Música por computadora

Ernesto Romero y Hernani Villaseñor
Centro Multimedia 2012

Sesión 19

Secuenciadores

Un secuenciador es un dispositivo en el que acomodamos eventos sonoros en dos ejes: altura y tiempo. El secuenciador viene de la tecnología de los primero sintetizadores, y toma como modelo la estructura musical, entonces la altura está basada en tonos y el tiempo en ritmo musical.

Originalmente los secuenciadores funcionaban a partir de pasos, o sea escaños donde acomodar en evento, la distancia entre cada escaño era simétrica de tal forma que con 16 pasos se formaba un compás de 4 cuartos subdividido en 4 dieciseisavos pr cuarto, o sea 16 pasos.

Posteriormente con el protocolo MIDI, esta resolución de 16 pasos se hizo más compleja, y la posibilidad de acomodar alturas quedó determinada por los 8 bits del protocolo, lo que dió 128 lugares para acomodar notas, más de las que tiene un piano convencional, así las notas se asignaron a un número del 0 al 127 donde 60 es el do central y cada número hacia arriba o hacia abajo representa un semitono.

```
s.boot
(
SynthDef(\gacho, {|freq=100, dur=0.25, amp=0.5, add=1, fmFreq=0, d=0, out=60, pan=0|
		var sig, env;

sig=Mix(RLPF.ar(Saw.ar(freq*[1,1.005]*LFNoise1.kr([0.01,0.011],0.01,1)*SinOsc.kr(fmFreq,0,d,a
dd),amp),LFNoise2.kr(0.013,5000,12000),0.01));
		sig=LPF.ar(sig,1200,amp*2)+SinOsc.ar(freq*[1,16],0,amp*[0.75,0.025]);
		env=EnvGen.kr(Env.perc(0.02,dur),doneAction:2);
		Out.ar(out,Pan2.ar(sig, pan)*env);
		}).send(s);
);
```

```
/*
Synth(\gacho, [\dur, 10])
*/
(
SynthDef(\gachoRev, {|mix=0.5, room=0.7, gate=1|
		var sig, env;
		sig=FreeVerb.ar(In.ar(60,2),mix,room);
		env=EnvGen.kr(Env.asr(0.01,1,4),gate,doneAction:2);
		Out.ar(0,sig*env);
		}).send(s);
);
/*
~gachoRev=Synth(\gachoRev, [\mix, 0.8815, \room, 0.865419], addAction: \addAfter)
~gachoRev.set(\mix, 0.8815, \room, 0.865419)
*/



~notas=[0,2,3,2,5,3,7,12,7,3,5,5,3,2]
~ritmo=[0,2,5,7,8]
```

Se puede hacer un secuenciador también con el método switch dentro de un Tdef. Usamos el argumento i del inf.do para crear un contador y lo limitamos a 8 pasos con el modulo %8. Escogemos despues en que numero de la cuenta habrá un evento y lo encerremos en una función.

```
(
Tdef(\secSwitch, {
		inf.do{|i|
		switch(i.asInteger%8, 0, {Synth(\gacho, [\dur, 0.1, \freq, 1])},
				2, {Synth(\gacho, [\dur, 0.1, \freq, 55])},
				3, {Synth(\gacho, [\dur, 0.1])},
				5, {Synth(\gacho, [\dur, 0.1, \freq, 100* -5.midiratio])}
			);
		0.2.wait;
		}
		}).quant_(0)
)

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

Podemos meter varios switch dentro del mismo Tdef usando diferentes módulos.

```
(
Tdef(\secSwitch, {
		inf.do{|i|
		switch(i.asInteger%8, 0, {Synth(\gacho, [\dur, 0.1, \freq, 1])},
				2, {Synth(\gacho, [\dur, 0.1, \freq, 55])},
				3, {Synth(\gacho, [\dur, 0.1])},
				5, {Synth(\gacho, [\dur, 0.1, \freq, 100* -5.midiratio])}
			);

		switch(i.asInteger%16, 0, {Synth(\gacho, [\dur, 0.1, \freq, 100* 12.midiratio])},
				4, {Synth(\gacho, [\dur, 0.1, \freq, 100* 11.midiratio])},
				7, {Synth(\gacho, [\dur, 0.1, \frec, 100* 3.midiratio])},
				11, {Synth(\gacho, [\dur, 0.1, \freq, 100* 2.midiratio])},
				12, {Synth(\gacho, [\dur, 0.1, \freq, 100* 1.midiratio])},
				14, {Synth(\gacho, [\dur, 0.1, \freq, 2])},
				15, {Synth(\gacho, [\dur, 0.1, \freq, 100* 13.midiratio])}
			);

		0.2.wait;
		}
		}).quant_(0)
)

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

Otro ejemplo con desplazamientos rítmicos usando módulo 7 y 8.

```
(
Tdef(\secSwitch, {
		inf.do{|i|
		switch(i.asInteger%8, 0, {Synth(\gacho, [\dur, 0.1, \freq, 100* -5.midiratio])},
				2, {Synth(\gacho, [\dur, 0.1, \freq, 100* -4.midiratio])},
				3, {Synth(\gacho, [\dur, 0.1, \freq, 100* -3.midiratio])},
				5, {Synth(\gacho, [\dur, 0.1, \freq, 100* 0.midiratio])}
			);

		switch(i.asInteger%7, 0, {Synth(\gacho, [\dur, 0.1, \freq, 100* 12.midiratio])},
				1, {Synth(\gacho, [\dur, 0.1, \freq, 100* 11.midiratio])},
				4, {Synth(\gacho, [\dur, 0.1, \frec, 100* 3.midiratio])},
```

```
                        6, {Synth(\gacho, [\dur, 0.1, \freq, 100* 2.midiratio])}
                    );

                0.2.wait;
                }
                }).quant_(0)
)

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

Otra técnica, mas poderosa es hacer arrays con valores específicos para cada evento. En el siguiente ejemplo tenemos un array cuyos elementos son arrays con pares de valores para altura y duración.

```
~sec=[[0,1/16],[0,1/16],[0,1/16],[0,1/16], [12,1/4],[0,1/8],[0,1/8], [12,1/4]];
~fund=60.midicps;

(
Tdef(\secuenciador, {var altura, dur;
                inf.do{|i|
                        altura=~sec[i%~sec.size][0].midiratio;
                        dur=~sec[i%~sec.size][1]*2;

                        Synth(\gacho, [\freq, altura*~fund, \dur, dur]);

                        dur.wait;
                        }
                }).quant_(0);
);
/*
Tdef(\secuenciador).play;
Tdef(\secuenciador).stop;
*/

~sec2=[[8,1/2],[8,1/4],[7,1/4],[5,1], [7,2],[8,1/2],[8,1/4],[7,1/4],[5,1], [0,2]];
~fund=60.midicps;

(
Tdef(\secuenciador2, {var altura, dur;
```

```
		inf.do{|i|
			altura=~sec2[i%~sec2.size][0].midiratio;
			dur=~sec2[i%~sec2.size][1]*2;

			Synth(\gacho, [\freq, altura*~fund/4, \dur, dur*2]);

			dur.wait;
			}
		}).quant_(0);
);
/*
Tdef(\secuenciador2).play;
Tdef(\secuenciador2).stop;
*/

/*
(
Tdef(\secuenciador).play;
Tdef(\secuenciador2).play;
)
(
Tdef(\secuenciador).stop;
Tdef(\secuenciador2).stop;
)
*/
```

Se pueden agregar también valores especiales para eventos en particular. El siguiente array incluye el string "f" en algunos de sus elementos. El Tdef usa una condicionante if para detectar si existe una modulación.

```
~sec=[[-10,1/16],[0,1/16],[-9,1/16],[0,1/16], [12,1/4, "f"],[1,1/8],[0,1/8], [24,1/4, "f"]];
~fund=60.midicps;

(
Tdef(\secuenciador, {var altura, dur;
		inf.do{|i|
			altura=~sec[i%~sec.size][0].midiratio*~fund;
			dur=~sec[i%~sec.size][1]*4;
			if(~sec[i%~sec.size][2]==nil,{
			Synth(\gacho, [\freq, altura, \dur, dur*0.1, \amp, 0.4, \pan, 0.5]);
					},
					{
			Synth(\gacho, [\freq, altura, \dur, dur, \fmFreq, 8, \d, 0.5, \pan, 0.5]);
					}
```

```
                        );
                        dur.wait;
                        }
                }).quant_(0);
);
/*
Tdef(\secuenciador).play;
Tdef(\secuenciador).stop;
*/
```

Ahora veamos como crear arrays independientes de alturas y duraciones para después unirlos en un array que los agrupe por pares en arrays más pequeños de altura y duración

```
~notas=Array.rand(12,0,12);
~dur=Array.rand(12,0,1.0).normalizeSum*4;
~sec=[~notas,~dur].flop;

/*
Tdef(\secuenciador).play;
Tdef(\secuenciador).stop;
*/
```

Agreguemos la posibilidad de hacer cambios en el tempo mediante una cifra metronómica bmp (bits por minuto).

```
~bpm=60

(
Tdef(\secuenciador, {var altura, dur;
                inf.do{|i|
                        altura=~sec[i%~sec.size][0].midiratio*~fund;
                        dur=~sec[i%~sec.size][1]*((60*4)/~bpm);
                        Synth(\gacho, [\freq, altura, \dur, dur*2, \amp, 0.65, \pan, -0.5]);
                        dur.wait;
                        }
                }).quant_(0);
);
/*
Tdef(\secuenciador).play;
Tdef(\secuenciador).stop;
*/
```

Y por último agreguemos otra voz con otro secuenciador.

```
~notas2=Array.rand(12,0,12) -12;
~dur2=Array.rand(12,0,1.0).normalizeSum*16;
~sec2=[~notas2,~dur2].flop;
(
Tdef(\secuenciador2, {var altura, dur;
                inf.do{|i|
                        altura=~sec2[i%~sec2.size][0].midiratio*~fund;
                        dur=~sec2[i%~sec2.size][1]*((60*4)/~bpm);
                        Synth(\gacho, [\freq, altura/2, \dur, dur*2, \amp, 0.65, \pan, -0.5]);
                        dur.wait;
                        }
                }).quant_(0);
);
/*
Tdef(\secuenciador2).play;
Tdef(\secuenciador2).stop;
*/

/*
(
Tdef(\secuenciador).play;
Tdef(\secuenciador2).play;
)
(
Tdef(\secuenciador).stop;
Tdef(\secuenciador2).stop;
)
(
~dur=[1/16!8, 1/8!4, 1/4!2,1/2].flat.scramble;
~notas=Array.rand(~dur.size,0,12);
~sec=[~notas,~dur].flop;

~dur2=[1/16!8, 1/8!4, 1/4!2,1/2].flat.scramble;
~notas2=Array.rand(~dur2.size,0,12) -12;
~sec2=[~notas2,~dur2].flop;
)
*/
```

Máquina de Estados Finitos

Es un modelo de comportamiento de un sistema donde la señal de salida depende de una señal de entrada más sus estados anteriores.

La máquina de estados finitos Se compone de una serie de estados. En cada estado se tienen diferentes opciones para moverse a otros estados. Por ejemplo, imaginemos que tenemos 4 estados en los que podemos estar. Partimos de un estado inicial, sea el 0. Del 0 podríamos ir al estado 1 o al 2. Si vamos al estado 1 podemos ir al estado 2. Si vamos al estado 2 encontramos que de ahi podriamos ir de regreso al estado 0 o ir al estado 3. Si estamos en el estado 3 ya acabamos.

Empiezas en 0
Si estas en 0 puedes ir a 1 o 2
Si estas en 1 puedes ir a 2
Si estas en 2 puedes ir a 0 o 3
Si estas en 3 ya acabaste
Hacer el recorrido una vez

Este es solo un caso hipotético. La cantidad de estados y las opciones a donde se puede dirigir en cada estado son determinadas por quien diseña la máquina.

En SC existe el patrón llamado Psfm *Pattern Finite State Machine*, para llevar acabo este tipo de secuencias.

Pfsm(lista, repeticiones)

Cada estado consiste de un artículo y un Array de índices enteros de posibles estados próximos. El artículo inicial se escoge de manera aleatoria del Array que representa los estados de entrada, ese estado que acaba de ser escogido se regresa y el siguiente estado es escogido de los posibles estados próximos. Cuando el último estado es escogido, ahí termina el flujo de estados.

```
Pfsm([
        #[estados iniciales posibles],
        estado_1, #[estados a los que se puede ir],
        estado_2, #[estados a los que se puede ir],
        estado_3, #[estados a los que se puede ir],
        .
        .
        .
        ultimo_estado, nil
        ],
```

```
	cantidad_de_repeticiones
	)
```

Hagamos un ejemplo lineal para empezar. En este caso solo podemos empezar en el estado 0. Del estado 0 solo podemos ir al 1, del 1 solo al 2 y del 2 solo al 3.

```
~mef=Pfsm(
	[
	#[0],
	0, #[1],
	1, #[2],
	2, #[3],
	3, nil
	],1).asStream
```

Necesitamos mandar el mensaje .asStream al Pfsm para poder pedir sus resultados. Para pedir los resultados usamos el mensaje .next

```
~mef.next
```

Si declaramos varias veces ~mef.next nos arrojará los números 0, 1, 2 y 3. Después de esto nos marcará error por que el Pfsm ha terminado de hacer su recorrido y solo le pedimos que lo hiciera una vez.

Ahora hagamos el caso hipotético que planteamos al inicio.

Esto:

Empiezas en 0
Si estas en 0 puedes ir a 1 o 2
Si estas en 1 puedes ir a 2
Si estas en 2 puedes ir a 0 o 3
Si estas en 3 ya acabaste
Hacer el recorrido una vez

Se convierte en esto:

```
~mef=Pfsm(
	[
	#[0],
	0, #[1, 2],
	1, #[2],
```

```
2, #[0, 3],
3, nil
],1).asStream
```

La máquina de estados finitos es perfecta para establecer recorridos armónicos obedeciendo las reglas cadenciales de la armonía clásica. Estas reglas en palabras son:

Tenemos los siguientes acordes:

i = primer grado
ii = segundo grado
iv = cuarto grado
v = quinto grado
vi = sexto grado

Las reglas de conducción cadencial son las siguientes:

Empiezas en i
Si estas en i puedes ir a ii, iv o a v
Si estas en ii puedes ir a v
Si estas en iv puedes ir a i o a v
Si estas en v puedes ir a i, vi
Si estas en vi puedes ir a iv

Implementado en un Pfsm queda así:

```
~mef=Pfsm(
[
#[0],
1, #[1, 2, 3], // 0
2, #[3],// 1
4, #[0, 3], // 2
5, #[0, 4], // 3
6, #[2], // 4
],inf).asStream

(
SynthDef(\cadencias, {|freq=#[200,300,400,450], gate=1|
var sig, env;
sig=SinOsc.ar(freq,0,1/3).mean;
env=EnvGen.kr(Env.asr(0.1,1,2),gate, doneAction:2);
Out.ar(0,sig*env);
}).send(s);
)
```

```
(
SynthDef(\cadencias2, {|freq=200, gate=1|
var sig, env;
sig=SinOsc.ar(freq,0,1/3);
env=EnvGen.kr(Env.asr(0.1,1,2),gate, doneAction:2);
 Out.ar(0,sig*env);
}).send(s);
)

~synth=Synth(\cadencias, [\freq, [0,4,7,12].midiratio*200])
```

Con un Tdef evaluamos el ~mef.next y definimos diferentes funciones para cada estado. en estas funciones estamos estableciendo las notas de los diferentes acordes.

```
~wait=1;

(
Tdef(\cadencias, {
inf.do{
switch(~mef.next.postln, 1, {~acorde=[0 -12,4,7,12]},
2, {~acorde=[2 -12,5,9,14]},
 4, {~acorde=[-7 ,5,9,12]},
 5, {~acorde=[-5 -12,5,7,11]},
 6, {~acorde=[-3 -12,4,9,12]}
);
~synth.set(\freq, ~acorde.midiratio*200);

~wait.wait;
}}).quant_(0);
)

Tdef(\cadencias).play
Tdef(\cadencias).stop

~synth.set(\gate, 0)

~synth2=Synth(\cadencias2, [\freq, [0,2,4,5,7,9,11,12].choose.midiratio*200])
~synth2.set(\freq, [0,2,4,5,7,9,11,12].choose.midiratio*200)
~synth2.set(\gate, 0)

(
Tdef(\melodia, {
inf.do{
```

```
~synth2.set(\freq, [0,2,4,5,7,9,11,12].choose.midiratio*200);

(~wait*[1/4,1/2,1].choose).wait;
}}).quant_(0);
)

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