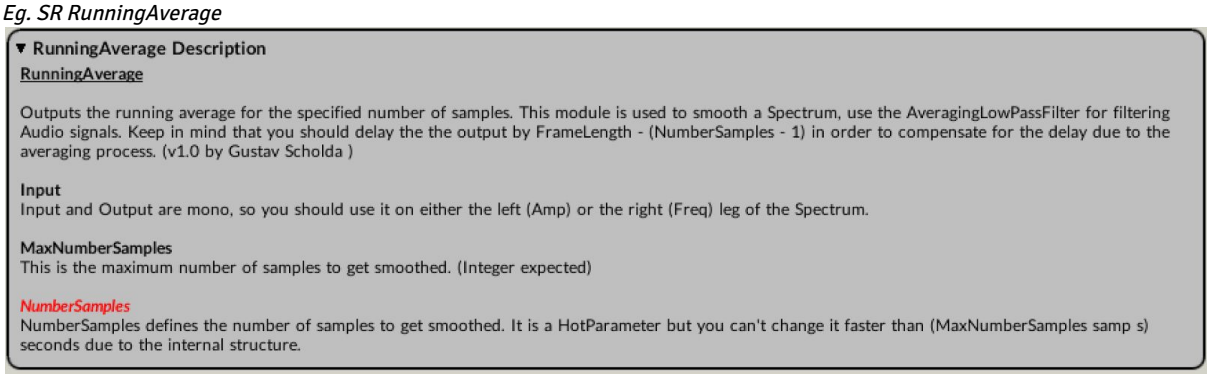
NEVERENGINE LABS

Spectral SR Volume One Examples

Spectral SR Volume One is a library of 33 prototype Classes designed for working with sample accurate sequential frame signals in Kyma Seven, in particular those generated by spectral analysis.

These Classes will eventually form part of a complete programming system we are developing for Kyma 7. Called **WireFrames**[1] , it is designed for highly optimised real time modifications of sequential frame data, such as spectral analysis data. 'Spectral SR Volume One Examples.kym' documented here, is a collection of just a few sound design examples included with the SPC SR Volume One pack originally published in 2015. The Class prototypes they are built with are also internally documented in their parameter descriptions.

Eg. SR RunningAverage



Some Basic Concepts

- A **Frame** is a collection of a fixed number of values streamed sequentially.
- The number of the values in one frame is referred to as its **FrameLength**.
- Kyma reads through a Frame at sample rate "*in the wire*"[2]
- A Frame of sample values in Kyma is analogous to a frame of pixels in video
- Every value in a Frame can be shaped separately by a sample rate **FrameShaper** signal

Spectral SR Volume One.kym

SplitSpectrum Basic (no effect)

Very basic example of two convenience modules, which you will use a lot to split the 'wires' carrying the frame stream of Amplitudes and the frame stream of Frequencies, for separate processing 'in the wire'. In this example, a spectral analysis is simply split and recombined, with no alteration. Use it as a template.

Formant Preserving Pitch and Formant

An advanced example of **'WireFrames'** programming. It wires up a number of SPC SR modules, before any resynthesis occurs, to do some quite complex spectral processing (only

[1] **'WireFrames'** is currently under development - expect full launch in Autumn 2016
[2] a term coined by Pete Johnston in his KISS2011 lecture 'The Wire Between'

frequency and amps, no time stretching manipulations). Uses ExtractFormants filter which works with a threshold to filter out the peaks of the spectral envelope, also known as Formants. Notice how the output of the ExtractFormants is subtracted from the original signal, leaving only the cancelling the formants from the orignal signal, the inverse result of ExtractFormants (effectively ExtractNonFormants). Notice how simple Constants are used as **FrameShapers** to control the formant shift ammount equally across the whole frame.

RealTimeStretch v1.2 w/TapeMode

The RealTimeStretch lets you perform timestretching in realtime. You can also do timecompression but obviously you need to do some stretching before (unless SSC invents a Time Machine!) This example uses the Product1 module to shift the frequencies frame data, in relation to the !Stretch parameter. Its an experiment - once the natural linear relationship between rate change and pitch change is decoupled, what kind of arbitrary relationships can we design? When we slow down an analogue medium, such as tape, the frequencies of the recording 'slow down' too. Here we do that, but as its a time stretch, why not experiment with non-linear curvatures on the frequency change?

RealTimeStretch + Formant Shifter (revised)

Introduces the Formant Shifter Class, which should be used on the Amps frame stream for best results. Here we have a clear example of the **FrameShaper** concept in 'WireFrames', which is present throughout the SPC SR modules.

- The concept of the **FrameShaper**, is that of a sample rate signal, usually an Oscillator or function generator, which ideally should be in sync with the frame rate. In the WireFrames system, frame rate is equivalent to the **FrameLength** in samples. This input shapes by multiplication usually (think along the lines of a grain envelope) and effectively shapes each individual value in the stream at frame rate.

The formant shifter has a **FrameShaper** input, characterised by only accepting a Kyma sound as input, not Capytalk. In this case it is a sine wave oscillator set to the **FrameLength** frequency (256 samp) and also scaled by a hotvalue !**FormantModRate**. This means that the frame array is modulated by this oscillator shape (which itself can have its phase modulated). The FrameShaper's signal is also shifted and offset by a **ScaleAndOffset** module. This kind of programming demonstrates how complex modulations of many parameters (in this case 256) is achievable with simple modular thinking, using the 'WireFrames' system.

ExtractFormants w/ GranularReverb

This is a straight-forward usage of the Extract Formant class. Both legs (amps & freqs) are needed to make the Extract Formant work. The amps are smoothed with a constant value 0f 0.98 which makes the whole thing quite smeared. After resynthesis we fed it into a granular reverb to smear it even more. !**Density** and !**GrainDur** of the granular reverb are controlled in a 2D widget for intuitive use.

BlurAverage

Using a **RunningAverage** on the amps we are blurring the spectral shape. You can think of it as blurring the border between tracks so each track kind of spills over neighbouring tracks.

Complete Derange

The **PartialDerange** is like a Vocoder Patchbay where you can mix up the bands (e.g. connecting Band 1's output to Band 2's input). Positive values in the **FrameShaper** input will interpolate to the value of the higher partials while negative values will interpolate to the lower partials. The **FractalNoise** controls the Deranging and its persistence is controlled using a random walk which is also dependent on the amount of deranging:

```
(((1 bpm: 1000) randomWalkStartingFrom: 0 stepSize: (1
ramp:30/!DerangeAmmount s) reset: 0) smooth: (2000 bpm s)) *
!DerangeAmmount * 0.1
```

Dual running average (leave running a long time!)

Both legs (amps & freqs) are processed using a **RunningAverage**. The RunningAverage blurs the borders between tracks so each track kind of spills over neighbouring tracks.

Holt-Winters Amp Smoothing / Exponential Freq Smoothing

Two different types of smoothing are used here: The **ExponentialSmoother** applies a straight-forward smoothing effect, preventing the freqs to move or change fast. The Holt-Winters is a more complex type of smoothing which can get into self-oscillation with low **SmoothAmpTend** settings.

Transient Preserving Stretch and Smear (with frame blur strategy)

> *@Cristian:* You should write this one, I don't know what the hell is going on XD
> *@Gustav:* OK. Well I can't promise I will make it any more clear.... This is an experimental design which tries to blur the resynthesis of an elaborate smearing approach. The smearing of the Frames is exagerrated by the FrameMax analysis. Then the resynthesis occurs, and after that a sample rate interpolator tries to rapidly cross fade between the straight resynth and a jittery delayed version. I think I was going for some kind of reverse engineering of the famous Ircam Labs TS spectral stretch.

Simulated Annealing / Transient Preserving Smear / Frame Blur

SimulatedAnnealing is like a mutant morph, it tries to build the spectrum in its input using values it gets from the comparison input. The results are most often quite noisy and unpredictable. After Resynthesis it goes through a little jitter network that interpolates between the previous and the current frame.

SpectralBlur and Formant/Pitch Shift

After applying a pitch and formant shift the amps & freqs both get blurred out. The blur takes snapshots of the spectral signal and interpolates between them. In a way it's like downsampling (or downframing?) the spectral signal and interpolating between the snapshots.

FormantShift & PitchShift

This is a straight-forward implementation of Pitch & Formant Shifting. The constants can be replaced by frame based control data to apply a **FrameShape** and shift the partials individually.

Random Derange

Here the **FrameShape** input of the **PartialDerange** class is **HotPink** Noise. You can think of this as a vocoder patchbay where you swap around Band inputs and outputs randomly at a certain frequency.

Document sprinted in Vienna, 28th June 2016 by Cristian Vogel and Gustav Scholda