

## Granular Synthesis

A grain is a very short segment of sound, usually shaped with a particular envelope. Gabor (in 1947) proposed that any signals could be conceived as the combination of these short grains.

Usually, durations between 10 and 60 ms are used. Grains can often be featureless, on their own, but create complex morphologies when accumulated.

Processing and synthesis techniques which manipulate very short fragments of sound can be described as *granular*.

## Grain parameters

- Grain size: length (in secs or msecs)
- Grain density: number of grains per unit time (sec)
- Grain envelope (attack, decay): also known as *local* envelope, the amplitude shape of each grain
- Grain pitch: this is either the fundamental freq of each grain, or the playback 'speed' (in case of sampled-sound grains).

# Granular Synthesis

- **Synchronous:** all grain streams are synchronous, produced at a certain rate. It can be used to generate pitched material.
- **Asynchronous:** grain streams are not synchronised, individual grains can have varying start times. The overall result is an ‘statistical’ event, where the general tendency, shape and characteristic of the sound is known, but the behaviour of an individual grain is undefined.

# Asynchronous Granular Synthesis

In this technique, ‘clouds’ of grains are generated. Grains are generated both from sampled sound or standard waveforms.

When using sampled-sound grains, this technique is very similar to granular processing (brassage, etc...) . The main parameters relative to those processes also apply here: density, grain size and pitch, read pointer speed, etc..

Usually this technique is implemented using tables of stored waveforms (sampled sound or standard shapes). Sometimes the grain envelope is also stored in a function table.

# Implementing granular synthesis

Granular synthesis can be implemented in several different ways:

- Generating small sound events synthesised using simple envelope+oscillator wavetable instruments. The grain size, duration and density will be controlled by the way events are organised in a 'score'.
- Constructing an instrument that itself generates streams of grains, using amplitude modulation to envelope-shape continuous sounds into grains.
- Using a grain-generating opcode, such as `grain`, `granule`, `FOF`, `FOF2` and `FOG`

## Granular synthesis UGs

**Granule:** this UG generates a user-defined number of **parallel grain streams**. Grains are generated from sound stored in a function table.

*Grain density:* determined by the number of parallel streams.

*Readout pointer speed:* controlled by a readout *ratio* value. This can be used for timescale modifications. The start readout position can also be randomic.

*Grain pitch, size and generation rate:*

can be either deterministic or randomic. When grains are generated at a constant rate, the synthesis is *quasi-synchronous*, otherwise the result is *asynchronous*.

## Grain

**grain:** is a simpler (and faster) grain generator, with the usual controls of density, pitch, amplitude and grain size.

The readout pointer can be set to read from the beginning of the table or from random positions along the table (the default). There is no 'speed' control for it, so that timescale modifications are not possible.

This generator can generate synchronous, quasi-synchronous or asynchronous grain streams, depending on how the grain density is controlled (deterministically or randomic).

## **FOF synthesis**

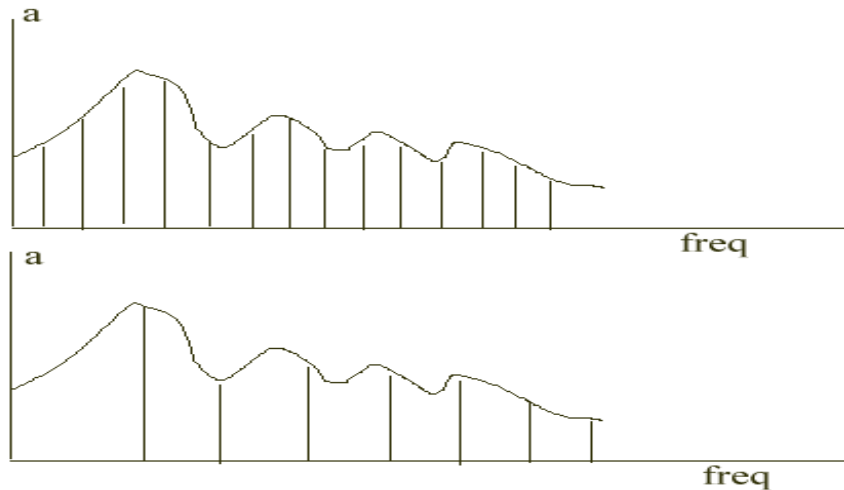
FOF (Fonction d'Onde Formantique) is a synthesis technique that was originally developed by Xavier Rodet at IRCAM.

In principle it is a technique that is very similar to synchronous granular synthesis.

The main application of this technique is to simulate vocal formants used in the synthesis of the singing voice.

## Background: Formants

Formants are fixed regions of resonance in the spectrum:



The figure shows two spectra with the same formant characteristics (e.g. two notes sung on the same vowel).

**Formants** are a special feature of the singing voice, but appear on all sorts of different instrumental sounds.

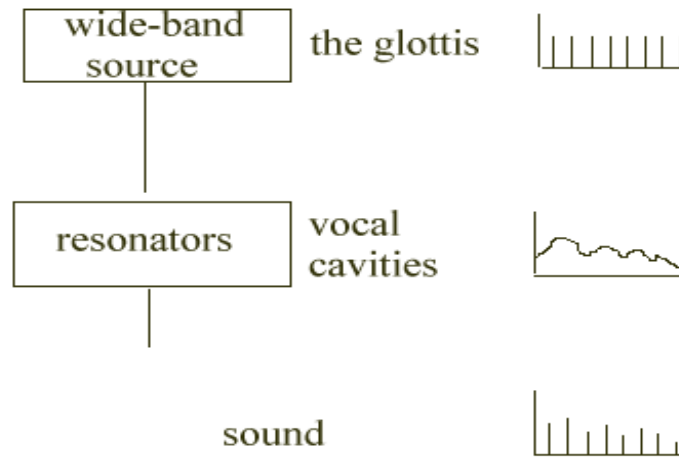
Woodwind instruments, for instance, will also feature formants. Mutes on brass instruments, the instrument body on stringed instruments will also create important formant characteristics on the source sounds.

Formants can be simulated with resonator (band-pass) filters, creating zones of boost/attenuation in the spectrum.

The *vocoder* sound is also dependent on the existence of formants.

# The singing voice, vowels and formants

A simple model for the vocal sound production is shown below:



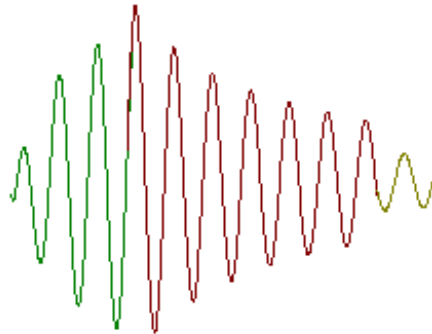
Each vowel will be created with a different set of formants.

An “o” sung by a soprano will have the first 5 formants centred at 450, 800, 2830, 3800 and 4950 Hz.

An “a” will have formants at 800, 1150, 2900, 3900, 4950 Hz.

## FOF synthesis

The principle of FOF synthesis is the generation of a stream of (sinewave) grains which might (or might not) overlap, depending on the size of the grain and the rate of generation (how fast they will be created). This is a typical FOF grain:



A stream of FOF grains generates a tone with several harmonics shaped by one formant region.

## **Fundamental frequency & formant regions**

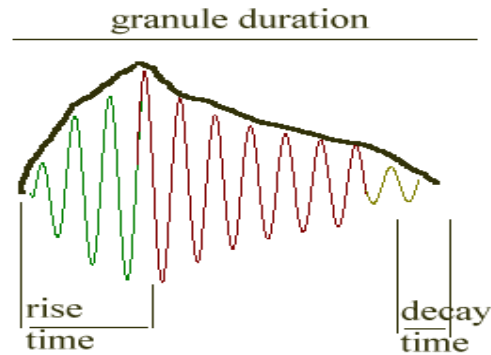
The rate at which the grains are generated on a stream determines the fundamental frequency of the tone.

Depending on the duration of the grains, a number of overlaps will occur.

The centre frequency of the formant region is determined by the frequency of the sinewave inside the grain.

The FOF principle is based on the fact that the grains are basically damped sinewaves.

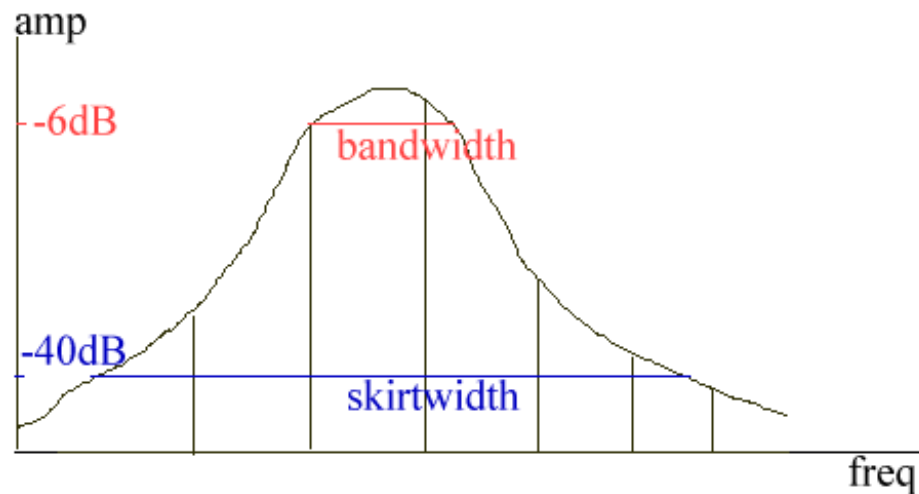
# The local envelope



The FOF grain **local** envelope (the bold line in the picture) is characterised by a **rise time**, a **duration** and a **exponential decay**. Since an exponential decay never falls to zero, this is tailed off by a fast **decay** at the end of the envelope duration.

## Bandwidth and skirtwidth

The local envelope basically determines how **wide** the formant region is. There are two parameters that influence the width of the band: the **bandwidth** at **-6dB** and the **skirtwidth** at **-40dB**.



the spectrum resulting from a FOF grain stream

## **Bandwidth & rate of decay**

The bandwidth at -6dB is determined by the rate of exponential decay of each grain. A **fast** decay rate widens the bandwidth, whereas a slow rate makes it narrow.

## **Skirtwidth & rise time**

The skirtwidth at -40dB is determined by the rise time: longer rise times make the skirtwidth smaller. Faster rise times widens the -40dB spectral region.

## FOF and Filtering

Effectively what FOF does is to package a band-limited pulse generator and a filter together. The grain start times can be seen as an impulse train, having a pulse-wave spectrum.

We recall that a pulse-wave spectrum contains all harmonics up to the Nyquist with the same amplitude (or weight).

The grains themselves can be seen as *impulse responses* of a resonating band-pass filter. Impulse responses are the output of filters when fed with a single impulse.

BP impulse responses are decaying sine-waves at the filter centre frequency, whose decay rate will determine the filter bandwidth.

## **Bandwidth and impulse response length**

FOF generate BP impulse responses at every fundamental period, overlapping them. This will effectively generate a pulse wave spectrum modified by the frequency response of the BP filter.

If the filter has a long impulse response (a long decay), the amplitude response will show a very narrow bandwidth. Conversely, if the impulse response is short, the bandwidth is wide.

At one extreme, the output is an impulse train (shortest decay), and at the other, it is a sinewave (no decay).

## **FOF grain waveform and FOF spectrum**

We can see that there is a straightforward relationship between FOF grain waveforms and the resulting sound spectrum.

Since the waveform is sinusoidal, the resulting spectrum will have a single peak at the grain frequency (the FOF formant frequency), as a band-pass filter would do.

If, instead, we use a complex wave, we should expect to generate spectra that will have more than one spectral peak, at multiples of the grain frequency. This will of course depend on the amplitude of the partials in that wave. If a non-sinusoidal wave is used, we need to be careful with possible aliasing effects.

## **The FOF UG in csound**

The csound FOF implementation is based on the FOF opcode which generates one stream of grains (and thus one formant region). In order to generate tones with several formant regions a number of these opcodes can be combined.

The basic parameters of the FOF opcode control the aspects of the synthesis mentioned before: fundamental frequency, formant frequency, grain duration and local envelope.

# The FOF opcode

```
ar  fof    xamp,xfund,xform,koct,kband,kris,kdur,kdec,iolaps,  
        ifna,ifnb,itotdur[,iphs][,ifmode]
```

**xamp** = overall amplitude

**xfund** = fundamental frequency, rate of creation of new grains

**xform** = formant frequency, the frequency of the grain wave

**koct** = octaviation.

[local envelope:]

**kband** = formant bandwidth at -6dB, in Hz

**kris** = rise time in seconds

**kdur** = grain duration in seconds

**kdec** = length of the decay tail-off in seconds.

[other:]

**iolaps** = max number of overlaps (*grain dur x fund freq*)

**ifna, ifnb** = function table numbers for **grain wave** and rise/decay shapes.

**itotdur** = total duration of synthesis

**iphs** = phase offset affecting the grain wave

**imode** = **0** => xform varies only from grain to grain

**1** => xform varies continuously

## Key FOF parameters

The pitch of the sound is basically determined by the grain generation rate, *xfund*. Grains are generated every fundamental period. The *xform* parameter is the frequency of each grain wave, controlling the centre of the spectral peak.

The local envelope is determined by *kband*, bandwidth at -6dB, which controls the gain rate of decay and an attack time, *kris* plus a decay tail-off, *kdec*. The shape of the attack and decay tail-off is taken from a function table.

The grain duration is *kdur* and this value will determine the number of overlaps at a certain fundamental (*grain dur x fund freq*).

## Octaviation

The last parameter to be discussed is the *octaviation*. This is controlled by the **koct** argument.

Octaviation changes the octave smoothly by fading out alternate grains in the stream. When these are completely faded out a fall in octave is perceived, because the rate of creation of new grains is now 1/2 the original rate.

The argument **koct** produces a change in octave every time it is increased by **1**, **0** meaning no change. So 1 is one octave down, 2 is two octaves, etc..

This can generate a granulation sensation in the resulting tone.

## **FOF and Synchronous Granular Synthesis**

FOF can be used for granular synthesis, with sampled-sound tables (GEN01).

The original FOF UG is somewhat limited and do not provide the full range of parameters needed for granular synthesis.

Two variations on this design exist to process stored sounds using granular technique.

These two opcodes are FOF2 and FOG.

## FOF2

FOF2 is very similar to FOF, except that the i-time parameter *iphase* is substituted by the k-rate *kphase*, making it possible to start the table readout at different positions during performance.

In the original FOF, grains are always generated from a fixed start position in the table. With FOF2, it is possible to continuously change it and read at different points along the table.

FOF2 also has an extra k-rate parameter (instead of FOF's *imode*), *kgliss*, which provides a *glissando control* for each grain, relative to the start pitch. This creates upwards or downwards glissandos of a set interval on each grain.

## Playback Ratio

It is important to note that In granular synthesis, the FOF parameter *kfund* is re-intepreted as *grain density*. The parameter *kform* is grain frequency.

In order to playback a sound stored in a table at the original pitch, we have to observe the following relationship:

$$formant\_freq = pitch\_transposition \times \frac{sampling\_rate}{table\_length}$$

This is related to the *sampling increment* (SI) calculation used in oscillators. To transpose a sound, we just set the pitch transposition ratio to the required interval.

# FOG

The FOG UG is a complete re-working of FOF to allow for synchronous granular synthesis. These are the changes:

- **Fundamental** frequency is now grain density.
- **Formant** frequency is substituted by a transposition ratio.
- A **new** parameter is added: *aphs*, **a-rate** table start position for grain readout (normalised). This is used for time-scale modifications. This parameter can take a position counter and use it to progress through a stored-sound table, eg.:

```
/* position counter goes from 0 to 1 in isdur*itimescale secs */  
apos   line 0, isdur*itimescale, 1  
agrain fog iamp, idens, 1, apos, 0, 0, 0.003, igd, 0.005, iol, 1, 2, idur
```

## Time-scale and Pitch modifications

One of the applications of granular (re)synthesis is to modify the timescale and pitch of a sound. With FOG this is very simple to implement:

- Pitch transposition is fed directly into FOG
- The timescale of a sound stored in a table with GEN01 is modified by controlling the parameter *aphs*. All is needed is to obtain the original sound length by using `ftlen(itable)` which returns the size of the table:

$$\mathbf{isdur = ftlen(itable) / sr}$$

This is the duration in secs of the sound stored in the table.

## Code detail

```
itime = 1.8 /* time ratio */
ipitch = 1.5 /* pitch ratio */
isampdur = ftlen(1)/sr /* sound duration */
ifindur = itime*isampdur /* final duration */
iols = 2 /* grain overlaps */
ia = 10000 /* amplitude */
igd = 0.02 /* grain duration */
ids = (1/igdur)*iolaps /* grain density (grains/sec)
                        this determined by the overlaps */

/* read pointer pos (0 - 1) */
ap phasor 1/ifindur

/* FOG */
asig fog ia,ids,ipitch,ap,0,0,0.003,igd,0.005,iols,1,2,p3

out asig
```

## Summary: key concepts

- Granular synthesis is based on composing a sound with short-time blocks of sound called *grains*.
- Granular synthesis can be synchronous or asynchronous.
- Asynchronous techniques can generate complex statistical granular sounds.
- FOF is a granular-based formant synthesis.
- FOF can be used to generate typical formant-dependent sounds such as the singing voice.
- FOF can be extended to support synchronous granular synthesis.
- In csound, the opcodes FOF2 and FOG can be used for synchronous granular techniques.