Seminario: Matemática Aplicada a la Composición Musical
Trabajo Final

Docentes: Dr. Pablo Cetta – Prof. Constanza Galdo
Alumno: Lucas Samaruga

1. Introducción

La obra presentada es electroacústica (acusmática) y emplea únicamente materiales generados mediante distintos recursos de síntesis y procesamiento de señales. La generación de los materiales sonoros y la puesta en forma de los mismos está realizada íntegramente en SuperCollider[1], sin la intervención de ningún otro software en ninguna etapa de la producción. Si bien se adjunta una grabación a este escrito, la pieza puede ser reproducida en tiempo real, situación prevista como medio ideal de reproducción en concierto, a partir del código empleado para su composición[2]. En la concepción de este trabajo es importante el rol que cumple el lenguaje de programación, el código actúa tanto como recurso para generar los materiales así como medio de representación y organización en cuanto a su prescripción temporal.

2. Materiales básicos (*SynthDefs*)

Una *SynthDef* es la definición de un proceso de síntesis sonora (que se envía como "programa" a la máquina de síntesis), es el equivalente a la definición de instrumento en *Csound*. La obra emplea solo cuatro *SynthDefs* diferentes y una variación por razones prácticas. A continuación se detallan cada uno de estos elementos. Los nombre de los materiales básicos son "nombres de trabajo" cuya función es la de identificar los materiales fácilmente y pueden ser o no descriptivos del material en cuestión.

Ticks

```
// En el archivo ticks.rtf
SynthDef(\tick, {
	arg out = 0, freq = 400, amp = 1, pan = 0,
		ata = 0.02, rel = 0.18, reso = 3.5;
	var src, fil, env;

	src = BrownNoise.ar;
	fil = MoogFF.ar(
		src,
		freq,
		reso
	);
	env = EnvGen.kr(Env.perc(ata, rel), doneAction:2);

	Out.ar(out, Pan2.ar(fil * env * amp, pan));
}).send(s);
```

Esta unidad de síntesis se compone de una fuente de ruido generado mediante movimiento browniano, por medio de la unidad generadora *BrownNoise*. El espectro del ruido resultante tiene la cualidad de decrecer a 6 db por octava (ver figura 1).

1 MCCARTNEY, J., 2002: 61-68
2 La versión de SuperCollider empleada es la 3.3.1 (Revisión 9267) accesible en http://supercollider.sourceforge.net.

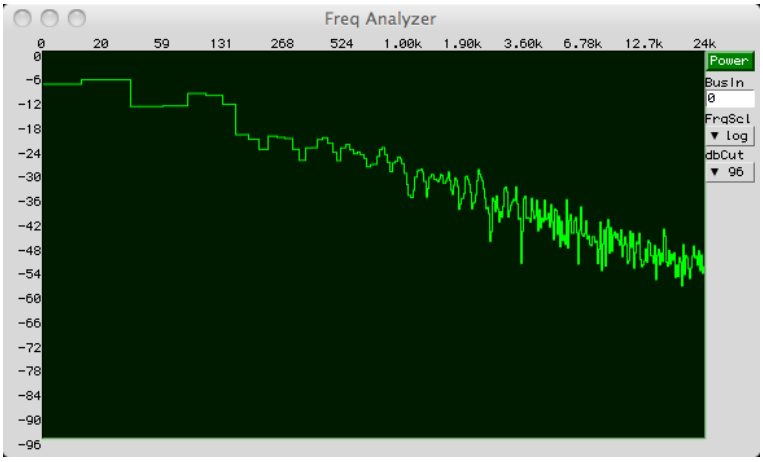

Figura 1: Representación espectral ruido browniano, frecuencia en escala logarítmica, amplitud en db.

El ruido generado es filtrado mediante un filtro resonador pasa bajos (unidad generadora *MoogFF*) y al sonido resultante se le aplica una envolvente percusiva. Los parámetros relevantes de esta *SynthDef* son: el tiempo de ataque y caída de la envolvente dinámica, que afecta directamente la duración del sonido y la ganancia de la frecuencia de resonancia del filtro pasa bajos que cambia el espectro del ruido (ver figura 2) generando mayor o menor tonicidad en el sonido[3]. Actuando conjuntamente, estos parámetros pueden cambiar notablemente la cualidad del sonido producido lo que es desarrollado en la primera parte de la obra (ver más abajo).

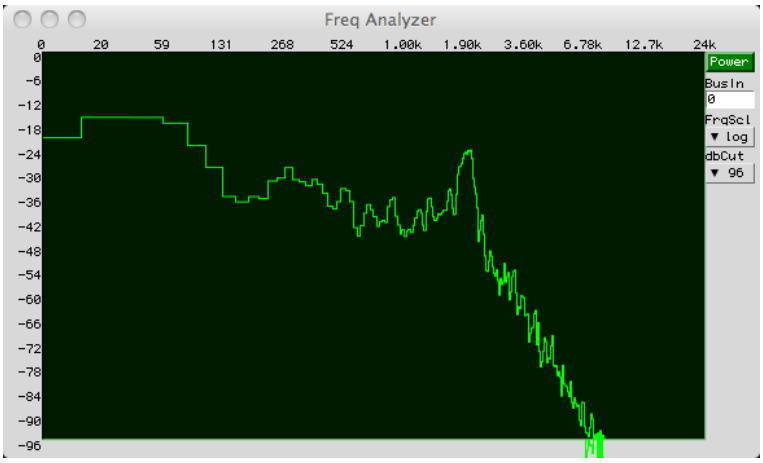

Figura 2: Ruido browniano filtrado, frecuencia de corte del filtro 2000 Hz, ganancia de la frecuencia de resonancia 3.75.

Rrrsine

```
// En el archivo rrrsine.rtf
SynthDef(\rrrsine, { arg out = 0, freq = 440, amp = 0.1, pan = 0, ata = 0.02, rel = 1.48;
      var src, noise, env;

      env = EnvGen.kr(Env.perc(ata, rel), doneAction:2);
      noise = LFNoise2.kr(20000 * amp, 0.3, 0.7);
      src = SinOsc.ar(freq, 0, amp) * noise * env;

      Out.ar(out, Pan2.ar(src, pan));
}).send(s);
```

Esta *SynthDef* se compone de un oscilador sinusoidal al que se le agrega una envolvente dinámica percusiva, con las mismas características que en el material anterior. Al mismo tiempo

3 FONTANA, F., 2007: 25-31

actúa una envolvente de ruido (*LFNoise2*) cuya frecuencia varía de manera proporcional a la amplitud del sonido, entre 0 y 20 Khz cuando la amplitud va de 0 a 1. El espectro que genera *LFNoise2* a partir de la frecuencia de la sinusoide se asemeja al generado por la caída del filtro *MoogFF* (ver figura 3). Este material es complementario al material anterior. En la composición del sonido se usa como fuente principal un sonido tónico, con un ancho de banda muy reducido, al que se le agrega una componente de ruido (ver figura 4), mientras que en el material anterior, partiendo del ruido browniano se genera un sonido tónico al modificar la envolvente espectral mediante un filtro.

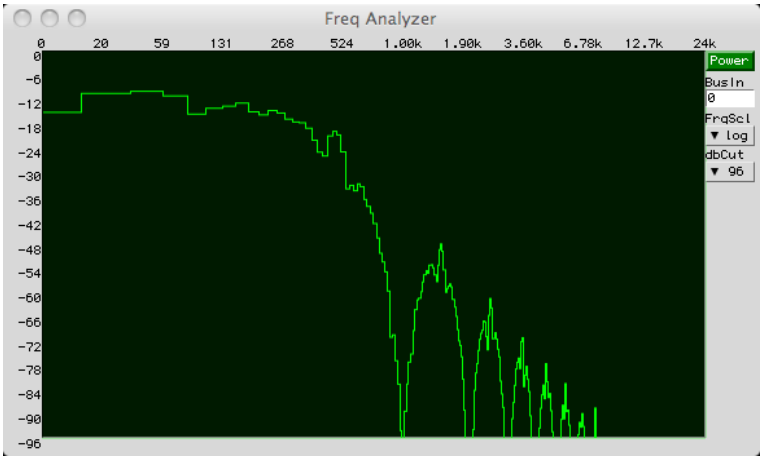

Figura 3: Oscilador de ruido de baja frecuencia con (*LFNoise2*) generando valores a 1000 Hz.

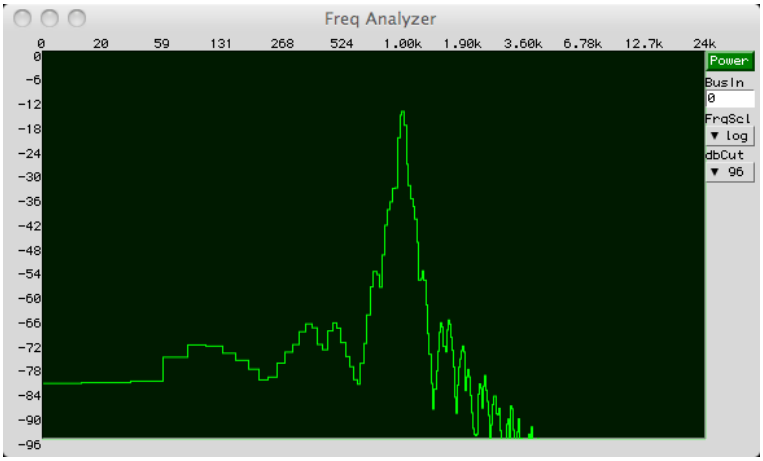

Figura 4: Sinusoide a 1000 Hz con ruido agregado, la amplitud empleada es de 0.5 y la frecuencia resultante del generador de ruido es de 20000 Hz * 0.5 Hz o 10 Khz.

BinsFiltrado

Este material emplea la transformada rápida de Fourier de dos maneras: filtrando los sonidos que pasan a través de él y distribuyéndolos de manera equitativa en dos canales. Esta *SynthDef* está diseñada para actuar en conjunto con el resto de los materiales relacionándolos mediante un mismo tipo de procesamiento y permitiendo la alternancia entre ellos.

```
// En el archivo binsFiltrado.rtf
SynthDef(\rango0256, {
      arg out = 0, inBus = #[0, 0, 0], amp = 0.2, oscil = 0.2, t_gesto = 0,
          posx = 1, posy = 1, amplitud = 1;
      var in, chain1, chain2, envgesto, geoscil, geamp;

      envgesto = EnvGen.kr(Env.perc(0.2, 1.98), gate: t_gesto);
```

```
	geoscil = envgesto * 2 + oscil;
	geamp = envgesto * 0.5 + amp;

	in = SelectX.ar(posy, [
		SelectX.ar(posx, [
			Mix.new(In.ar(inBus[0])) * geamp, // rrrsine
			Mix.new(In.ar(inBus[1])) * geamp // columbia
		]),
		SelectX.ar(posx, [
			Mix.new(In.ar(inBus[2])) * geamp, // ticks
			BrownNoise.ar * geamp
		])
	]);

	chain1 = FFT(LocalBuf(1024), in);
	chain2 = FFT(LocalBuf(1024), in); //PV_Copy(chain1, LocalBuf(1024));

	chain1 = chain1.pvcollect(1024, { arg mag, phase, i;
		if(i.even) {
			[mag * SinOsc.kr(geoscil + rand2(0.1), 2pi.rand, 0.5, 0.5), phase]
		} {
			[0, 0]
		}
	}, frombin: 0, tobin: 256, zeroothers: 1);

	chain2 = chain2.pvcollect(1024, { arg mag, phase, i;
		if(i.odd) {
			[mag * SinOsc.kr(geoscil + rand2(0.1), 2pi.rand, 0.5, 0.5), phase]
		} {
			[0, 0]
		}
	}, frombin: 0, tobin: 256, zeroothers: 1);

	Out.ar(out, [IFFT(chain1), IFFT(chain2)] * amplitud);
}).load(s);
```

Es el material básico más complejo y se compone de dos partes. La primera se encarga de recibir los materiales básicos, “Ticks”, “BinsFiltrado” y “Columbia”, y de distribuirlos junto con un generador de ruido browniano en una matriz cuadrada. Esto se realiza mediante la combinación de tres unidades generadoras *SelectX*, que se encargan de realizar un *cross fade* entre dos señales de entrada. Cómo las señales provenientes de los otros materiales son estéreo primero se realiza una mezcla de ambos canales por cada señal (ver figuras 5 y 6).

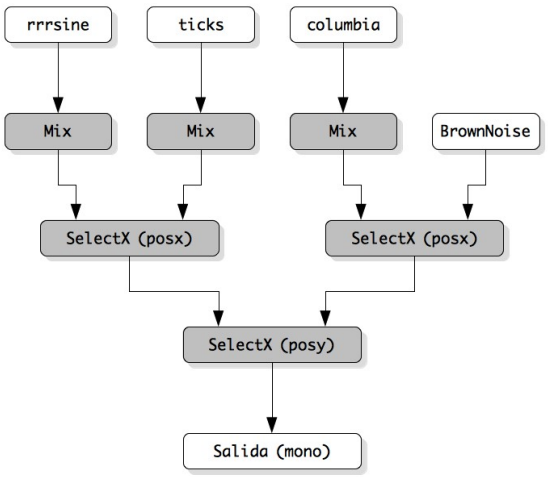

Figura 5: Configuración de las unidades generadoras para la matriz de señales.

Esta matriz se emplea como un espacio bidimensional donde cada material corresponde a un cuadrante sobre el cual tiene mayor grado de amplitud que los demás. Esta configuración es empleada para realizar recorridos, asignando secuencias de valores a las variables *posx* y *posy*, que produzcan cambios graduales en la mezcla de salida. Por ejemplo, el siguiente código:

```
//...
w.posy = Pseg(Pseq([2, 0.000001], inf), Pseq([10, inf], inf), \exponential).asStream;
w.posx = Pseg(Pseq([2, 0.000001], inf), Pseq([10, inf], inf), \linear).asStream;
//...
```

genera dos funciones, una lineal para la variable *posx* y otra exponencial para la variable *posy*, que al actuar temporalmente como valores de entrada de la *SynthDef* producen el siguiente recorrido en un plano:

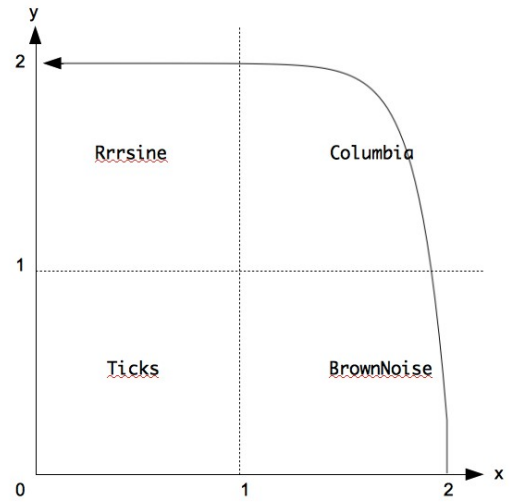

Figura 6: Recorrido sobre la matriz de materiales como espacio bidimensional.

En la segunda parte de este material se aplica la transformada rápida de Fourier sobre la señal resultante del proceso anterior. Primero se realiza la transformada propiamente dicha por medio de la unidad generadora *FFT*. Se emplea una ventana de 1024 muestras, con un corrimiento de 512 muestras entre ventana y ventana. El tipo de ventana empleado es una función sinusoidal, que es el valor por defecto de la unidad de análisis, y además, es el valor recomendado al emplear *FFT* para *phase vocoder*.[4]

El proyecto emplea una frecuencia de muestreo de 48 Khz, lo que resulta en una frecuencia de análisis de 48000 Hz / 1024 = 46.875 Hz, lo que se aproxima a un $fa\#_{-1}$ en notación musical. Los datos espectrales resultantes se procesan para cada canal de salida, por medio de la función *pvcollect*, desde el *bin* 0 hasta el 256 (por eso la *SynthDef* es llamada *\rango0256* en el código fuente) poniendo en cero todos las bandas superiores a 256. Esto genera un filtrado del espectro de la señal de entrada, dejando pasar solo las bandas por debajo de 48.875 Hz * 256 = 12000 Hz.

La distribución de las bandas se hace enviando las pares al canal izquierdo y las bandas impares al canal derecho. Además, por cada banda se emplea un generador sinusoidal independiente como envolvente dinámica, con frecuencia y fase aleatorias entre 0.1 Hz y 0.3 Hz (por defecto) y 0 y 2π respectivamente. Esto último genera un cambio tímbrico periódico en el sonido que pasa a través de esta *SynthDef*.

Esta *SynthDef* tiene una variación, *\rango1035*, que emplea los mismos recursos y parámetros antes mencionados pero con dos diferencias: solo utiliza ruido browniano como sonido de entrada y el ancho de banda resultante va desde el *bin* 10 hasta el *bin* 35. Quedando el espectro en un rango de frecuencias que va desde 468.75 Hz a 1640.625 Hz.

4 Véase la documentación de SuperCollider disponible en http://supercollider.svn.sourceforge.net/viewvc/supercollider/trunk/common/build/Help/UGens/FFT/FFT.html.

Columbia

```
// En el archivo columbia.rtf
SynthDef(\columbia, {
	arg out = 0, amp = 0.1, wipe = 1.0, width = 1.0, ata = 0.02, rel = 2.48, dust = 20;
	var in, chain1, chain2, env;

	env = EnvGen.kr(
		Env.new([0, 1, 0.9, 0], [ata, 0.2, rel], [3, -3, -3]),
		doneAction: 2
	);

	in = PlayBuf.ar(1, c, BufRateScale.kr(c), loop:1) * env;
	chain1 = FFT(LocalBuf(1024), in);
	chain1 = PV_BinScramble(chain1, wipe, width, Dust.kr(dust));

	chain2 = FFT(LocalBuf(1024), in); // PV_Copy(chain1, LocalBuf(1024));
	chain2 = PV_BinScramble(chain2, wipe, width, Dust.kr(dust));

	chain1 = chain1.pvcollect(1024, { arg mag, phase, i;
		if(i.even) {
			[mag * Line.kr(1, 0, Rand(0.02, ata + rel)), phase]
		} {
			[0, 0]
		}
	}, frombin: 5, tobin: 160, zeroothers: 1);

	chain2 = chain2.pvcollect(1024, { arg mag, phase, i;
		if(i.odd) {
			[mag * Line.kr(1, 0, Rand(0.02, ata + rel)), phase]
		} {
			[0, 0]
		}
	}, frombin: 5, tobin: 160, zeroothers: 1);

	Out.ar(out, [IFFT(chain1), IFFT(chain2)]);
}).load(s);
```

Este material básico procesa mediante FFT un archivo de audio que se distribuye junto con *SuperCollider* entre los ejemplos de audio: *a11wlk01-44_1.aiff*.

El procesamiento realizado es similar al de la *SynthDef* anterior en cuanto al filtrado y la distribución estéreo del espectro. Emplea las mismas dimensiones y tipo de ventana pero procesa la información espectral entre los *bin* 5 y 160. Si nuestra frecuencia de análisis era de 46.875 Hz, esto nos da un rango de frecuencias entre 46.875 * 5 = 234.375 Hz y 46.875 * 160 = 7500 Hz.

Una de las diferencias con el material “BinsFiltrado” es que antes de realizar el filtrado y la distribución, mezcla aleatoriamente las bandas del espectro analizado por medio de la unidad generadora *PV_BinScramble*. La cantidad de *bins* que se mezclan está determinada por la variable *wipe*, que varia entre 0 (ninguno) y 1 (todos). La variable *width* determina la máxima distancia aleatoria que un *bin* puede alejarse de su posición original, también es un valor normalizado entre 0 y 1. La variable *dust* de este material, determina la cantidad promedio de eventos aleatorios por segundo en los cuales se vuelven a mezclar los bins mediante *PV_BinScramble*, lo que produce un sonido más o menos iterado. Estos tres parámetros proporcionan un alto grado de variación tímbrica en el sonido resultante.

La otra diferencia con el material “BinsFiltrado” está en las envolventes dinámicas empleadas. Para cada *bin* se emplea una envolvente de caída lineal cuya duración varia aleatoriamente entre 0.02 seg. y el tiempo de ataque más el tiempo de caída de la envolvente global, esto produce una variación adicional en el espectro resultante. La envolvente global del material tiene tres segmentos de los cuales se manipula solo la duración del ataque y la caida.

3. Materiales Elaborados

La generación de eventos sonoros se produce casi exclusivamente en sucesión, las superposiciones son producto del solapamiento entre el cuerpo o la caída de un sonido y el comienzo de otro. Todos los eventos son generados sucesivamente a excepción de los ataques repetidos a partir de la mitad de la obra y la superposición por *cross fade* en el material básico de filtrado y distribución en estéreo por medio de la FFT (ver el material básico "BinsFiltrado").

En un nivel estructural y temporal mayor al de las *SynthDefs*, la obra se compone de tres principios generadores que sirven para elaborar los materiales básicos, estos son: manipulación paramétrica, síntesis granular y agrupaciones. Complementariamente, los criterios de organización del código que genera la obra, sirven para relacionar constructivamente los materiales básicos con estos tres principios y para organizar los elementos dentro de una abstracción más amplia. Esto es lo que denominamos "material elaborado", el cual debe ser entendido como una abstracción de más alto nivel que sirve para manipular tanto los objetos como los procesos sonoros.

Es necesario tener en cuenta que el límite entre materiales y procesos se ve difuminado constantemente y que los principios generadores propuestos pueden estar actuando de manera jerárquica. La resultante sonora final de todos estos materiales, procesos y principios se expone en la sección siguiente (ver sección 4.).

Manipulación paramétrica

Las modificaciones en los parámetros de síntesis de los materiales básicos, descritos en la sección anterior, pueden generan diferentes tipos de objetos sonoros contenidos dentro de un evento como unidad. Estos eventos individuales pueden estar contenidos dentro de distintas agrupaciones.

Síntesis granular

El agrupamiento de los materiales básicos pueden generar sonidos complejos, perceptibles como un objeto o unidad en un nivel temporal mayor, según se modifique la densidad cronométrica en una sucesión de eventos[5].

Agrupaciones

Las diferentes agrupaciones entre los materiales sonoros básicos descritos en la sección anterior (ver sección 2.), generan diversos materiales a nivel textural. Por supuesto pueden generarse configuraciones melódicas, pero los materiales son tratados como objetos más amplios, agrupaciones, que son manipuladas como una unidad.

Organización del código

Cada uno de los materiales básicos es envuelto en en un objeto *Environment* y asignado a una variable global[6]. Lo que se hace con este objeto es agrupar uno o varios materiales básico con un proceso que se puede ejecutar en paralelo junto con los demás. Así, los materiales básicos envueltos se corresponden con las letras "t", "u", "v", "w" y "x" de la siguiente manera:

- La letra "t" organiza al material básico "Ticks".
- La letra "u" organiza al material básico "Rrrsine" junto con el material "Ticks".
- La letra "v" organiza al material básico "Columbia".
- La letra "w" organiza al material básico "BinsFiltrado".
- La letra "x" organiza a la variación del material básico "BinsFiltrado".

5 ROADS, C., 1996, 2002.

6 Las variables globales en *SuperCollider* están representadas por los caracteres de la "a" a la "z".

Estas variables representan abstracciones más complejas de materiales musicales, que actúan al nivel de los "materiales elaborados". En resumen, al material básico se le vincula un proceso que actúa como material elaborado, según los principios generadores, y que define un comportamiento básico.

Por ejemplo, el siguiente código pertenece al objeto *Environment* que envuelve al material "Columbia":

```
// En el archivo columbia.rtf
v = Environment.make({
	~outBus = 0;
	~grupo = nil;

	~amp = 1;
	~wipe = 1.0;
	~width = 1.0;
	~ata = 0.02;
	~rel = 2.48;
	~dust = 20;

	~ritmo = 1;
	~vel = 1;

	~task = Task.new({
		loop {
			Synth(\columbia, [
				out: ~outBus,
				amp: ~amp.next,
				wipe: ~wipe.next,
				width: ~width.next,
				ata: ~ata.next,
				rel: ~rel.next,
				dust: ~dust.next
			], ~grupo);
			(~ritmo.next * ~vel.next).wait;
		};
	});
});
```

Las variables de este objeto están precedidas por el carácter "~", la variable *~task* es la que contiene al proceso que se ejecuta en paralelo. Los valores asignados a las demás variables definen el comportamiento por defecto del material elaborado. En el código puede verse que no solo están los parámetros correspondientes a este material básico sino que también se agregan las variables *~ritmo* y *~vel*, que definen el ritmo y el *tempo* de la sucesión de eventos de este proceso. De esta manera si ejecutamos la siguiente sentencia:

```
v.task.play;
```

obtenemos una sucesión de eventos del material básico "Columbia" con sus parámetros por defecto y un ritmo de negra a un *tempo* de negra 60, que son los parámetros por defecto asignados al proceso. Esta sucesión de eventos puede variar de muchas maneras cuando se manipulan los parámetros contenidos en el objeto "v". Por ejemplo:

```
v.task.stop;
v.ritmo = Pseq([1, 0.5, 0.5], inf).asStream;
v.dust = 60;
v.rel = 0.75;
v.task.play;
```

La primera sentencia detiene el proceso iniciado anteriormente, la siguiente le asigna un ritmo de negra más dos corcheas, la tercera hace que el sonido sea más iterado, la cuarta hace que la duración

sea más corta y la quita ejecuta el proceso con la nueva configuración.

Esta última abstracción es la que se emplea para manipular los materiales sonoros a lo largo de la obra. No se explican en detalle todos los recursos empleados, propios del lenguaje, porque exceden ampliamente el alcance de este análisis. Vale mencionar que el lenguaje posee otras abstracciones que posibilitan agrupar *SynthDefs* y procesos musicales, pero que fueron descartadas porque no se adaptaban exactamente con la concepción compositiva de este trabajo.

4. Organización formal

La pieza se divide en cuatro secciones definidas por ciertas características comunes en el empleo de los materiales y procesos que las constituyen. Dentro de cada una se producen diversas inflexiones y desarrollos mediante la elaboración de los materiales.

Primera sección

Va desde 0" hasta 1' 30" aproximadamente. Está compuesta en su totalidad por el material básico “Ticks”, organizado dentro del objeto “t”. Sus variaciones emplean los tres principios generadores descritos anteriormente.

El desarrollo de esta sección está basado en lo cambios de densidad cronométrica y en las transiciones entre sonidos complejos, texturas y sucesiones que esto provoca. Creando subdivisiones formales y nuevos materiales.

La mayoría de los materiales se manejan como masas sonoras, empleando procesos aleatorios simples. El control de las alturas se realiza según probabilidades de aparición, con distribuciones variables, de clases de alturas y octavas. El material rítmico está compuesto de tres valores (1, 0.7 y 0.4) que se suceden aleatoriamente con una distribución uniforme.

Segunda sección

Va desde 1' 30" hasta 2' 51" aproximadamente. Está compuesta por el materiales básicos “Ticks” y “Rrrsine”, y su organización en el objeto “u”. Sus variaciones se rigen según el principio agrupamiento. A esto se le superpone el proceso “v” que desarrolla el material básico “Columbia” según el principio de manipulación paramétrica.

El material elaborado contenido en el objeto “u” consiste en una sucesión de ataques cuasi isócronos al comienzo, los valores rítmicos reales son, en segundos, 0.4, 0.5 y 0.45. El proceso rítmico de este objeto se basa en una transición gradual desde estos valores hacia 0.5, 0.2 y 0.3. El cambio rítmico, junto con una ampliación gradual en el conjunto de alturas generan una direccionalidad claramente definida que se interrumpe abruptamente a los 2' 51", luego de alcanzar cierto grado de densidad, por un sonido generado por síntesis granular. Los componentes rítmicos y tímbricos de este nuevo sonido son exactamente los mismos que venían sonando, pero que al estar centrados en una sola altura, generan un sonido continuo.

Una cualidad importante de este material es que realiza un “paneo rítmico” de sus ataques, entre izquierda, centro y derecha. Para la distribución temporal de la disposición espacial se empleó una permutación circular de la sucesión 1, -1, 0[7], de derecha a izquierda, a lo que se le agregó la primer fila con los elementos uno y dos intercambiados como se muestra a continuación:

```
// En el archivo rrrsine.rtf
~pan = Pseq([
      Pseq([1, -1, 0]),
      Pseq([-1, 0, 1]),
      Pseq([0, -1, 1]),
      Pseq([-1, 1, 0])
], inf).asStream;
```

7 1 representa al canal derecho, 0 al centro y -1 al canal izquierdo.

Superpuesto al objeto “u” aparece el material básico “Columbia” cuyo comportamiento está basado en una sucesión de apariciones, mucho más expandidas y espaciada temporalmente. A la vez que las apariciones y silencios de este material se van expandiendo gradualmente, el sonido va en *crescendo*, acompañando la direccionalidad del objeto “u”.

Tercera sección

Va desde 2' 51" hasta 3' 55" aproximadamente. Es una sección breve, que actúa como transición formal entre la segunda y la cuarta parte. Elabora de manera diferente los mismos materiales que la sección anterior, pero introduce el comportamiento de ataques repetidos característicos del material con el que comienza la sección siguiente. En esta sección se emplea la variación del material básico “BinsFiltrado” envuelta en el objeto “x”. El material elaborado “u” pasa a usarse como generador de síntesis granular[8], realizando la nota tenida que se yuxtapone con la sección anterior y que, al interrumpirse, enlaza con el comportamiento de ataques repetidos en conjunto con los materiales “Columbia” y la variación de “BinsFiltrado”.

Cuarta sección

Va desde 3' 55" hasta el final. Está compuesta por los ataques repetidos del material básico “Ticks” y la superposición de los materiales anteriores, a través del material “BinsFiltrado” controlado mediante el objeto “w”.

Internamente esta sección tiene una forma clara, ABABA, siendo A los ataques repetidos y B el objeto “w”. Los ataques repetidos son isócronos. Este es un material extremadamente minimalista, que se compone de tres “Ticks” superpuestos y cuya única variación es la de retrasar progresivamente y de manera leve, dos de sus tres componentes.

El objeto “w” controla la apariciones de variaciones de los materiales anteriores de manera estocástica. Para determinar las funciones que recorren la matriz de materiales se emplea el movimiento browniano, solo que a una escala temporal mucho mayor a la empleada para generar ruido. La aparición de los materiales previos está sujeta al azar, distintas ejecuciones de la pieza pueden resultar en distintas sucesiones de materiales, pero el producto final se encuadra siempre dentro de las características tímbricas del procesamiento de filtrado y distribución estéreo que implica el material “BinsFiltrado”.

5. Conclusiones

El análisis de la obra está basado, en parte, en las preconcepciones compositivas y la exposición de los recursos empleados. Las secciones 2 y 3 exponen estas ideas estructurales y la sección 4 expone la puesta en forma de los materiales musicales. Mi principal interés compositivo y analítico está orientado hacia los límites que existen entre las distintas estructuras que conforman una obra de música electroacústica. Estas estructuras, a veces conceptuales, a veces sonoras, y a veces impuestas de manera externa por el software, no son tan claras como en la música instrumental. La música por computadora no solo permite un grado de control temporal y estructural más fino sobre el sonido[9] sino que además desdibuja los límites convencionales entre materiales y procesos[10], más claramente disociados en la música instrumental.

Con respecto a los recursos matemáticos, en este tipo de composiciones se convierten en elementos básicos para trabajar con el sonido y las estructuras musicales. No se emplean solo como procedimiento aplicables a un material musical sino que conforman los parámetros elementales que el compositor manipula de manera directa, desde la generación del sonido hasta la prescripción de

8 CURTIS, R., 1996
9 CURTIS, R., 2002
10 WISHART, T., 1996

los eventos. Esto es consecuencia del empleo de los lenguajes de programación como medio de representación y producción musical, lo cual tiene sus ventajas, pero no se puede omitir que también impone sus restricciones.

6. Bibliografía

CETTA, Pablo., DI LISCIA, Oscar. P., Elementos de contrapunto atonal. En imprenta.

DI LISCIA, Oscar. P., 2004 Generación y procesamiento de sonido y música a través del programa Csound. Bernal: Universidad Nacional de Quilmes.

FONTANA, F., 2007 “Preserving the digital structure of the Moog VCF”, Actas de la ICMC07, Copenhagen, pp. 25-31, Agosto de 2007.

MCCARTNEY, James, 2002 “Rethinking the computer music language: SuperCollider”, Computer music Journal, vol. 26, no 4, pp. 61-68.

ROADS, Curtis, 1996, The Computer Music Tutorial. Cambridge, Massachusetts: MIT press.

ROADS, Curtis, 2002, Microsound. Cambridge, Massachusetts: MIT press.

SMITH, Steven W., 1997, The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing. Versión electrónica disponible en http://www.dspguide.com. Fecha de último acceso: 1-4-2010.

WISHART, Trevor, 1996, On Sonic Art. New York: Routledge Tylor & Francis Group.