Música fractal-algorítmica y antropología de la música: aportes computacionales a la búsqueda de *universales*

Lucas M. Sgrecia
UNIVERSIDAD DE BUENOS AIRES

Introducción

En este artículo se presentarán algunos programas informáticos cuya función principal es la de producir música a través de algoritmos y fractales. Para ello se procederá, en primer lugar, a describir con cierta profundidad las prestaciones que aporta uno de ellos, en particular, y los métodos que suelen emplear, en general, para computar música. En segundo lugar, se compartirán algunos comentarios personales del autor suscitados a partir de la experiencia de uso. En tercer lugar se propondrán algunos elementos de juicio que el empleo de este tipo de software puede aportar, según el autor lo considera, a la búsqueda de *universales* en la antropología de la música. En cuarto y último lugar, se ofrecerán unas pocas líneas a modo de cierre.

Se ha seleccionado en particular el programa **FractMus 2000** para este análisis, de entre muchos otros sistemas del mismo tipo, por un par de razones. La primera es que, como también estima el Prof. Carlos Reynoso en su libro *Complejidad y Caos: una exploración antropológica* (Reynoso; 2006a: 403, 405) integra un selecto grupo de "herramientas recomendadas para la investigación científica o de alta calidad técnica". Este programa pone al servicio del usuario una numerosa cantidad de prestaciones muy interesantes y presentadas de una manera considerablemente sencilla para que puedan ser aprovechadas al máximo, a diferencia de otros sistemas que el autor ha conocido y explorado y que no ofrecen una gama tan variada y rica de opciones y un modo de uso tan claro y simple. No obstante esto, algunos de esos programas también serán tenidos en cuenta brevemente aquí. La segunda es que este programa puede conseguirse gratuitamente en Internet y no es necesario registrarlo para que todas sus funciones estén habilitadas y el usuario las pueda disfrutar directamente.

Es necesario mencionar un inconveniente que se presenta al momento de la elección de este programa en especial: no se ha actualizado y nada indica que serán lanzadas al público nuevas versiones mejoradas, como sí ha ocurrido con otros programas, lo cual atenta contra su popularidad y dificulta un debate más amplio sobre sus alcances con otros colegas y entusiastas.

De todas maneras, el autor de este artículo estima que la facilidad de empleo que le brinda la interfaz gráfica de este programa al usuario inexperto, junto con la cierta riqueza de sus herramientas musicales, permiten en buena medida una aproximación inicial a los elementos que la utilización de este tipo de software en general puede aportarle al antropólogo de la música, aunque no sea más que para estimular la imaginación.

FractMus 2000

Registrado bajo la modalidad de Freeware por Gustavo Díaz-Gerez, pianista español nacido en las Islas Canarias, quien luce una trayectoria relativamente afamada a juzgar por las referencias que uno puede encontrar en Internet, fue escrito en C++ con Microsoft Visual C++ 5.0 y tiene más de 30.000 líneas de código.

En base a la información aportada en la sección de Ayuda del programa, junto a lo aprendido a través de su empleo directo, podemos describir su funcionamiento de manera general. **FractMus 2000** computa música teniendo en cuenta principalmente tres elementos: los Parámetros de la Ventana Principal (Main Window Parameters), los valores del Contador de Inicio (Start Counter) y el Algoritmo.

Parámetros de la Ventana Principal (Main Window Parameters)

Para ir formando una idea más clara, reproducimos a continuación, en la Figura 1, el aspecto parcial del sector de Parámetros de la Ventana Principal con la que el usuario se encuentra.

Figura 1 – Sector Parámetros de la Ventana Principal

La columna *Voices*, en primer lugar desde la izquierda, indica qué instrumento es reproducido en cada canal numerado (voces) y también cuál de ellos está activado, es decir que genera sonido, o bien está desactivado. En el ejemplo de esta figura, la voz número 1 está activada y las otras dos no lo están. La elección del instrumento para cada voz queda a cargo del usuario a partir de las opciones típicamente estandarizadas en la lista de instrumentos midi, a la cual se accede en la ventana de Parámetros Extra (Extra Parameters) que emerge al oprimir sobre el botón *Set* que se puede apreciar en la última columna sobre la derecha, titulada *Extra* en la figura.

La columna *Offset*, inmediatamente a la derecha de la recién examinada, muestra un valor especificado por el usuario, para cada voz, que será multiplicado por el valor del Contador de Inicio de manera que si un valor *n* de la columna es negativo, el instrumento comenzará a ser ejecutado antes del momento de la reproducción en curso que se expresa en el valor del Contador de Inicio, siempre y cuando el Contador no esté en cero o el producto de la multiplicación no sea un número negativo, sucediendo en ambos casos que ninguna nota será producida hasta que el Contador "se ponga a la par" del valor de *Offset*. Mientras que en el caso de que el valor *n* sea positivo el mismo instrumento comenzará a producir sonido en un momento de la reproducción en curso posterior al representado por el valor del Contador de Inicio, siendo este momento un producto, aquí también, de la

multiplicación de los dos valores. Finalmente, si *n* es igual a cero, el instrumento comenzará a sonar en el momento correspondiente a la duración del pulso (*Note value*) asignada, en consonancia con el incremento del Contador de Inicio desde el valor cero. Todo esto quedará más claro cuando se vea en particular el Contador de Inicio.

De la columna *Modulate* emerge la ventana que puede verse en la Figura 2, si se tilda la casilla *Yes*.

Figura 2 – Ventana emergente de la opción Modulate

Mediante esta opción se enriquece la melodía fluctuando de una escala a otra durante la ejecución del instrumento. Si se selecciona la casilla *Every*, el valor que se introduzca, debiendo ser siempre un entero entre 1 y 50000, hará que la melodía module cada esa cantidad de notas. Si se selecciona la casilla *Random*, se calculará aleatoriamente el número de notas que deberán ser ejecutadas antes de que la melodía module. La modulación efectiva depende del valor generado por el algoritmo escogido para esa voz al momento de modular.

En el menú desplegable de la columna *Octaves* puede ajustarse el rango de la melodía desde 1 a 4 octavas. La conjunción del número de octavas con la nota *tónica* y la tonalidad elegidas en la columna *Note* y en la columna *Scale*, respectivamente, establece la escala de la melodía así como la nota más grave que se le permite ejecutar a la voz que se está definiendo (no sonará una nota más grave que la seleccionada en la columna *Note*). Puede escogerse una tonalidad de entre 15 distintas y la duración de las notas de entre 11 valores predefinidos, o bien es posible establecer escalas propias y valores rítmicos a gusto del usuario. En la Figura 3 se reproducen las listas desplegables de las columnas *Scale* y *Note Value*.

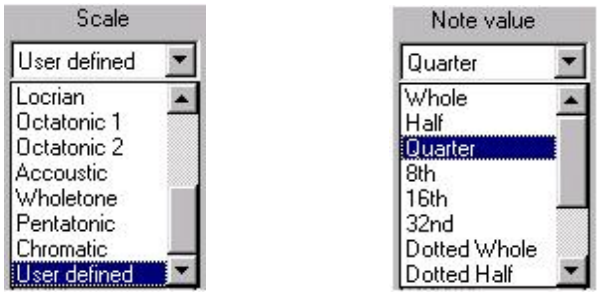

Figura 3 - Listas desplegables de las columnas Scale y Note Value

Al seleccionar la opción *User defined* en la columna *Scale* emerge la ventana que se reproduce en la Figura 4. El usuario define la tonalidad simplemente tildando las notas que desea incluir.

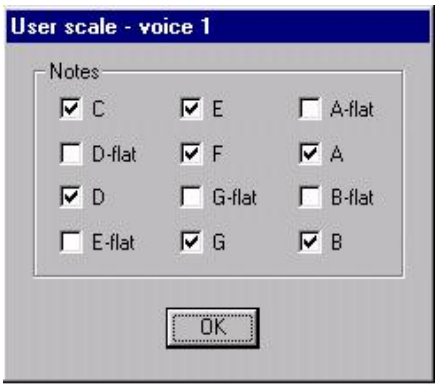

Figura 4 - Ventana emergente de la opción User defined en la columna Scale

Las escalas definidas por el usuario son siempre transpuestas automáticamente a Do (C) de manera que la tónica siempre es esa nota. Sin embargo, lo que se mantiene es, por supuesto, la relación interválica entre las alturas.

En este artículo, y para ajustar los términos que han empleado los diseñadores del programa, se ha establecido que donde dice *Note*, debe leerse **tónica**, donde dice *Scale*, debe leerse **modo** ó **tonalidad** (en algunos casos conviene traducirlo directamente como **escala**) y donde dice *Note value*, debe leerse **unidad de pulso**. Es así que, en la columna con ese nombre, el usuario puede elegir valores rítmicos desde redondas (Whole) hasta fusas (32^{nd}) los cuales pueden ser, también, tanto valores enteros como unidades con puntillo (Dotted).

Además, es posible definir otras unidades de pulso y subdivisiones si se elige la opción *Other* en el menú desplegable de la columna *Note value*.

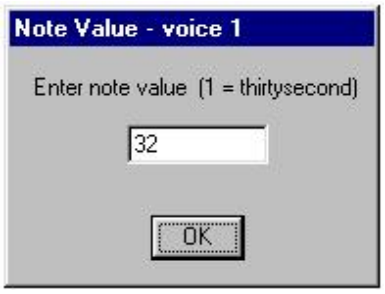

Figura 5 – Ventana emergente de la opción Other en la columna Note value

En ese caso, el usuario debe definir la unidad que desee midiéndola en cantidad de fusas. Por ejemplo, si uno quiere que la unidad sea equivalente a la duración de una negra ligada con una semicorchea, deberá escribir el valor 10 en el cuadro de texto de la ventana que se despliega al elegir la opción *Other*, lo cual indica las 8 fusas que caben en una negra

sumadas a las dos fusas que caben en una semicorchea. De la misma manera, en la Figura 5 el valor 32 significa que la unidad de pulso escogida coincide con una redonda (32 fusas).

Al seleccionar la opción *Set* en la columna titulada *Extra*, emerge la ventana que se reproduce en la Figura 6.

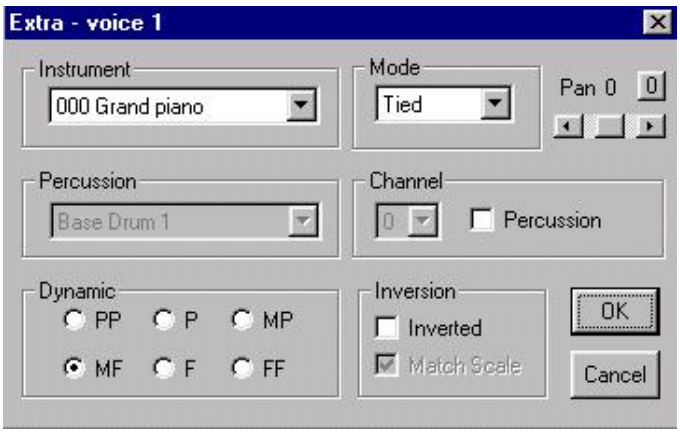

Figura 6 – Ventana emergente de la opción Set en la columna Extra

Los parámetros extra son ajustables individualmente para cada una de las 16 voces que soporta el programa. En esta ventana se selecciona de entre los típicos 128 instrumentos midi cuál será el ejecutado en la voz particular, siempre y cuando se desee producir un resultado melódico. Si el resultado buscado para esa voz es la producción de determinado ritmo, entonces el usuario deberá tildar la casilla *Percussion* para activar el canal que corresponde a esa función. De esta manera se activará el *combo box* para elegir el instrumento de percusión deseado. Se puede modificar el canal predeterminado por el cual se reproducirá la percusión (por defecto es el canal 9) en la lista desplegable titulada *Channel* para ajustar la configuración según la tarjeta de sonido disponible. En el área del marco titulado *Dynamic* se escoge la intensidad desde *pianissimo* (*PP*) hasta *fortissimo* (*FF*). La opción *Pan* permite graduar la salida del sonido ejecutado por la voz en cuestión a través de los parlantes.

Finalmente, las opciones *Mode* e *Inversion* producen cambios estructurales en la melodía. Eligiendo *Tied* en *Mode*, se ejecutará una nota sólo si es diferente de la anterior; escogiendo *All notes* se ejecutarán todas las notas correspondientes, sin importar cuántas veces se repita la misma nota en la secuencia. Mientras tanto, activar *Inversion* significará espejar la melodía, es decir que si el intervalo entre una nota y otra es de, por ejemplo, una tercera ascendente, será transformado en una tercera descendente. Esto, naturalmente, alterará la tonalidad, salvo que se tilde la opción *Match scale*, con lo cual se modificarán los intervalos de manera de mantener la tonalidad original.

Contadores (Counters)

Los Contadores funcionan como un control de iteraciones para los algoritmos. El Contador de Inicio (*Start counter*) establece tanto el momento en que los parámetros de la ventana principal comienzan a ser ejecutados como el punto en el que se origina un nuevo

Evento. Se incrementa secuencialmente mientras se ejecuta la música, y sus valores representan múltiplos, medidos en duraciones de fusa, de la unidad de pulso (*Note value*) seleccionada para la voz que se esté reproduciendo. Es decir que si la unidad de pulso escogida para la voz fuera de una nota blanca (16 fusas) el Contador de Inicio incrementaría sus valores según la secuencia 0, 16, 32, 48, 64, 80, etc.

Figura 7 – Sector de la Ventana Principal donde se ubican los Contadores

Ahora puede entenderse mejor el funcionamiento de la opción *Offset* que describíamos antes. El valor de la unidad de pulso, en cantidad de fusas, es multiplicado por el valor de *Offset*, y el resultado es sumado al valor actual del Contador de Inicio. Podemos también expresarlo matemáticamente de la siguiente manera:

nCI = P*O + aCI

entendiendo **nCI** como el nuevo Contador de Inicio, **P** como la unidad de pulso, **O** como el valor de *Offset* y **aCI** como el actual Contador de Inicio.

De esta manera, la opción *Offset* permite emplear técnicas contrapuntísticas como cánones y fugas. Además, el usuario puede saber el nombre y número de las notas que serán ejecutadas en cada momento de la secuencia del Contador de Inicio, utilizando la herramienta *Get Current Notes* que se encuentra en la barra de herramientas, por sobre la ventana principal (Main Window) anteriormente reproducida.

Por último, el Contador de Fin (*End counter*) indica el punto en que termina un Evento. Los Eventos son conjuntos de parámetros (voces activas, instrumentos, tempo, algoritmos, tonalidades, contadores, etc.) que son extraídos de los Parámetros de la Ventana Principal. Cada Evento es independiente de los demás y con ellos pueden crearse composiciones enteras de manera muy sencilla haciendo uso de la herramienta *Composition Maker*.

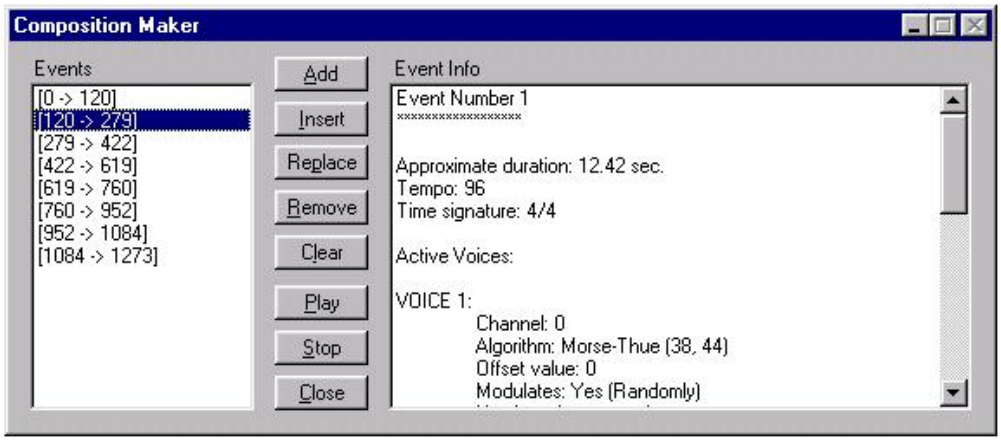

Figura 8 – Ventana de la herramienta Composition Maker

Algoritmos (Algorithms)

Echaremos ahora un vistazo a los diferentes Algoritmos de los que dispone este programa para generar las melodías, siguiendo el siguiente orden:

1. Morse-Thue Sequence
2. Earthworm Sequence
3. Wolfram one-dimensional Cellular Automata
4. 3n+1 numbers
5. Logistic Map
6. 1/f noise
7. Henon
8. Hopalong
9. Martin
10. Gingerbread man
11. Lorenz
12. Random

1. Morse-Thue Secuence: Es una secuencia de números que se genera a partir de dos parámetros: una *base* y un *multiplicador*.

1º El número que se muestra en el Contador de Inicio es multiplicado por el *multiplicador*.
2º El resultado es convertido en la *base* elegida.
3º Los dígitos que conforman el número que resultó de los pasos anteriores son sumados.

Ejemplo:

base 3 multiplicador 3 (contador 17) = 3*17 = 51 → 3 = 1220
1+2+2+0 = 5

Una secuencia generada de esta manera tiene la propiedad de que si le son quitados los números pares se vuelve a obtener la misma secuencia. Es esta autosimilitud la que le da carácter fractal.

2. Earthworm Sequence: Esta secuencia de números es generada a partir de tres parámetros: un *valor inicial*, un *multiplicador* y un *valor máximo de dígitos*.

1º Se multiplica el valor inicial **A** por el multiplicador constante **B**.
2º El resultado vuelve a ser multiplicado por **B** sucesivamente hasta que la cantidad de sus dígitos supere el valor máximo de dígitos **C**.
3º Se parte el resultado de manera que sólo posea la cantidad **C** de derecha a izquierda.
4º Se continúa a partir del paso **2º**.

Ejemplo:

Siendo **A = 2, B = 3 y C = 2;**
Entonces → **6, 18, 54, 62** (el resultado **162** fue partido quitándole el **1**) **86, 58, 74, 22, 66, 98, 94, 82, 46, 38, 14, 42, 26, 78, 34, 2, 6, 18...**

La secuencia resultante es un ciclo infinito de valores que se repiten luego de un período de determinado rango. Con los valores de este ejemplo se genera un ciclo de 20 valores diferentes pero, teóricamente, pueden producirse otros de miles de valores de largo. Nótese cómo un procedimiento tan simple como este permite producir una gran complejidad, y cómo es posible aprovechar esa complejidad en la creación musical.

3. Wolfram one-dimensional Cellular Automata: Fue el matemático Stephen Wolfram quien desarrolló esta versión unidimensional de Autómata Celular en base al original de John von Neumann[1]. Aquí, el universo es un *torus*, la vecindad es de tres celdas en línea y los estados que ellas pueden asumir son dos: *viva* (Alive) y *muerta* (Dead). Partiendo de una configuración inicial de los estados de las celdas que conforman la primera generación, se derivan las líneas de celdas subsiguientes (una generación debajo de la otra) en base a las reglas de supervivencia que se hayan predeterminado. Ya que los estados que pueden adoptar las celdas son 2 y las combinatorias posibles de los estados de las tres celdas por generación es de 8, desde todas vivas (AAA) hasta todas muertas (DDD) el número total de reglas que pueden seleccionarse es de 256 (2 a la 8ª potencia). Estas reglas establecen la configuración que deben adoptar las celdas de la generación anterior para que la celda de la nueva generación subsista.

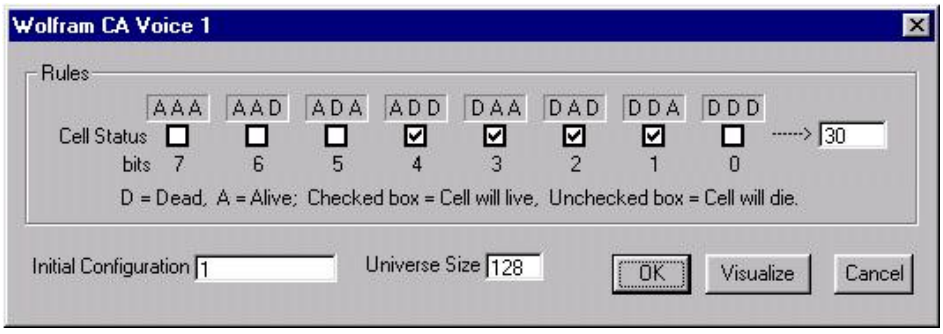

Figura 9 – Parámetros para el Autómata Celular de Wolfram

[1] Para una aproximación más profunda a la temática de los autómatas celulares y de otros algoritmos de la complejidad, desde una perspectiva antropológica, véase Reynoso (2006a).

La *configuración inicial* es un número entre 1 y 1410065408 que representa el estado inicial de las celdas en la generación 1, codificado como un entero de 32 bits. El *tamaño de universo* es el número de celdas encadenadas que conforman el autómata. Este universo es un *torus*, de manera que la última celda se conecta con la primera, y los valores permitidos para definirlo se encuentran dentro del rango [128, 512].

4. 3n+1 numbers:

1º Se toma cualquier número entero mayor que 1.
2º Si el número es par se lo divide por 2 y si es impar se lo multiplica por 3 y se le suma1.
3º Si el resultado es un número distinto de 1 se continúa a partir del paso 2º y si es igual a 1 se repite el ciclo generado hasta el momento.

Ejemplo:

Siendo A = 27;

Entonces → 27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1074, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Como se puede ver, la secuencia que comienza con 27 recorre 112 números hasta llegar a 1 y ninguno se repite en el trayecto. En teoría, todos los números finalmente convergen en 1 luego de un número finito de iteraciones. Considérese la posible longitud de otras secuencias teniéndose en cuenta que la que se origina con el número 27 es una de las más cortas, y así se entenderá la potencia de estos algoritmos cuando se los asocia con la producción de música. Formalmente son muy simples, mientras que nos brindan una complejidad enorme.

5. Logistic Map: Una de las más conocidas ecuaciones no-lineales y un buen ejemplo del caos determinista.

Está delimitada de la siguiente manera:

$$X(n+1) = X(n)*\mu*(1-X(n))$$

siendo **μ** cualquier número real dentro del intervalo [0, 4]. La iteración de esta fórmula devuelve valores entre [0, 1].

1º Se itera la fórmula acorde a la secuencia de incremento de los valores del Contador de Inicio.
2º El resultado (un valor entre 0 y 1) se multiplica por el número de notas de la escala o tonalidad y por el número de octavas, ambos parámetros elegidos por el usuario.
3º Se aplican estas modificaciones a la nota inicial y así se obtiene la nueva nota que será ejecutada.

Ya que esta secuencia de valores es determinista, para cada valor de **μ** siempre se conseguirá la misma secuencia de valores y, por lo tanto, las mismas melodías. Los valores de **μ** menores a 3.5 producen secuencias que se estabilizan en un solo valor al cabo de pocas iteraciones, mientras que a lo largo del resto del recorrido hasta **μ = 4** se produce el fenómeno comúnmente conocido como *duplicación del período*, de manera que las secuencias encuentran soluciones de entre distintos valores posibles: comenzando por 2 valores posibles, la secuencia aumenta a razón de 4, 8, 16... y así en más. En medio de períodos de caos se siguen encontrando secciones donde el comportamiento es estable, hasta que se arriba a la completa aleatoriedad del valor **μ = 4**.

6. 1/f noise: Si bien el valor de **μ = 4** es un camino para experimentar lo que se llama *ruido blanco*, este algoritmo lo utiliza para producir *ruido rosa*, lo cual no es sino una secuencia reconocible como música (Reynoso; 2006b: 352-355). Sin embargo, el comportamiento de la secuencia se puede tornar un poco más asemejable al movimiento browniano hacia los valores extremos de la variable **þ**. Como quiera que sea, el ruido de distribución 1/f es simulado con la siguiente ecuación[2] no-lineal:

X(n+1) = (X(n)*þ)+(sqr(1-þ^2)*r)

siendo **þ** un número real entre [0, 1] y **r** cualquier valor aleatorio. El parámetro **þ** debe ser especificado por el usuario, mientras que el valor de **r** se define a través del empleo del algoritmo explicado inmediatamente antes que este (Mapa Logístico) empleando el valor de **μ = 4**, el cual genera números aleatorios pero determinísticos. Si no fuera así, la secuencia nunca sería predecible para cada parámetro determinado.

Finalmente, aquí también el Contador de Inicio funciona como un control de iteraciones.

7. Henon: Este atractor es un mapa caótico bidimensional que se define por las siguientes ecuaciones:

X(n+1) = 1 + Y(n) - a*X(n)^2
Y(n+1) = b*X(n)

El usuario elige los valores de los parámetros X e Y dentro del rango [1, 2].
Cualquier corte transversal que se hiciera en un brazo de este mapa daría como resultado un conjunto de Cantor, lo cual evidencia sus propiedades fractales.

[2] Se transcribe como figura en la sección Ayuda del programa, por posibles impericias a la hora de intentar una traducción.

Figura 10 – Atractor de Henon

8. Hopalong: Este es otro mapa caótico bidimensional y se define por las siguientes ecuaciones[3]:

$$X(n+1) = Y(n) - sign(X(n))*sqrt(|sin(a)*X(n)-cos(a)|)$$
$$Y(n+1) = a - X(n)$$

El único parámetro que decide el usuario es el valor del ángulo **a** expresado en radianes[4] dentro del rango [-2π, 2π] es decir, entre -6.28319 y 6.28319.

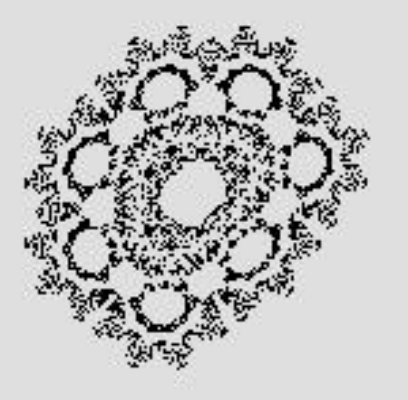

Figura 11 – Atractor Hopalong

9. Martin: Mapa caótico bidimensional delimitado por las siguientes ecuaciones[5]:

$$X(n+1) = Y(n) - sin(X(n))$$
$$Y(n+1) = a - X(n)$$

Aquí también el único parámetro que debe ser decidido es el valor del ángulo **a** según las mismas especificaciones del caso anterior.

[3] Ídem nota 2.
[4] El radián es una unidad utilizada en Geometría para la medida de ángulos. Es igual a la longitud de un arco que mide lo mismo que el radio del círculo de ese arco.
[5] Ídem. Nota 2.

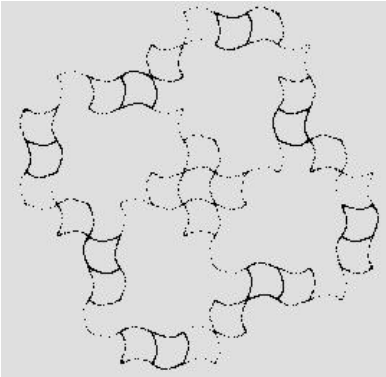

Figura 12 – Atractor de Martin

10. Gingerbread man: Este mapa caótico bidimensional se define por las siguientes ecuaciones:

$$X(n+1) = 1 - Y(n) + |X(n)|$$
$$Y(n+1) = X(n)$$

El usuario establece los valores iniciales de X y de Y en el rango [-20, 20].

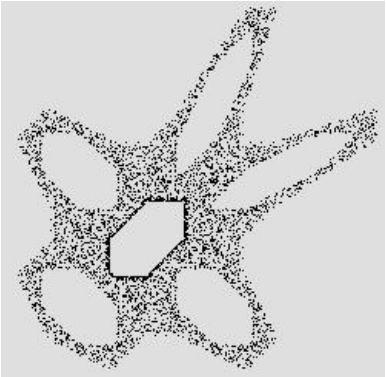

Figura 13 – Atractor Gingerbread man

11. Lorenz: Algoritmo basado en la famosa ecuación propuesta por Edward Lorenz, reproducida a continuación:

$$X(n+1) = a(3X(n) - 4X(n)^3)$$

El usuario decide el valor del coeficiente **a** dentro del rango [0, 1].

12. Random: Finalmente, nos encontramos con este algoritmo que no es otra cosa que un generador de números pseudo-aleatorios. Las melodías producidas de esta manera no muestran patrones discernibles y parecen siempre impredecibles, variando cada una de las veces que el usuario echa a correr el algoritmo, incluso en la ventana de visualización previa de la que dispone este software.

Debe seleccionarse dentro de qué rango se generarán los números, definiendo un valor *mínimo* y un valor *máximo* según las siguientes especificaciones:

- Tanto el valor mínimo como el valor máximo deben ubicarse dentro del rango [0, 1]
- Obviamente, el valor *mínimo* debe ser menor al valor *máximo*

Computación de la Música

Habiendo descrito cada uno de los elementos, procedemos inmediatamente a considerar sus interacciones, a partir de las cuales se generan las secuencias musicales.

Los Parámetros de la Ventana Principal son insertados en las funciones algorítmicas, cuyas iteraciones se expresan en los valores del Contador de Inicio. En la sección Ayuda del programa se brinda un prototipo de estas funciones y aquí se reproduce:

int (*(Algorithm[*n*]))(int , long , long , const int* ,int ,long , long , int ,BOOL , BOOL , int, BOOL, float, float, float, float, float,float, float, unsigned long, unsigned long, unsigned, double, double, BYTE, int, long)

siendo ***n*** el número del algoritmo. La función del algoritmo devuelve un entero que es la nota midi generada según los parámetros elegidos, la cual será ejecutada de acuerdo a la relación que se explicó anteriormente entre el Contador de Inicio (**CI**) y la unidad de pulso (**P**). Este procedimiento se realiza en tiempo real para cada voz de la composición.

Algunos comentarios sobre la experiencia de uso

Desde mi primera incursión en este tipo de programas, a través de **MusicLab I - Music from Chaos**, reconocí la importancia de que la interfaz con el usuario fuese muy explícita al respecto de la relación entre los algoritmos y atractores y los instrumentos musicales, de modo que se entendiera claramente cómo es posible ejemplificar estos principios matemáticos por medio de secuencias melódicas. En ese programa en especial, este requisito me pareció más que satisfecho. Su autor, David T. Strohbeen, publicó la versión 3.0, con la que yo comencé a experimentar, en 1997. En esa versión se mostraba una pantalla principal donde se graficaba en un eje de coordenadas la función del atractor que uno elegía y, a su izquierda, una lista de voces por las cuales se ejecutaban los instrumentos seleccionados. Cada una de esas voces tenía asociado un color, de manera tal que al activar una de ellas aparecía en el área del atractor representado, un pequeño cuadrado del color correspondiente a la voz en cuestión, simbolizando físicamente al instrumento de esa voz. Así, uno podía arrastrar el cuadradito hacia la zona del atractor que uno quisiese, lo cual no era otra cosa que asignar manualmente al instrumento el motivo matemático que generaría la secuencia melódica. Resultaba, entonces, en una manera muy didáctica de aplicar lo poco que yo sabía de atractores a lo poco que sabía de música.

Sin haber incrementado mucho ninguno de esos saberes, me encontré con la versión de **FractMus 2000**. La total carencia de gráficos en la ventana principal, atestada en cambio de botones y palabras, me retrajo un poco de comenzar a explorarlo. Finalmente, cuando venció el período de prueba de mi versión Shareware de **MusicLab I**, lo hice.

Los demás programas que tenía en ese momento a disposición no eran tan completos. Para dar una idea, mientras que **Fractal Music 1.9** computa música sólo utilizando autómatas celulares, **MusiNum 2.08 Beta-The Music in the Numbers** emplea únicamente el tipo de secuencias que aquí explicamos bajo el título *Morse-Thue*. Probablemente **a Musical Generator 3.1 (Beta 1)** sea, de los que conozco, el software con mayor número de herramientas a disposición y con mayores probabilidades de ser sucedido por nuevas versiones actualizadas. Sin embargo, encontré más comodidad en el manejo de la interfaz de **FractMus 2000** y, desde entonces, hace ya más de un año y medio, me dediqué a investigarlo exhaustivamente. Esta misma dedicación seguramente ha implicado que dejara de lado, por el momento, otras exploraciones. Para dar un ejemplo, me gustaría nombrar, de entre muchos otros programas que existen, la versión de **Fractal Tune Smithy 2.2**, la cual, por lo poco que pude apreciar, incluye numerosas prestaciones de interés.

Uno de mis intereses originales era el de poder entender mejor el funcionamiento y las propiedades de algunos conceptos matemáticos que recién comenzaba a vislumbrar (dinámicas no-lineales, atractores, algoritmos complejos, geometría fractal, etc.) para enriquecer mis herramientas metodológicas. Siempre he disfrutado de la música y, casi inmediatamente, me pareció un terreno sumamente apropiado para observar estos principios matemáticos en acción. De todas maneras, no sólo presté atención a cómo funcionaba lógicamente el sistema para la consecuente producción musical, sino que también lo utilicé con fines propiamente estéticos, como medio de creación. En poco tiempo, descubrí que la antropología de la música me estaba interesando de manera especial.

Dejé sin mencionar, hasta ahora, algunas herramientas útiles que completan el repertorio. Se pueden agregar comentarios a las composiciones, predefinir eventos para utilizarlos en distintas oportunidades, crear composiciones aleatoriamente, editar simultáneamente múltiples voces, obtener una lista con información sobre la composición, transponer automáticamente voces por separado y eventos enteros, redondear automáticamente los contadores para que no queden accidentalmente eventos mal ensamblados, traducir la composición a una imagen, proteger las composiciones con una clave de acceso, leer y escribir archivos en formato midi, agregar al nuevo archivo midi creado información sobre los algoritmos empleados y hasta abrir la calculadora estándar de Windows desde la propia interfaz.

De todas ellas, la opción para editar simultáneamente múltiples voces es especialmente útil, ahorrando mucho trabajo en la creación de las composiciones. Por su parte, la opción para traducir la composición a una imagen me parece de sumo interés teórico y le dedicaré unas líneas más adelante en este artículo. Podría pensarse, en un principio, que el juicio que nos merece la imagen puede corresponder con el que nos suscita la misma melodía. Estimo, por mi propia experiencia personal, que esto es posiblemente más cierto para consideraciones estructurales que para apreciaciones estéticas. En este programa, al menos, la imagen nos aproxima a una más clara percepción de la estructura constitutiva de la melodía debido a que ayuda a intuir cierto isomorfismo. Pero más de una vez he observado que un cambio notoriamente más valorable estéticamente en la melodía correspondía a un cambio notoriamente menos valorable estéticamente en la imagen. Las pruebas que puedo aportar para sostener esta sospecha son, en el mejor de los casos, altamente subjetivas, en base a estadísticas improvisadas sobre un número de casos suficientemente dudoso. No he podido todavía resolver algunos problemas técnicos para

someter tales imágenes al análisis de su dimensión fractal, lo que probablemente conduciría a una mejor fundamentada reflexión para decidir el asunto.

Para finalizar este apartado, me gustaría comentar que me parece preferible un programa que genere composiciones en base a eventos definidos por el usuario con contadores numéricos a uno que lo haga mediante una grabación en tiempo real o bien por simple selección y modificación de piezas ya predefinidas, grabando la secuencia resultante *a posteriori*. Dentro del primer conjunto encontramos programas como **MusiNum 2.08 Beta**, **a Musical Generator 3.1 (Beta 1)** y, por supuesto, **FractMus 2000**. Dentro del segundo conjunto encontramos programas como **MusicLab I** y **Fractal Tune Smithy 2.2**. En el tercer conjunto se incluye **Fractal Music 1.9**.

El motivo de la preferencia es que cuando el usuario, como ocurre con el primer conjunto de programas referido, tiene a disposición un espacio para probar primero cómo suena un algoritmo determinado con distintos parámetros y, luego, otro espacio distinto donde define claramente qué partes de las melodías generadas prefiere preservar y cuáles desechar, mediante el uso de unidades discretas que conoce de antemano, entonces puede realmente aprovechar la novedad que este tipo de programas en general se esfuerza en presentar, es decir, la generación de sonido reconocible como música y, posiblemente incluso de cierta calidad estética, en base a *procedimientos que no dependen directamente, en última instancia, de la voluntad del compositor*[6]. En cambio, si se está experimentando el sonido que produce cualquier algoritmo, interpretando a través de él parámetros que también vamos modificando simultáneamente, y todo esto al mismo tiempo que estamos grabando, es fácil reconocer que no vamos a tener la misma oportunidad de desechar o preservar las partes de la secuencia generada exactamente como nosotros lo desearíamos. Si, finalmente, tampoco se nos permite más que sólo modificar unos cuantos valores de eventos ya predefinidos de antemano, se quita todavía más capacidad de decisión al compositor y, según creo, se reducen también las combinatorias posibles para la generación de melodías: el *grado* de complejidad de las piezas no necesariamente se restringiría, pero sí el número posible de melodías distinguiblemente creativas.

La antropología de la música[7] y la búsqueda de *universales*

Ciclo de simulación

Las relaciones que pueden encontrarse entre la música y la matemática no son, por supuesto, ninguna novedad. Ahora bien, lo que se está explorando hace unos años y efectivamente constituye una innovación, es la posibilidad de delegar en el funcionamiento de una computadora ciertos procedimientos formales básicos que parecen intervenir en la producción cultural espontánea de la música.

Como sabemos, desde los trabajos de Richard Voss (1989) y en colaboración con John Clarke (1975 y 1978), aquello que, al menos occidentalmente[8], se concibe como **música** presenta, independientemente del contexto cultural en que se genera, propiedades fractales. Y si, además, estamos de acuerdo en que todo algoritmo es una sucesión finita de

[6] Como explicaremos, un poco más adelante.
[7] Para contextualizar al lector en elementos de juicio que aquí puedan darse por sentado, remito a Reynoso (2006b).
[8] Lo expreso de esta manera para mantenerme separado de algún tipo de nominalismo.

pasos orientada a la consecución de determinado fin u objetivo, entonces podemos pensar que la producción musical bien puede ser representada algorítmicamente, dado que cualquier pieza musical es una serie temporal de eventos, pasible de ser generada y vuelta a generar la cantidad de veces que se desee, dependiendo la fidelidad de cada generación de la fidelidad hacia los eventos constituyentes y sus parámetros, lo cual nos recuerda el problema de la entropía. Si es posible generar algorítmicamente algo parecido a *ruido rosa* por medio de unos simples dados, lápiz y papel, como demuestra Michael Bulmer (2000) con fines didácticos, también es posible que determinados programas de computadora, como los que revisamos en este artículo, ejecuten esos y muchos otros algoritmos afines por nosotros. En base a la descripción que hicimos de su funcionamiento, es fácil reconocer a programas como **FractMus 2000** como ejemplos de la "concepción paramétrica" de la música que propone Jean-Jacques Nattiez (2004).

Por lo visto, la implementación computacional se encuentra en condiciones de generar resultados musicales comparables a los que pueden generarse por medios de composición más convencionales. Si sabemos, por ejemplo, que una secuencia numérica estudiada dentro del marco de la Teoría de Números posee propiedades fractales, y además sabemos que la música espontáneamente producida en contextos culturales diversos también posee tales propiedades, podemos suponer que el algoritmo empleado para generar la secuencia numérica sea uno de los tantos que cualquier población humana tiene la potencialidad de aplicar, lo sepa o no, para producir música. Al asignar notas musicales a la secuencia numérica en cuestión, sin olvidar que lo que realmente presenta propiedades fractales es "(…) la incidencia de los intervalos de frecuencia, o de los cambios de la frecuencia acústica (…)", como señalan Kenneth Hsü y Andrew Hsü (1991) sometemos a prueba aquella suposición a través de una simulación por computadora. El mismo intento de simular el fenómeno nos enfrenta a una multiplicidad de disyuntivas que deberemos resolver tomando las decisiones teóricas y metodológicas que nos parezcan eventualmente más pertinentes, lo cual nos ayuda a avanzar en una mejor delimitación y comprensión de lo que deseamos estudiar. A este intento podemos llamarlo ***ciclo de simulación***, el cual seguiría los criterios de "convergencia empírica" planteados por Jorge Miceli (2006).

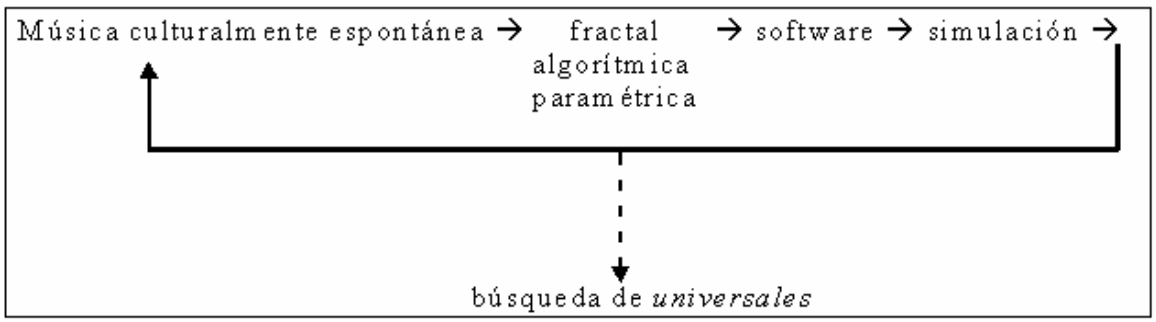

Figura 14 – Ciclo de simulación

Hago algunas aclaraciones del cuadro precedente:

1. Lo que estoy denominando "música culturalmente espontánea" no es otra cosa que la música producida de manera cotidiana dentro de algún contexto cultural y sin arreglo, por parte de los sujetos performadores, a fines de investigación.

2. Se espera que la lista de “propiedades” que se encuentra en el medio sea ampliada y corregida en base a las consideraciones de “convergencia empírica” antes referidas, como pasos necesarios en la búsqueda de *universales*.
3. Por supuesto, y en consonancia con el punto anterior, se espera también que el software de simulación vaya enriqueciéndose en el camino.

Ciclo de traducción

A través de diferentes programas de computación, del estilo de **FractMus 2000**, **a Musical Generator 3.1 (Beta 1)**, **Visions of Chaos 41.2** y **Visual Recurrence Analysis**, por nombrar unos pocos, es posible transformar porciones de información de un *soporte* a otro. Para ilustrar a qué me refiero, supongamos que un programa traduce una pieza musical a una imagen, como lo hace el último nombrado: consideraré que se ha traducido un *soporte sonoro* a un *soporte gráfico* o *geométrico*. Recordamos que también **FractMus 2000** traduce composiciones en imágenes y, entonces, comenzamos a pensar en la riqueza que tendría cotejar ambos *soportes gráficos*.

Desde una aproximación cibernética al problema de la transformación, similar a la planteada por Ross Ashby (1972), propongo que es posible hablar de cierta “conservación de la información” más allá de los cambios en su forma. Es decir que podríamos construir *cadenas de transformación* en las cuales un determinado *imput*, o *soporte* de entrada, fuera traducido en un *output*, o *soporte* de salida, por medio de un proceso de cambio ejecutado por determinado programa de computación. Si, prosiguiendo, uniéramos varias de esas cadenas, ejecutando sucesivos procesos de cambio a través de diversos programas que tomaran un *output* como *imput*, de manera que el *output* final de tal cadena mayor fuera el mismo *imput* inicial, lo llamaríamos ***ciclo de traducción*** y nos permitiría tener un panorama más amplio de lo que constituye la identidad del fenómeno observado y, por lo tanto, contar con criterios más claros para establecer comparaciones.

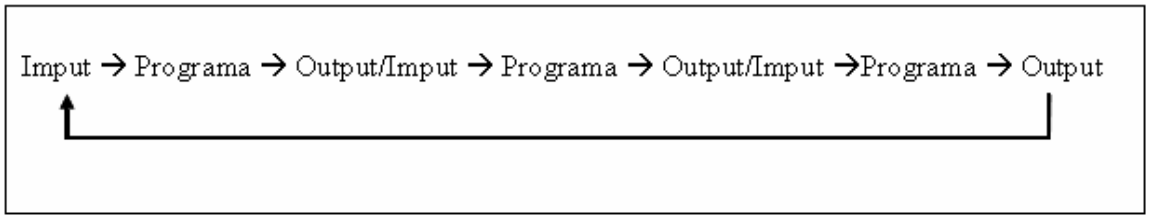

Figura 15 – Ciclo de traducción

Para ampliar el ejemplo anterior, supongamos que genero una composición con **FractMus 2000** en formato MIDI y, luego de convertirlo en formato WAV, lo abro con **Visual Recurrence Analysis**. En ese caso, obtendré un gráfico de recurrencia (recurrence plot) a partir de una serie temporal sonora. Pero si, por otro lado, a la melodía del archivo MIDI original la represento en un pentagrama, mediante programas como **Cakewalk Sonar**, para después ejecutarla con instrumentos convencionales y, finalmente, grabar la ejecución convirtiéndola en un archivo en formato WAV que fuera abierto con **Visual Recurrence Analysis**, entonces me vería tentado a realizar un análisis comparativo entre ambos gráficos de recurrencia. El primero de ellos es un *soporte gráfico* resultante de la

transformación de un *soporte sonoro* generado con un programa de música fractal-algorítmica; el segundo es, asimismo, un *soporte gráfico* resultante de la transformación de un *soporte sonoro*, pero esta vez generado por una emulación de lo que anteriormente denominé "música culturalmente espontánea": no se ajusta estrictamente a mi definición porque fue performada con arreglo a fines de investigación. Dejaría de ser una emulación, en ese sentido, si pudiéramos realmente lograr, de manera reconocible, una melodía musical "culturalmente espontánea" a través del empleo de programas de música fractal-algorítmica. Ese es el objetivo principal de lo que llamé ***ciclo de simulación***.

Considero de cierto interés pensar en construir un ***ciclo de traducción*** que incluya varias *cadenas de transformación*, representando el fenómeno estudiado en diferentes *soportes*. Así, una pieza musical sería entendida como una *configuración* pasible de ser traducida, con cierto isomorfismo, en gráficos, textos o números, por poner ejemplos concretos de los programas nombrados, para volver a convertirla en sonido. Algo parecido a esto ha realizado Roger DuBois (2003) con sistemas-L.

A lo largo de un ***ciclo de traducción*** esperaremos que pueda existir, como nos muestra la física en el caso de la transformación de la energía, una porción de "ruido" o entropía, lo cual no ocurre al utilizar solo un programa, es decir, en lo que estamos llamando *cadenas de transformación*. Pero más allá de esas pérdidas subsistirán rasgos estructurales, diferencias discretas que nos puedan ayudar a concebir comparaciones entre piezas musicales de manera algorítmica y paramétrica. Este trabajo se vería facilitado si todo el ***ciclo de traducción*** fuera realizado por un solo programa, suprimiendo los efectos entrópicos[9].

Al referirme a una "conservación de la información", no creo estar haciendo otra cosa que extender algunas ideas propuestas por Gregory Bateson. En *El temor de los ángeles* (Bateson y Bateson; 2004: 30) se define el alcance de dos de sus nociones, a mi entender, más ricas:

> En suma, nos valdremos del término Pleroma de Jung para designar ese mundo inanimado descrito por la física que en sí mismo no contiene distinciones ni las hace, aunque, por supuesto, debemos hacer distinciones en nuestra descripción de ese mundo.
>
> En cambio, usaremos el término Creatura para designar ese mundo de explicación en el cual los fenómenos que se describen son fenómenos gobernados y determinados por la diferencia, la distinción y la información.
>
> Aunque existe un aparente dualismo en esta dicotomía de Creatura y Pleroma, es importante tener presente que esas dos esferas no están en modo alguno separadas o puedan separarse, salvo como niveles de descripción. Por un lado, todo lo de la Creatura existe dentro del Pleroma y por obra de este; el empleo del término Creatura afirma la presencia de ciertas características de organización y comunicación que no son ellas mismas materiales. Por otra parte, el conocimiento del Pleroma existe sólo en la Creatura. Podemos abordar ambas esferas sólo en combinación, nunca separadamente. Las leyes de la física y de la química no son en modo alguno irrelevantes para la Creatura -esas leyes continúan aplicándose-, sólo que no son suficientes para la explicación, de manera que Creatura y Pleroma no son, como "el espíritu" y la "materia" de Descartes, sustancias separadas, pues los procesos mentales exigen disposiciones de la materia para darse, exigen zonas en las que el Pleroma está caracterizado por la organización que lo hace susceptible de ser afectado por la información así como por sucesos físicos.

[9] Lo que aquí sostengo es que los criterios de *traducción* empleados por los diferentes programas no están normalmente adecuados unos a otros, desvirtuándose la *configuración*.

Aunque ambas esferas sólo son separables como niveles de descripción, el Pleroma posee ciertas regularidades inmanentes. Empleando otros términos batesonianos afines, podemos decir que la física constituye, ella misma, un *mapa* humano para describir ese *territorio*. Al decir que los procesos mentales necesitan de ciertas disposiciones de la materia para darse, estamos diciendo, según la física actual, que necesitan de energía para darse, como lo afirma el mismo Bateson cuando define qué es una mente (requiere energía colateral).

Si la energía, según la física, se conserva en su cantidad universal absoluta (no se crea ni se destruye) más allá de que pueda cambiar en su forma de manifestarse (cinética, térmica, lumínica, potencial, como materia, etc.) sostengo que la información conserva, no digamos su cantidad universal absoluta[10], pero sí la *configuración* de un sistema determinado, de una porción distinguible de la información factual total, más allá de que pueda cambiar en su forma de manifestación. A estas formas es a las que yo denomino, porque debo hacerlo de alguna manera, con el término *soportes*.

Las *traducciones* pueden ser más o menos analógicas o, incluso, arbitrarias. Pero algunas de ellas presentarán una mayor "convergencia empírica" que otras, y esta última sólo puede darse en relación a las *configuraciones* perceptibles en la interacción entre nuestro cerebro humano y las inmanencias pleromáticas. Concentrándonos en lo pleromático de toda información, nos es posible evitar el retórico tedio de sólo hacer *mapas* de otros *mapas*.

Ciclo de composición

Si estos programas, como **FractMus 2000**, fuesen utilizados para componer música sin arreglo a fines de investigación formal por parte de sus performadores, sino tan sólo por motivos estéticos, entonces podríamos considerar a la *música fractal-algorítmica* como algo parecido a un *género* innovador, como parece ser el caso entre los compositores que integran la lista de correos electrónicos de Yahoo conocida como **cnfractal**, de la cual soy también miembro hace más de un año. De manera que trato de jugar ambos juegos: el de investigador y el de compositor. En tal contexto, creo que la habitual denominación de *música fractal-algorítmica* incurre en una redundancia: como sostuvimos hasta aquí, toda música presenta una distribución fractal y puede ser pensada tanto algorítmicamente como paramétricamente. Por el momento, sin embargo, dudo que resolver la redundancia cambie algo más que el nombre.

Al comienzo de mi experiencia personal, me dediqué a generar piezas en formato MIDI con **FractMus 2000**. Pero, con el tiempo, al encontrar que iba manejando el programa de manera más fluida y rica, me surgió el interés de convertir esas piezas, algunas de las cuales me habían llevado incluso horas de trabajo, en un formato más estable y con instrumentos que sonaran más "profesionales", en el sentido de que, según la tarjeta de sonido de la computadora en la que se reproduzcan, las piezas MIDI pueden sonar como "música de videojuego", y ese no era mi propósito. Entonces utilicé programas como **Cakewalk Sonar**, a través de los cuales es posible transformar el archivo MIDI en, por ejemplo, un archivo WAV. A su vez, ese archivo WAV puede ser retocado de diversas maneras con programas como **Sony Sound Forge**, **Guitar Rig** y muchos otros. Así lo hice para que, finalmente, mis composiciones sonaran de manera similar a como lo haría si

[10] Ya que sabemos que la muerte térmica triunfará, más tarde o más temprano, destruyendo toda información.

hubieran sido ejecutadas con instrumentos convencionales. Ahora bien, justamente a este punto quisiera dedicarle unas líneas.

Se puede componer música por computadora de, al menos, dos maneras distintas:

- suplantando instrumentos convencionales (**Cakewalk Sonar**, etc.)
- además de ello, delegando el algoritmo (**FractMus 2000**, **a Musical Generator 3.1**, etc.)

siendo los programas incluidos en la segunda clase a los que nos queríamos referir especialmente en este artículo.

Lo que antes mencioné en alusión a la "voluntad del compositor" tiene directamente que ver con lo que ahora agrego sobre "delegar el algoritmo". Valiéndome de estas expresiones trato de explicar que, según lo entiendo, la estructura de la composición no se le ocurre al compositor de otra manera que no sea a partir de las "propuestas" que presenta tal o cual algoritmo cuando se lo "pone a correr". Esas "propuestas" no eran, en la gran mayoría de los casos, conocidas ni imaginadas de antemano, ya que no solemos estar al tanto de las propiedades de, por ejemplo, secuencias provenientes de la Teoría de Números. Incluso estando al tanto de ellas, la máquina computa esas secuencias a una velocidad que nunca podríamos equiparar y que tampoco tendría sentido que lo hiciéramos. Tomemos el caso del algoritmo *3n+1 numbers*. En un principio, ni siquiera estaba enterado de que existiera una secuencia numérica con las propiedades que ya repasamos. Pero descubrirlo no me llevó inmediatamente a saber el trayecto que se recorre desde el número 27, sino hasta que quise ponerlo como ejemplo aquí, debiendo primero calcularlo posición por posición en lápiz y papel. Mucho menos me llevó instantáneamente a saber qué melodía resultaba porque, entre otras cosas, no tenía idea de cómo la generaba el programa. En resumen, la forma más razonable de componer música de una vez por todas era "echar a correr" el algoritmo con ciertos parámetros que, si bien elegía positivamente de manera directa, seguiría sin saber cómo se conjugarían sonoramente hasta que la melodía comenzara efectivamente a salir por los parlantes.

Pues bien, todos estos desconocimientos no se ponen en juego al componer por otros medios. Si tomo mi guitarra tanto para ejecutar una melodía ya existente como para crear una nueva, soy directamente yo quien selecciona y genera audiblemente las notas. Si, en lugar de tomar mi guitarra, la suplanto con sonidos de guitarra genérica grabados en archivos que ejecuta un programa de computación, también soy directamente yo quien selecciona las notas, aunque sea el programa quien las genera audiblemente. Pero si empleo un programa de música fractal-algorítmica, es el algoritmo el que termina seleccionando, en última instancia, la sucesión de notas, incluso si ese programa es tan participativo y amigable como **FractMus 2000**. Es que para eso fueron diseñados: para generar sonido a partir de objetos y estructuras que tienen ciertas propiedades inherentes y, así, poner a prueba si esas propiedades constituyen ese sonido de manera que suene a música. Lo que denomino ***ciclo de simulación*** incluye tales pruebas, con objetivos como el de replicar famosas piezas musicales "culturalmente espontáneas".

Partiendo de mi propia experiencia, voy a proponer que se considere un último ***ciclo de composición***, según el cual se emplean los programas que a cada quien resulten más apropiados, siempre que éstos permitan, en primera instancia, generar una pieza de *música fractal-algorítmica*, en segunda instancia, retocarla de manera que suene lo más parecido posible a una pieza "profesional" y, en tercera instancia, representarla como una partitura

ejecutable por medio de instrumentos convencionales. No tengo noticia de que exista un programa que permita realizar las tres instancias sin que resulte necesario apelar a algún otro. Pero, aunque existiera, lo único que implica un ***ciclo de composición*** es que las tres instancias, a las cuales llamaremos respectivamente *generación*, *edición* y *transcripción*, efectivamente se realicen.

Entonces, mi ***ciclo de composición*** personal ha consistido, hasta la fecha, en la utilización de **FractMus 2000**, para realizar la primera instancia de *generación*, **Cakewalk Sonar**, **Sony Sound Forge** y **Guitar Rig** para realizar la segunda instancia de *edición* y nuevamente **Cakewalk Sonar** para realizar la tercera instancia de *transcripción*.

Palabras finales

Según era mi propósito argumentar en este artículo, considero que es, al menos, a través de estos tres *ciclos* que programas computacionales como los referidos prometen interesantes aportes a la búsqueda de *universales* en antropología de la música.

Describí el funcionamiento de uno de estos programas y mostré algunas formas de componer música fractal-algorítmica. De aquí en más, posiblemente también estemos hablando de un *género* musical que, aunque novedoso, pueda él mismo ser estudiado en términos culturales.

Bibliografía

Ashby, W. R. *Introducción a la cibernética*, Nueva Visión, Buenos Aires, 1972.

Bateson, G. y Bateson, M.C. *El temor de los ángeles. Epistemología de lo sagrado*. Gedisa. Colección El Mamífero Parlante. Barcelona, 2000 [1987].

Bulmer, Michael.. *Music from fractal noise*. Proceedings of the Mathematics 2000 Festival, Melbourne, 13 de enero de 2000.

DuBois, Roger Luke. *Applications of generative string-substitution systems in computer music*. Tesis de doctorado, Universidad de Columbia, 2003.

Hsü, Kenneth y Andrew Hsü. "Self-similarity of the '1/f noise' called music". Proceedings of the National Academy of Sciences U. S. A., 88: 3507-3509. 1991.

Miceli, J. *Pasos Generales para la Programación de un Modelo*. Material de Cátedra, Seminario de introducción a las sociedades artificiales y a los modelos computacionales en Antropología FFyL UBA 2006. www.antropocaos.com.ar/seminario.htm

Nattiez, Jean-Jacques. "Is the search for universals incompatible with the study of cultural specificity?". *John Blacking Memorial Lecture. XX European Seminar in Ethnomusicology*, Venecia, 2 de octubre de 2004. http://217.57.3.105/cinipdf/vari/blaking.pdf

Reynoso, C.

-2006a, *Teoría y Métodos de la Complejidad y el Caos: Una exploración antropológica*, Editorial SB Buenos Aires

-2006b, *Antropología de la Música: De los géneros tribales a la globalización* – Vol. 2, Teorías de la complejidad, Editorial SB Buenos Aires

Voss, Richard F. "Random fractals, self-affinity in noise, music, mountains, and clouds". Physica D 38: 362-371. 1989.

Voss, Richard F. y John Clarke. "1/f noise in music and speech". *Nature*, 248: 317-318. 1975.

Voss, Richard F. y John Clarke. 1978. "1/f noise in music: music from 1/f noise". *Journal of the Acoustical Society of America*, 63: 258-263.