Música por computadora

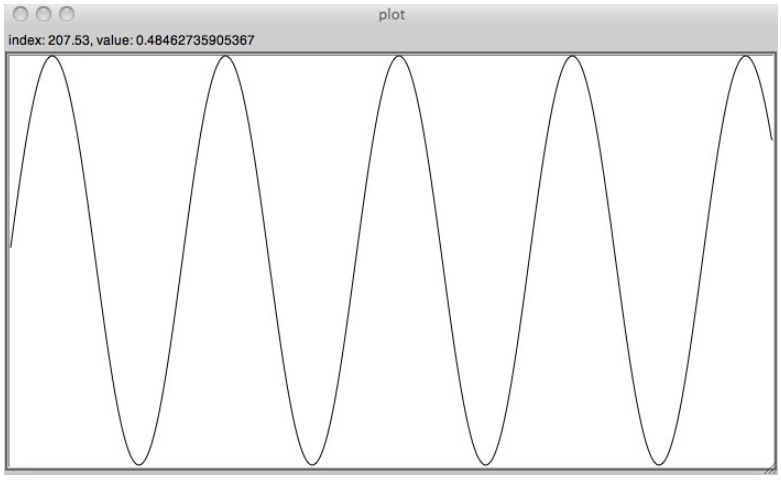
Ernesto Romero y Hernani Villaseñor

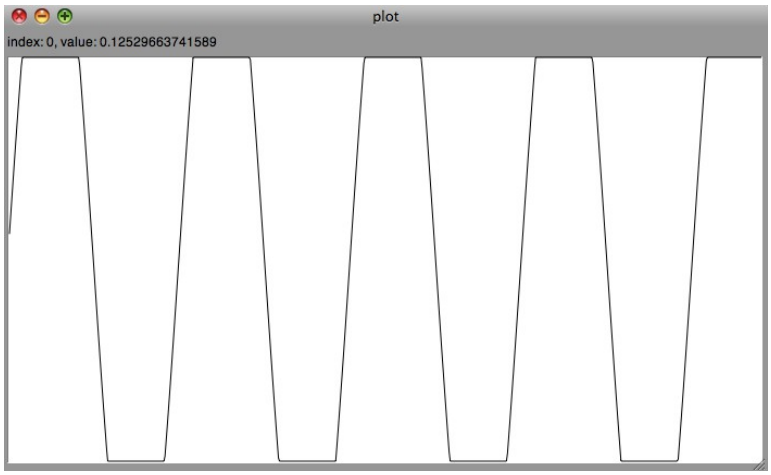
Centro Multimedia 2012

SESIÓN 15

Distorsión

La distorsión en la señal de audio se da cuando está sobrepasa la capacidad de un componente para reproducir adecuadamente el proceso al que se está sometiendo. La distorsión más común es la generada por el proceso de incrementar la amplitud. Cuando la señal sufre un recorte abrupto sobre la amplitud esta se cuadratiza, fenómeno que es conocido como *clipping*. Lo que sucede es que se la señal adquiere una forma de onda cuadrada en los picos lo que genera un gran contenido de armónicos y así se produce el sonido que conocemos como “distorsionado”. Este sonido entonces es una distorsión de la amplitud.





En las gráficas anteriores vemos en el primer caso un SinOSc con mul:1, en el segundo caso un SinOsc con mul:2, lo que sobrepasa su capacidad de rango dinámico y cuadratiza la forma redonda de la onda sinoidal.

En SC existen objetos y mensajes que tienen como finalidad generar distorsión de manera deliberada, veamos algunos.

Clip

Cuadratiza la señal a partir de un umbral determinado.

Clip.ar/kr(entrada, bajo, alto)

entrada: señal de entrada
bajo: umbral de cuadratización bajo
alto: umbral de cuadratización alto

```
// cuadratizando la onda sinoidal, prueba otros UGens
{Clip.ar(SinOsc.ar(440,0,1), -0.5, 0.5)}.scope
```

Nota que ambos umbrales deben ser un valor menor al del argumento de mul de la señal, de otra manera no ocurrirá la cuadratización.

.clip2

Este método cuadratiza de igual manera la señal, solo que a diferencia del Objeto Clip en este método no se definen los umbrales. Es necesario que el argumento de este mensaje sea menor al valor de argumento mul de la señal al que se esta aplicando.

```
// para verlo
{SinOsc.ar(440,0,1).clip2(0.15)}.plot

// para oirlo
{SinOsc.ar(440,0,1).clip2(0.15)}.scope
```

Wrap

Enrolla una señal a partir de umbrales definidos.

Wrap.ar/kr(entrada,bajo,alto)

entrada: señal que será enrollada
bajo: umbral bajo del enrrollado, debe ser menor que el alto
alto: umbral alto del enrollado, debe ser mayor que el bajo

.wrap2

Este método enrrolla la señal de manera bilateral, es un operador binario, que actúa sobre la señal de tal manera que la dobla, a diferencia del objeto Wrap, este método no define el umbral alto y bajo.

El mensaje .wrap2 recibe el argumento de la cantidad de doblado que haremos, este valor debe ser menor que el argumento :mul de nuestra señal para que actue.

```
// para ver como actua
{SinOsc.ar(200,0,1).wrap2(0.2)}.plot

// para escucharlo
{SinOsc.ar(200,0,1).wrap2(0.2)}.scope
```

.distort

Es un método de distorsión no lineal perteneciente al grupo de operadores unitarios.

Manipulación de muestras de audio

Primero cargamos un sonido, recuerda cambiar la ruta a donde guardas tus sonidos. Solo es posible usar wav y aiff

```
b=Buffer.read(s, "/home/tito/sounds/noPontoLoop.wav");
b.play; // podemos hacer sonar el audio antes
b.query; // podemos ver las características de nuestro sonido
```

También podríamos grabar un sonido, asignandolo a un buffer, para lo que usaremos el objeto RecordBuff.

RecordBuf.ar(inputArray, bufnum, offset, recLevel, preLevel, run, loop, trigger, doneAction)

Metemos el objeto RecordBuff a un SynthDef para comenzar a grabar.

```
(
SynthDef(\record, {|buf=0, dur=44100|
     var sig, env;
     sig=RecordBuf.ar(SoundIn.ar(0),buf,loop:0);
     env=EnvGen.kr(Env([0,1,1,0],[0.01,dur,
-0.02,0.02]),doneAction:2);
}).add;
)

// nos ayuda a visualizar las entradas
s.meter;

// alojamos tiempo en del Buffer
c=Buffer.alloc(s,44100*4,1);

// comenzamos a grabar
Synth(\record, [\buf, c.bufnum, \dur, 44100*4]);

// tocamos el Buffer para escuchar su contenido
c.play;

// guardamos el sonido en la ruta que queramos
c.write("/home/tito/sounds/voz.aiff", "AIFF", "int16");
```

.write(path, headerFormat, sampleFormat, numFrames, startFrame, leaveOpen, completionMessage)

path: string, entre comillas. Indica la ruta en donde se quiere escribir el archivo.

headerFormat: Formato de audio con el que se quiere escribir el archivo. Debe ir entre comillas y con mayúsculas, por ejemplo "AIFF".

sampleFormat: formato de sampleo con el que se quiere escribir el archivo, por ejemplo "int16".

```
d=Buffer.alloc(s,44100*4,1);

Synth(\record, [\buf, d.bufnum, \dur, 44100*4]);

d.play;

d.write("/home/tito/sounds/garganta.aiff", "AIFF", "int16");

e=Buffer.alloc(s,44100*4,1);
Synth(\record, [\buf, e.bufnum, \dur, 44100*4]);
e.play;

e.write("/home/tito/sounds/masca.aiff", "AIFF", "int16");

f=Buffer.alloc(s,44100*1,1);
Synth(\record, [\buf, f.bufnum, \dur, 44100*4]);

f.play;

f.write("/home/tito/sounds/pop.aiff", "AIFF", "int16");
```

Para reproducir un sonido asignado a un Buffer utilizamos el objeto PalyBuf.

PlayBuf.ar(numChannels, bufnum, rate, trigger, startPos, loop, doneAction)

```
(
SynthDef(\play, {|bufnum=0, rate=1, trigFreq=0.01, pos=0,
loop=1, dur=44100, amp=1|
     var trigger,sig, env;
     trigger=Impulse.kr(trigFreq);
     sig=PlayBuf.ar(1,bufnum,rate,trigger,pos,loop)*amp;
     env=EnvGen.kr(Env([0,1,1,0],[0.01,dur,
-0.02,0.02]),doneAction:2);
     Out.ar(0,sig*env);
}).add;
)
```

```
Synth(\play, [\bufnum, c.bufnum, \dur, (c.numFrames/44100),
\rate, 1, \trigFreq, 0.01, \pos, 0, \loop, 0]);

Synth(\play, [\bufnum, c.bufnum, \dur, (c.numFrames/44100),
\rate, 2, \trigFreq, 0.01, \pos, 0, \loop, 0]);

Synth(\play, [\bufnum, c.bufnum, \dur, (c.numFrames/44100)*4,
\rate, 1, \trigFreq, 0.01, \pos, 0, \loop, 1]);

Synth(\play, [\bufnum, c.bufnum, \dur, (c.numFrames/44100),
\rate, 1, \trigFreq, 2, \pos, 0, \loop, 0]);

Synth(\play, [\bufnum, c.bufnum, \dur, (c.numFrames/44100),
\rate, -1, \trigFreq, 0.01, \pos, (c.numFrames/44100)/4, \loop,
1]);

// empleamos una rutina con Tdef para tocar los Buffers
anteriores

(
Tdef(\bufnum, {var buf;
     inf.do{
          buf=[c,d,e,f].choose.postln;
          Synth(\play, [\bufnum, buf.bufnum,
          \dur, (buf.numFrames/44100),
          \rate, 1,
          \trigFreq, 0.01,
          \pos, 0,
          \loop, 1]);
          (buf.numFrames/44100).wait;
                    }
          }).quant_(0);
)

Tdef(\bufnum).play;
Tdef(\bufnum).stop;
```

```
(
Tdef(\rate, {var rate;
     inf.do{
          rate=rrand(0.5,2)*[1, -1].choose;
          Synth(\play, [\bufnum, c.bufnum,
          \dur, (c.numFrames/44100),
          \rate, rate,
          \trigFreq, 0.01,
          \pos, (c.numFrames/44100),
          \loop, 1]);
          1.wait;
               }
          }).quant_(0);
)

Tdef(\rate).play;
Tdef(\rate).stop;
```

BufDur.kr(bufnum) es la duración del Buffer en segundos.

```
u={PlayBuf.ar(1,e.bufnum,1,Impulse.kr(MouseY.kr(0.1,40)),MouseX.
kr(0,BufDur.kr(e.bufnum)*44100).poll)}.play;

u.free;

(
Tdef(\trigPos, {var trigFreq, pos;
     inf.do{
          trigFreq=[8,4,2].choose;
          pos=[91815, 79289, 140242].choose;
          Synth(\play, [\bufnum, e.bufnum,
          \dur, 1/trigFreq,
          \trigFreq, trigFreq,
          \pos, pos,
          \amp, 0.5,]);
          (1/trigFreq).wait;
               }
          }).quant_(0);
)
```

```
(
i={RLPF.ar(Impulse.ar(2,[0,0.5]), [1000,2000],0.1)*8}.play;
Tdef(\trigPos).play ;
)

(
i.free;
Tdef(\trigPos).stop;
)

(
SynthDef(\playMod, {|bufnum=0,dur=44100, amp=1, ringFreq=1000,
ringAmp=0, ringAdd=1, fmFreq=0,d=0,rate=1, loop=0|
          var rateFM,sig, env;
          rateFM=SinOsc.kr(fmFreq,0,d,rate);
          sig=PlayBuf.ar(1,bufnum,rateFM,
loop:loop)*SinOsc.kr(ringFreq,0,ringAmp,ringAdd);
          env=EnvGen.kr(Env([0,1,1,0],[0.01,dur,
-0.02,0.02]),doneAction:2);
          Out.ar(0,sig*env*amp);
}).add;
)

// Sin modulación
Synth(\playMod, [\bufnum, c.bufnum, \dur, (c.numFrames/44100)]);

// Modulación de Anillo
Synth(\playMod, [\bufnum, c.bufnum, \dur,
(c.numFrames/44100), \ringFreq, 1000, \ringAmp, 1, \ringAdd, 0])

// Frecuencia modulada (rate)

Synth(\playMod, [\bufnum, c.bufnum, \dur,
(c.numFrames/44100), \fmFreq, 7, \d, 0.05, \rate, 1]);

Synth(\playMod, [\bufnum, c.bufnum, \dur,
(c.numFrames/44100), \fmFreq, 100, \d, 10, \rate, 100, \loop,
1]);
```

```
// Control grano

(
SynthDef(\playGrano, {|bufnum=0, rate=1, pos=0, dur=44100,
amp=1|
          var sig, env;
          sig=PlayBuf.ar(1,bufnum,rate,1,pos)*amp;
          env=EnvGen.kr(Env([0,1,1,0],[0.001,dur,
-0.002,0.001]),doneAction:2);
          Out.ar(0,sig*env);
     }).add;
)

Synth(\playGrano, [\bufnum, d.bufnum, \dur, 0.01, \rate, 1,\pos,
0]);

~buf=d.bufnum;
~rate=rrand(0.5,2);
~dur=0.01;
~pos=rrand(0,d.numFrames);

(
Tdef(\grano, {var rate, dur, pos;
     inf.do{
          rate=rrand(0.5,2);
          dur= ~dur;
          pos=~ pos;
          Synth(\playGrano, [\bufnum, ~buf,
               \dur, dur,
               \rate, rate,
               \pos, ~pos ]);
               ~dur.wait;
               }
          }).quant_(0);
)

Tdef(\grano).play;
Tdef(\grano).stop;
```

Audacity es un programa de uso libre donde pueden editar sus sonidos antes de pasarlos a SuperCollider.