Música por computadora

Ernesto Romero y Hernani Villaseñor
Centro Multimedia 2012

Sesión 16

Síntesis Granular

La síntesis granular fue explorada de manera conceptual y aplicada en instrumentos y música electrónica a finales de los años 50 por Iannis Xenakis e implementada posteriormente en la computación por Road Curtis y Barry Truax en los años 70 (Collins, 5.2).

La síntesis granular parte del presupuesto que fragmentos microscópicos de sonido al ser escuchados juntos son percibidos como un sonido continuo (Valle, 245:2008).

Cuando combinamos muchos sonidos microscópicos para formar grandes nubes sonoras, podemos obtener paisajes sonoros macroscópicos (Collins, 5.2).

La duración de estos granos de sonido puede variar entre 1 y 100 ms, cada grano tiene un envolvente de amplitud definido lo que influye en el resultado sonoro.

En la síntesis granular podemos trabajar a partir de ondas sinoidales con un envolvente muy corto o pequeñas muestras de sonido.

Esta síntesis se basa en señal impulsiva, los granos pueden ser controlados sobre el tiempo para obtener masas sonoras, por ejemplo al alterar su densidad.

Cada grano tiene diferentes parámetros: afinación, duración del envolvente o ventana, posición en el campo estéreo o paneo.

La nube sonora cuyos parámetros afectan globalmente al sonido son: rango de frecuencias de los granos, densidad o cantidad de granos.

Materiales con los que podemos trabajar:

```
SinOsc.ar // con envolventes micro
FSinOsc.ar // con envolventes micro
PlayBuf.ar // Muestras de Sonido
```

Objetos para síntesis granular:

GrainBuf

GrainBuf.ar(numCanales, disparo, dur, sndbuf, velocidad, pos, interpolación, pan, envbufnum, maxGranos)

Es síntesis granular a partir de un sonido guardado en un buffer.

Necesitamos leer un sonido desde un buffer para lo que utilizamos Buff.read y lo igualamos a una variable.

```
b = Buffer.read(s, "sounds/a11wlk01.wav")
(
SynthDef(\GrainBuf,{|sndbuf,envbuf ,gate=1|
var sen, env, pan;
sen=GrainBuf.ar(1,
Impulse.ar(10),0.1,sndbuf,LFNoise1.kr.range(1,2),LFNoise2.kr.ran
ge(0,1),1,0,envbuf)*4;
env=EnvGen.kr(Env.asr(1,1),gate,doneAction:2);
Out.ar(0,sen*env)
}).send(s)
)

a=Synth(\GrainBuf)
```

TGrains

TGrains.ar (numCanales, disparo, bufnum, velocidad, posCentral, dur, pan, amp, interpolación)

Es un granulador de Buffer por lo que necesitamos crear un Buffer.alloc en el cual alojaremos sonido. Entonces el Buffer lo diseñamos para poder alojar sonido en segundos. Si sabemos que un segundo de audio en formato digital contiene 44100 muestras, ese número lo multiplicamos por la cantidad de segundos que queramos nuestro Buffer.

```
// buffer con dos segundos de duración en mono
b = Buffer.alloc(s, 44100 * 2, 1)

// luego hacemos un buffer para grabar sonido
```

```
(
SynthDef(\grabar,{|sen=0|
     var entrada;
     entrada=SoundIn.ar(sen);
     RecordBuf.ar(entrada, b.bufnum);
     }).send(s)
)

x = Synth(\grabar)
x.free // al declarar esta linea se queda un sonido en el buffer

// reproducimoslo grabado
(
SynthDef(\reproducir,{|salida=0|
     var senal;
     senal=PlayBuf.ar(1,b.bufnum,1);
     Out.ar(salida,senal!2)
     }).send(s)
)

y=Synth(\reproducir)

// granulamos el buffer con TGrains, usamos BufDur para saber la
duración del buffer

(
SynthDef(\Tgrains,{|salida=0, buffer=0|
     var senal;
     senal=TGrains.ar(2,Impulse.ar(26),b.bufnum,1,MouseX.kr(0,Bu
     fDur.kr(b.bufnum)),0.1,0,1,2);
     Out.ar(0,senal!2)
     }).send(s)
)

w=Synth(\Tgrains)
```

Warp1

Warp1.ar (numChannels, bufnum, pointer, freqScale, windowSize, envbufnum, overlaps, windowRandRatio, interp, mul, add)

Este UGen estira y comprime un buffer. Basado en un objeto de Csound hecho por Richard Krpens.

También existen esto objetos GrainIn, GrainSin y GrainFM

Método manual

Para hacer una sintesis granular con completo dominio de cada parametro se puede usar el siguiente algoritmo que consiste en tres partes:

a- Un SynthDef que genere los granos
b- Un Tdef que active la granulacion
c- Un grupo de Tdef que cambien los parametros en el tiempo

a- a- Un SynthDef que genere los granos

s.boot

Primero cargamos en un buffer la muestra de audio que queramos granular

```
g=Buffer.read(s,"/home/harpo/share/SuperCollider/sounds/soundsQ-System/feedback.wav")
g.play
```

Este es el SynthDef para los granos. Se crea con un PLayBuf y se crean argumentos para cada parametro de modo que podamos cambiar sus valores desde afuera. La envolvente debe ser pequeña para cada grano.

```
(
SynthDef(\granulador, {|buf=0,rate=1,pos=0, amp=1, att=1,dur=0.01,rel=1,pan=0|
                var sig, env;
                sig=PlayBuf.ar(1,buf,rate,1,pos)*amp;
                env=EnvGen.kr(Env([0,1,1,0],[dur*att,dur * (1 - (att+rel)),dur*rel]),doneAction:2);
                Out.ar(0,Pan2.ar(sig,pan)*env);
                }).send(s);
);
```

b- Un Tdef que active la granulacion

Creamos variables globales para cada parametro de nuestra granulacion

```
(
~rate={1};
```

```
~pos={0};
~amp={1};
~att={1/2};
~dur={0.01};
~rel={1/2};
~pan={0};
~dens={0.01};
);
```

Creamos un Tdef que envie estos valores a los argumentos del SynthDef. Usamos la variable ~dens para el wait. Notar que el wait y la duracion del grano pueden ser diferentes, dejando asi espacios entre los granos o bien superponiendolos. De esta forma podemos crear densidades diferentes independientes del tamaño del grano.

```
(
Tdef(\granulador, {
                inf.do{
                Synth(\granulador, [\buf, g.bufnum,
                                \rate, ~rate.value,
                                \pos, ~pos.value,
                                \amp, ~amp.value,
                                \att, ~att.value,
                                \dur, ~dur.value,
                                \rel, ~rel.value,
                                \pan, ~pan.value]);
                ~dens.value.wait;
                }
                }).quant_(0);
);


Tdef(\granulador).play
Tdef(\granulador).stop
```

Podemos asignar rangos de aleatoriedad para los parametros

```
(
~rate={rrand(0.9,1.1)};
~pos={44100*rrand(0.9,1.1)};
~att={rrand(1/2,1/100)};
~amp={rrand(1,0.1)};
~dur={rrand(0.001,0.1)};
```

```
~rel={rrand(1/2,1/100)}};
~pan={rrand(-0.5,0.5)};
~dens={rrand(0.1,0.001)}};
);
```

c- Un grupo de Tdef que cambien los parametros en el tiempo

////////////// RATE

Aqui se declara el Tdef que mueve los minimos y maximos del rango aleatorio del parametro de rate. Tambien indicamos una cantidad de tiempo que queremos que se tarde el Tdef en hacer el recorrido.

```
(
~rateSupIni=1;
~rateInfIni=1;

~rate={rrand(~rateSupIni,~rateInfIni)};

~rateSupFin=2;
~rateInfFin=0.5;

~rateTiempo=5;

Tdef(\rate, {var min, sup, pasoInf, pasoSup;
	min=~rateInfIni;
	sup=~rateSupIni;
	pasoInf=(~rateInfFin - ~rateInfIni)/100;
	pasoSup=(~rateSupFin - ~rateSupIni)/100;
	100.do{
		min=min+pasoInf;
		sup=sup+pasoSup;
		[min,sup].postln;
		~rate={rrand(min,sup)};
		(~rateTiempo/100).wait;
		}
	}).quant_(0);
)
Tdef(\rate).play
Tdef(\rate).stop
```

Aqui podemos indicar los nuevos minimos y maximos a los que queremos ir y el tiempo que se tomara en hacerlo

```
(
~rateInfIni=~rateInfFin;
~rateSupIni=~rateSupFin;

~rateInfFin=0.1;
~rateSupFin=0.8;

~rateTiempo=10;

Tdef(\rate).play
)
```

Lo mismo pero para el parametro de la posición

```
//////////////  POS
(
~posSupIni=44100;
~posInfIni=44100;

~pos={rrand(~posSupIni,~posInfIni)};

~posSupFin=44100*1.1;
~posInfFin=44100*0.8;

~posTiempo=5;

Tdef(\pos, {var min, sup, pasoInf, pasoSup;
        min=~posInfIni;
        sup=~posSupIni;
        pasoInf=(~posInfFin - ~posInfIni)/100;
        pasoSup=(~posSupFin - ~posSupIni)/100;
        100.do{
                min=min+pasoInf;
                sup=sup+pasoSup;
                [min,sup].postln;
                ~pos={rrand(min,sup)};
                (~posTiempo/100).wait;
```

```
                }
        }).quant_(0);
)
Tdef(\pos).play
Tdef(\pos).stop

(
~posInfIni=~posInfFin;
~posSupIni=~posSupFin;

~posInfFin=44100*0.5;
~posSupFin=44100*0.76;

~posTiempo=1;

Tdef(\pos).play
)
```

Biblografía

Collins, N. (s/a). SCCourse. Recuperado de: http://www.sussex.ac.uk/Users/nc81/courses/cm1/workshop.html

Valle, A. (2008). *The SuperCollider Italian Manual at CIRMA.* Torino: CIRMA.

Ixi Audio (2008). *SuperCollider Basics.* Recuperado de: www.ixi-audio.net