Música por computadora

Ernesto Romero y Hernani Villaseñor
Centro Multimedia 2012

Sesión 8

Live Coding

¿Qué es *Live Coding*?

Live coding es el ejercicio escénico de escribir o modificar código en el momento, con el fin de obtener un resultado estético, generalmente visual o sonoro. El código que se escribe normalmente se proyecta sobre una pantalla, de tal forma que puede ser visto por el público.

El *live coding* es una corriente que comenzó a tomar fuerza en Europa en los años 90 y actualmente es un ejercicio dentro del arte electrónico experimental. McLean (129), menciona que el término surgió alrededor del 2003, para describir una actividad con una aproximación a nuevas formas de hacer música por computadora y video animación. Asimismo sugiere que el término l*ive Coding* se usa más en el contexto de la improvisación.

Algunos festivales que integran las prácticas de *live coding* son Changing Grammars en Alemania, LOSS Livecode Festival y //FLOW en Inglaterra, Piksel en Noruega, Make Art en Francia, Lambda en Bélgica, /*vivo*/ en México y Live.Code.Festival en Alemania.

Desde el 2010 el Taller de Audio del CMM ha impulsado la práctica del *live coding* a partir de talleres, conferencias y conciertos mensuales y en 2012 con el Simposio Internacional de /*vivo*/.

Otras plataformas

ChucK
Pure Data
Impromptu
Fluxus
live-processing
Max/MSP

ProxySpace

ProxySpace es uno de los cuatro ambientes de la librería JITLib creada por Julian Rohrhurber, con la que modificamos código mientras este corre dentro de un ambiente especial llamado Proxy.

```
//prendemos el servidor y entramos al ambiente Proxy
p=ProxySpace.push(s.boot)

// declaramos un desvanecimiento del tiempo, 10 segundos
p.fadeTime=10

// creamos una variable global con el mensaje .play
~out.play

// la igualamos a una función con un UGen dentro
~out={SinOsc.ar(400,0,0.5)}

// agregamos un argumento a frecuencia y lo llamamos frec
~out={|frec=400| SinOsc.ar(frec,0,0.5)}

// modificamos el argumento frec mediante el método .set
~out.set(\frec, 600)
```

Ejemplo con Demand

```
~out2.play
~out2={SinOsc.ar (Demand.kr(Impulse.kr(6), 0,
Dseq([200,100,300],inf)))}

~out3.play
~out3={LFTri.ar (Demand.kr(Impulse.kr(2), 0,
Dseq([1000,500,250],inf)),0,0.2)}

~out4.play
~out4={LFTri.ar (Demand.kr(Impulse.kr(1), 0,
Drand([250,500,400,750],inf)),0,0.2)}
```

Ejemplo con Pbind

```
~out5= Pbind(\freq, Pseq([250,500],inf))
```

```
~out5.play
```

SynthDef dentro de Proxy

Para usar SynthDef dentro de ProxySpace, es necesario declarar el Synth con .add.

Ejemplo:

```
SynthDef(\sinte, {|frec=200,out=0|
Out.ar(out,SinOsc.ar(frec,0,0.1)
* EnvGen.kr(Env.perc(0.1,1),doneAction:2))}).add
```

De esta manera tenemos la posibilidad de usar un Synth en combinación con un Pdef. Otra forma de usar un SynthDef es de la siguiente manera.

```
// sint con demand y sin enviar al servidor

~out6=SynthDef(\hoy2,{
|frec=100|
Out.ar(0,SinOsc.ar(Demand.ar(Impulse.ar(4),0,Drand([100,200,400],
inf)),0,0.1)
* EnvGen.kr(Env.asr(0.1,0.4,2),1,doneAction:2))})

~out6.play
```

Ejemplo con Pdef

```
// usamos el SynthDef anterior llamado \sinte
~seq.play
~seq=Pdef(\uno,Pbind(\instrument,\sinte, \dur, Pseq([0.5],inf)))
~seq.release
~seq.send
```

Referencias

McLean, A. (2011). *Artist-Programmers and Programming Languages for the Arts.* (Tesis de Doctorado). University of London.