Música por computadora

Ernesto Romero y Hernani Villaseñor

Centro Multimedia 2012

SESIÓN 14

Convolución

Es una forma matemática de combinar dos señales para obtener una tercera, para lo cual se usa una estrategia de descomposición de impulso. En la convolución convergen tres señales: señal de entrada, señal de salida y respuesta al impulso (Smith, 1999:109).

Waveshaping

Es una técnica de síntesis por distorsión que produce un espectro con una evolución dinámica de sus componentes, pero de banda limitada, es decir, se establece un número máximo de armónicos (Dodge & Jerse, 1997:139).

El espectro en esta síntesis es generado a partir de una distorsión controlada sobre la amplitud, esto mediante un índice que permite una variación dinámica.

Para realizar esta técnica se utiliza un *waveshaper* que es el encargado de alterar la señal que pasa por el, esto ocurre cuando a la señal de entrada se le incrementa la amplitud, lo que causa un cambio de su forma en la salida, de esta manera se incrementa el número e intensidad de los armónicos.

En SC existe el objeto Shaper que sirve para hacer este tipo de síntesis. Antes, es necesario definir una tabla de onda con un Buffer.alloc

La tabla de onda almacenada en nuestro Buffer es para definir una serie de amplitudes que moldearan nuestra onda. Entonces con el método .cheby definimos el Array de amplitudes que leerá el Shaper para modificar el timbre.

Shaper.ar/kr(bufnum, entrada, multiplicación, adición)

bufnum: el número de buffer vertido en un formato de tabla de onda (wavetable) que cumple la función de transfer
entrada: la señal de entrada

```
// primero hacemos un buffer
b = Buffer.alloc(s, 1024, 1)

// luego utilizamos el método .cheby con el que mandamos un array
de amplitudes. Puedes añadir más amplitudes al array
b.cheby([1,0.5,1,0.125]);

// usamos nuestro waveshaper con una onda sonoidal
(
SynthDef(\ws,{|frec=440, gate=1|
var sen, env;
sen=Shaper.ar(b, SinOsc.ar(440, 0, 0.5));
env=EnvGen.kr(Env.asr(0.1,1,1),gate,doneAction:2);
Out.ar(0,sen*env);
}).send(s)
)

c=Synth(\ws)
c.set(\gate,0)
```

Este fenómeno ocurre de manera similar en los instrumentos acústicos, por lo que esta técnica es muy eficiente para sintetizar instrumentos, sobre todo de aliento/metal.

Karplus-Strong

Este tipo de síntesis pertenece al grupo de síntesis por modelado físico, técnica basada en la forma física de los instrumentos musicales. Entre las técnicas de modelado físico, esta es relativamente sencilla. Lo que modela esta síntesis es una cuerda pulsada.

Según Collins, se requiere de un ruido dentro de una línea de delay basado en la altura de la nota que queremos obtener, después se filtra el delay de manera sucesiva hasta que el sonido decae.

SC tiene implementado el UGen Pluck.ar para generar síntesis Karplus-Strong.

Pluk.ar (señal de exitación, *trigger,* tiempo max delay, tiempo delay, tiempo decay, coef)

señal de exitación: una señal de audio compleja
trigger: una señal de impulso para ingresar la señal a la línea de delay
tiempo max delay: tiempo máximo de delay en segundos que inicializa el *buffer* interno
tiempo delay: el tiempo de delay en segundos

tiempo decay: tiempo de caída del eco, valores negativos enfatizan armónicos nones
coef: el coef de del filtro interno, rango -1 a +1.

```
{Pluck.ar(PinkNoise.ar(0.5),Impulse.kr(3),440.reciprocal,440.rec
iprocal,10,0.1)}.play
```

.reciprocal se usa para obtener el recíproco, en este caso 1/440, lo que afina nuestra Karplus-strong en La, también puede usarse con .midicps para acceder a las notas que queramos.

Se puede diseñar este tipo de síntesis usando ruido y una línea de delay como puede ser CombL. Cottle (2005:151) propone el siguiente ejemplo, el cual hemos adaptado:

```
(
{
var arranqueEnv, atk = 0, dec = 0.001;
var arranque, delayTiempo, delayDecay = 0.5;
var notaMidi = 69; // A 440
delayTiempo = notaMidi.midicps.reciprocal;
//RandSeed.kr(Impulse.kr(1/delayDecay), 111);
arranqueEnv = EnvGen.kr(Env.perc(atk, dec), gate:
Impulse.kr(1/delayDecay));
arranque = PinkNoise.ar(arranqueEnv);
CombL.ar(arranque, delayTiempo, delayTiempo,
delayDecay, add: arranque);
}.play
)
```

Lo que sucede en el ejemplo anterior es que estamos mandando un ruido rosa con un envolvente muy corto a través de la línea de delay CombL, para repetirlo varias veces mientras decae, así es posible percibir un tono.

Nota que delayTiempo es convertido a nota midi y después se obtiene su recíproco, así conseguimos afinar el resultado.

Si descomentas RandSeed.kr obtendrás a cada impulso el mismo arranque de ruido, por lo que cada pulsación sonará igual.

Bibliografía

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

Cottle, D.M. (2005). *Computer Music with examples in SuperCollider 3.*

Dodge, Ch. y A. Jerse, T. (1997). *Computer Music: Synthesis, composition and performance*. Schirmer.

Smith, S.W. (1999). *The Scientist and Engineer's Guide to Digital Sound Processing.* San Diego: California Technical Publishing.