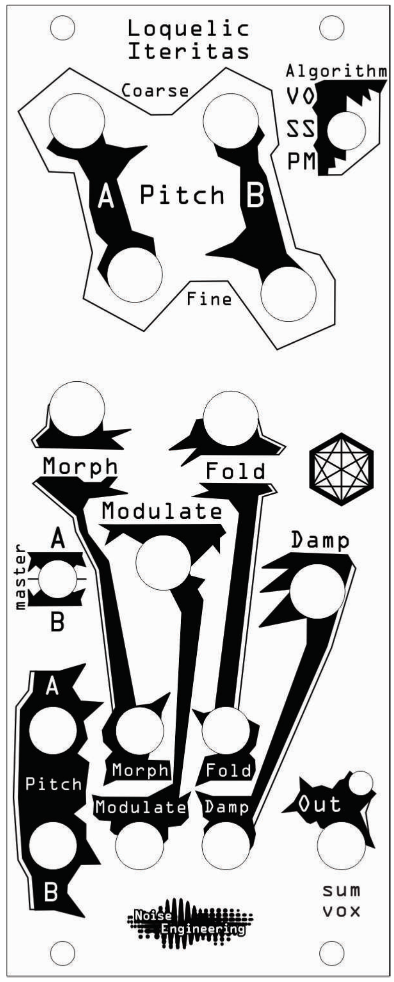
Code: DS

```
unsigned LI_FrameDS()
{
    fix24 out = 0;
    int delay = 0;

    state.dsPb = NextC( state.dsPc, state.dsPm, state.dsPb);

    int dc = state.dsOscC.delay - state.dsRc;
    int dm = state.dsOscM.delay - state.dsRm;
    int db = state.dsOscB.delay - state.dsRb;

    if(dc <= dm && dc <= db) //dc is next
    {
        fix24 phaseC = fix24_mul(state.dsOutM, state.dsMod);
        state.dsOutC = NeMoscSample(state.dsOscC, state.morph, phaseC);
        delay = dc;
        if(delay < 0) delay = 0;
        state.dsRc = -delay;
    }
    if(dm <= dc && dm <= db) //dm is next
    {
        fix24 phaseM = FIX24_QUARTER + state.morph;
        state.dsOutM = NeMoscSample(state.dsOscM, state.morph, phaseM);
        delay = dm;
        if(delay < 0) delay = 0;
        state.dsRm = -delay;
    }
    if(db <= dm && db <= dc) //db is next
    {
        state.dsOutB = NeMoscSample(state.dsOscB, state.morph);
        delay = db;
        if(delay < 0) delay = 0;
        state.dsRb = -delay;
    }

    if(state.current.syncSw == LI_SYNC_A)
    {
        if(state.dsOscM.sync) NeMoscReset(state.dsOscC);
    }
    else if(state.current.syncSw == LI_SYNC_B)
    {
        if(state.dsOscC.sync) NeMoscReset(state.dsOscM);
    }

    state.dsRc += delay;
    state.dsRm += delay;
    state.dsRb += delay;

    fix24 a = state.dsA;
    fix24 a2 = fix24_mul(a, a);
    fix24 n = state.dsOutC - fix24_mul(a, state.dsOutB);
    fix24 d = FIX24_128TH + FIX24_ONE + a2 - 2 * fix24_mul(a, state.dsOutM);
    out = fix24_mul(FIX24_3RD,fix24_div(n, d));
    out = fix24_mul(state.morphScale, out);
    out = fix24_soft_clip_poly(out);
    out = NeFoldSample(state.fold, out);
    return fix24_to_u16_audio_delay(out, 2 * delay);
}
```



Noise Engineering
Loquelic Iteritas
Complex Digital Oscilator

Code: PM




```
unsigned LI_FramePM()
{
    fix24 out = 0;
    int delay = 0;

    int updateDelay1 = state.pmOsc1.delay - state.pmR1;
    int updateDelay2 = state.pmOsc2.delay - state.pmR2;

    if(updateDelay1 <= updateDelay2) //update whichever osc is due next
    {
        state.pmOut1 = NeMoscSample(state.pmOsc1, state.morph, state.pmPhase1);
        delay = updateDelay1;
        if(delay < 0) { delay = 0; }
        state.pmR1 = 0;
        state.pmR2 += delay;
    }
    else
    {
        state.pmOut2 = NeMoscSample(state.pmOsc2, state.morph, state.pmPhase2);
        delay = updateDelay2;
        if(delay < 0) { delay = 0; }
        state.pmR1 += delay;
        state.pmR2 = 0;
    }

    if(state.current.syncSw == LI_SYNC_A && state.pmOsc2.sync)
    {
        NeMoscReset(state.pmOsc1);
    }
    else if(state.current.syncSw == LI_SYNC_B && state.pmOsc1.sync)
    {
        NeMoscReset(state.pmOsc2);
    }

    state.pmPhase1 = (7 * state.pmPhase1 + fix24_mul(state.pmMod1, state.pmOut2))>>3;
    state.pmPhase2 = (7 * state.pmPhase2 + fix24_mul(state.pmMod2, state.pmOut1))>>3;

    fix24 am1 = fix24_mul(state.pmOut1, state.pmAM1);
    fix24 am2 = fix24_mul(state.pmOut2, state.pmAM2);
    fix24 am3 = fix24_mul(am1,am2);
    out = am1 + am2 + am3;
    out = fix24_soft_clip_poly(out);
    out = NeFoldSample(state.fold, out);
    return fix24_to_u16_audio_delay(out, delay);
}
```

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Special Thanks

Kris Kaiser
Shawn Jimmerson
Cyrus Makarechian
William Mathewson
Mickey Bakas
Tyler Thompson
Alex Anderson