Música por computadora

Ernesto Romero y Hernani Villaseñor
Centro Multimedia 2012

Sesión 17

Delay

El delay es un objeto que trabaja con parámetros de tiempo creando repeticiones a partir de un sonido origen, este efecto se ha usado mucho en diferentes tipos de música, por ejemplo el Dub Jamaiquino. Cuando suena el sonido origen, inmediatamente después comienza una serie de repeticiones del mismo sonido las cuales sucede en un tiempo específico el cual se llama tiempo de delay, demás se establece la cantidad de repeticiones lo cual se llama *feedback.*

En SC exsiten tres diferentes tipos de *delay,* los denominados Delay, Comb y Allpass.

CombL es un objeto que genera una línea de delay con una interpolación lineal, para generar este proceso es necesario que el objeto contenga un buffer interno, en este caso esta dado por el tiempo máximo de Delay que establece la duración de un buffer interno, el parámetro de tiempo de decay es equivalente a parámetro de *feedback* de los delay hardware, y establece el número de repeticiones que genera la línea de delay.

CombL.ar/kr(entrada, tiempoMaxDelay, tiempoDelay, tiempoDecay, multiplicación, adición)

entrada: es la señal a la que aplicaremos el delay
tiempoMaxDelay: en segundos, su usa para inicializar el tamaño del buffer del delay
tiempoDelay: en segundos, duración entre cada repetición
tiempoDecay: en segundos, es el tiempo en el que el delay se va desvaneciendo

```
// intenta sustituir el objeto CombL por CombC y CombN
(
SynthDef(\Combdelay,{|amp=1|
     var sen,del;
     sen=WhiteNoise.ar(amp)*EnvGen.kr(Env.perc(0.01,0.1),Impulse.
     kr(0.25),doneAction:0);
     del=CombL.ar(sen,0.3,0.2,8);
     Out.ar(0,del);
}).send(s)
)
```

```
Synth(\Combdelay)
```

Existen otros dos delays de este tipo, CombN, que no usa interpolación y CombC que usa una interpolación cuadrada, la cual requiere de mayor procesamiento pero es más exacta.

AllpassL es un delay con una interpolación lineal, tiene los mismos argumentos que CombL

AllpassN.ar/kr(entrada, tiempoMaxDelay, tiempoDelay, tiempoDecay, multiplicación, adición)

```
// intenta sustituir el objeto AllpassL por AllpassC y AllpassN
(
SynthDef(\allpassdelay,{|amp=1|
     var sen,del;
     sen=WhiteNoise.ar(amp)*EnvGen.kr(Env.perc(0.01,0.1),Impulse.
kr(0.25),doneAction:0);
     del=AllpassL.ar(sen,0.3,0.2,8);
     Out.ar(0,del);
}).send(s)
)

Synth(\allpassdelay)
```

Igual que los delays de tipo Comb, Allpass tiene tres tipos de interpolación lineal, cuadrada y sin interpolación: AllpassL, AllpassC y AllpassN. La diferencia con los delays de tipo Comb es que Allpass manda a la salida de manera inmediata lo que está entrando por lo que su amplitud tiene a ser más baja, pero utilizado como delay no tienen mucha diferencia.

Delay es un objeto que genera ecos sin una parámetro que controla el *feedback.* Al igual que los anteriores, cuenta con tres tipos de interpolación lineal, cuadrada y no lineal: DelayL, DelayC y DelayN.

DelayL.ar/kr(entrada,tiempoMaxDelay, tiempoDelay, multiplicación, adición)

Para estos ejemplos usaremos el objeto Decay, de esta manera simulamos un tipo de feedback, el cual no contiene los objetos Delay.ar.

Decay.ar(señal, tiempoDecay, multiplicación, adición)

Ahora usamos el Decay y DelayL dentro de un Synth, para obtener un eco. En esta caso el argumento de multiplicación de Decay sirve como señal de entrada, lo mismo que el argumento de adición de DelayL.

```
// intenta sustituir el objeto DelayL por DelayC y DelayN
(
SynthDef(\delay,{|amp=1|
var sen,del;
sen=Decay.ar(Impulse.ar(0.5),0.1,WhiteNoise.ar);
del=DelayL.ar(sen,0.5,0.1,1,sen);
Out.ar(0,del);
}).send(s)
)

Synth(\delay)
```

PitchShift

Es un objeto que cambia el tono y esta basado en el uso de granos con un envolvente triangular y una densidad de 4:1.

PitchShift.ar(entrada, tamañoVentana, tonoRango, tonoDispersion, tiempoDispersion, multiplicación, adición)

entrada: señal de entrada
tamañoVentana: tamaño de la ventana del grano en segundos
tonoRango: rango del pitchshift entre 0.0 y 4.0
tonoDispersion: la máxima desvición aleatoria del del tono establecida por el tonoRango
tiempoDispersion: no puede ser mayor a tamañoVentana, es un offset aleatorio que va de 0 a toempoDispersión y se agrega al delay de cada grano.

```
(
SynthDef(\pitchshift,{|amp=0.5|
     var sen,rangoTono, ps;
     rangoTono=MouseX.kr(0.0,4.0);
     sen=Blip.ar(800, 6, amp);
     ps=PitchShift.ar(sen,0.03,rangoTono,0,0.02);
     Out.ar(0,ps);
}).send(s)
)

Synth(\pitchshift)
```

SoundIn

Con este objeto leemos el sonido que entra a través de la tarjeta de sonido o el micrófono de nuestra computadora. Así que toda señal analógica o acústica que quieras conectar a SuperCollider lo puedes hacer a través de este objeto.

Está basado en el objeto In, y toma 0 como la primer entrada disponible independientemente de que haya múltiples señales de entrada. En el caso de contar con una tarjeta nativa de la computadora, 0 equivale a la entrada de micrófono o al canal de entrada izquierdo, 1 al derecho y así sucesivamente.

SoundIn.ar(entrada, multiplicación, adición)

entrada: el canal o array de canales que serán leídos.

```
// este código activa el micrófono o canal de entrada de tu
computadora
{SoundIn.ar(0)}.play

(
SynthDef(\soundin,{|amp=0.5, gate=1, en=0|
var entrada, env;
entrada=SoundIn.ar(en,0.1);
entrada=Pan2.ar(entrada,0,1);
env=EnvGen.kr(Env.asr(0,1,0),gate,doneAction:0);
Out.ar(0,entrada*env);
}).send(s)
)

p=Synth(\soundin)
p.set(\gate,0) // apaga
p.set(\gate,1) // prende
```

Nota que doneAction esta en 0, esto permite usar el argumento \gate para prender y apagara la entrada de la señal.

In

La clase In nos permite leer señales de audio desde un bus, el cual es un tipo de canal auxiliar. Esto es muy útil cuando queremos procesar una señal independientemente de su proceso, es decir queremos la señal limpia en un Synth y el proceso en otro Sytnh. Para lograr esto mandamos la señal del sonido a el Synth de la señal de proceso, para lo que usamos una combinación entre el Out del Synth donde está la señal y la clase In en el Synth que recibiremos la señal que queremos procesar.

In.ar/kr(bus, numeroCanales)

bus: el número de bus desde donde leer la señal
numeroCanales: número de canales desde donde leer la señal

En un SynthDef el argumento *out* saca la señal por diferentes canales o buses. Al mandar la señal por el *out* 0, el sonido va a la bocina izquierda, pero si lo mandamos por el *out* 4 no sonará, al menos que tengamos una tarjeta de sonido multicanal. Podemos ver lo que hay en el *out* 4 si abrimos el scope del servidor con el argumento 5 entre paréntesis: s.scope(5).

```
(
SynthDef(\ki,{|gate=1, out=0|
     var sen,env;
     sen=WhiteNoise.ar(0.1);
     env=EnvGen.kr(Env.asr(0,1,0),gate,doneAction:2);
     Out.ar(out,sen*env)
}).send(s)
)

~ki=Synth(\ki)
~ki.set(\out, 4);
s.scope(5);
~ki.set(\out, 0);
~ki.set(\gate, 0);
```

Ahora construimos un Synth con la clase In y lo llamamos \in. La clase In recibe la señal de audio del *out* 4 y la manda por un Out al canal 0. Algo importante de destacar es que cuando llamamos al Synth(\in) debemos hacerlo con el mensaje .after. Este mensaje determina la ubicación de nuestro synth en un árbol de nodos. Ahora podemos tomar la señal que está en el bus 4 y sacarlo de nuevo al canal 0.

```
(
SynthDef(\in,{|gate=1|
var sen,env;
sen=In.ar(4);
env=EnvGen.kr(Env.asr(0,1,0),gate,doneAction:2);
Out.ar(0,sen*env)
}).send(s)
)
```

```
t=Synth.after(s,\in)
t.set(\gate,0)
```

Hagamos un ejercicio para utilizar las clases In y Out para asignar efectos a una señal. Primero hacemos un SynthDef que lea una muestra de audio de un Buffer y mandamos su señal al bus 9.

```
a=Buffer.read(s,"/sounds/voz.aiff")
a.play

(
SynthDef(\sample,{|gate=1|
var sen,env;
sen=PlayBuf.ar(1,a.bufnum,loop:1);
env=EnvGen.kr(Env.asr(0.1,0.5,0.1),gate,doneAction:2);
Out.ar(9,sen*env)
}).send(s)
)

d=Synth(\sample)
d.set(\gate,0)
```

Después creamos un SynthDef(\clean) que lea la señal del bus 9 y la saque limpia por el bus 0.

```
(
SynthDef(\limpio,{|gate=1|
var sen,env;
sen=In.ar(9);
env=EnvGen.kr(Env.asr(1,0.4,2),gate,doneAction:2);
Out.ar(0,sen*env)
}).send(s)
)

     e=Synth.after(s,\limpio)
e.set(\gate,0)
```

Un SynthDef(\efx1) leerá la señal del bus 9 pero le agregará un efecto reverb y la sacará por el bus 0.

```
(
SynthDef(\efecto,{|gate=1|
```

```
var sen,env;
sen=In.ar(9);
15.do({ sen = AllpassN.ar(sen, 0.05, [0.05.rand, 0.05.rand],
2) });
sen;
env=EnvGen.kr(Env.asr(2,0.5,2),gate,doneAction:2);
Out.ar(0,sen*env)
}).send(s)
)

t=Synth.after(s,\efecto1)
t.set(\gate,0)
```

Así, alternando entre el Synth(\limpio) y el Synth(\efecto1) podemos sacar la señal limpia o con efecto.

Aquí hacemos lo mismo pero con una señal proveniente del micrófono de la computadora usando la clase SoundIn.

```
(
SynthDef(\micro,{|gate=1|
var sen,env;
sen=SoundIn.ar(0);
env=EnvGen.kr(Env.asr(0,1,0),gate,doneAction:2);
Out.ar(9,sen*env)
}).send(s)
)

q=Synth(\micro)
q.set(\gate,0)

(
SynthDef(\limpio,{|gate=1|
var sen,env;
sen=In.ar(9);
env=EnvGen.kr(Env.asr(1,0.4,1),gate,doneAction:2);
Out.ar(0,sen*env)
}).send(s)
)
```

```
     w=Synth.after(s,\limpio)
w.set(\gate,0)

(
SynthDef(\rev,{|gate=1|
var sen, efecto, env;
sen=In.ar(9);
efecto= 15.do({ sen = AllpassN.ar(sen, 0.05, [0.05.rand,
0.05.rand], 2) });
env=EnvGen.kr(Env.asr(1,0.4,1),gate,doneAction:2);
Out.ar(0,efecto*env)
}).send(s)
)

z=Synth.after(s,\rev)
z.set(\gate,0)
s.queryAllNodes
```

Bibliografía

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

Netri, E. y Romero, E. (2008). *Curso de SuperCollider Intermedios.* Centro Multimedia: México.