Seminario de investigación:
Técnicas generativas para la producción audiovisual en entornos digitales

Joaquin Cerviño Tutor: Esteban Calcagno

1 de octubre de 2019

Índice

1. Introducción

El arte generativo comprende un corpus lo suficientemente amplio y variado como para provocar desde un primer momento una situación problemática al momento de definirse como disciplina. En el artículo "What is Generative Art?", Philip Galanter incurre en la tarea de encontrar los rasgos distintivos de esta rama del arte y establece, de una forma preliminar, su carácter interdisciplinario [7]. Galanter sostiene que el espectro de producciones que engloba incluye a la música, desde el punto vista del diseño sonoro y de la composición algorítmica; a la producción visual hecha con computadoras, aplicada a juegos o películas; a la *demoscene* y la cultura de *vjing*; al diseño industrial y la arquitectura.

1.1. Presentación del trabajo

En el proyecto de investigación se busca indagar sobre el proceso creativo del arte generativo combinando entornos digitales de audio como Pure Data [13] con visuales como Processing [9], a partir del estudio de tres algoritmos específicos: autómatas celulares, *L-systems* y algoritmos genéticos.

Para investigar la implementación de dichos algoritmos, se hizo un relevamiento de los desarrollos previos de éstos en aplicaciones de software relacionadas con la producción de imágines, con la composición algorítmica y con la secuenciación de sintetizadores.

1.2. Objetivo

El objetivo del trabajo es estudiar qué técnicas son y fueron empleadas para lograr, a partir del *output* de algoritmos, resultados audiovisuales estéticos que no dejen de lado la correlación del proceso generador con el sonido y la imagen obtenidos.

1.3. Metodología

En una primer instancia, se hará un relevamiento y análisis de las implementaciones realizadas investigando a partir de *papers* y otras publicaciones cómo y con qué fines se llevaron a cabo estos desarrollos previos de aplicaciones.

En una segunda instancia, se implementarán en los entornos de software previamente mencionados desarrollos preexistentes que hubieran aplicado los

algoritmos estudiados. Para tal fin se desarrollarán aplicaciones para generar un resulado audiovisual en tiempo real.

Para el estudio de los autómatas celulares, se escogió una implementación creada por Eduardo Reck Miranda, llamada *Chaosynth*. Miranda creó un sintetizador granular que utiliza dicha algoritmo como secuenciador [14].

En el caso de los *L-systems* y los algoritmos genéticos, se decidió realizar una implementación a partir de una publicación de Manousakis [12], que propone formas de utilizarlos, pero sin seguir un desarrollo de una aplicación puntual.

1.4. Conclusiones

De los estudios de caso indagados, y otros materiales investigados que también proponen formas de implemetación de algoritmos, se observó que el método principalmente utilizado para vincular el *output* de éstos con un resultado compositivo, sonoro o visual se puede denominar técnica de mapeo o *mapping technique*. Esta instancia correlaciona los valores entregados por el algoritmo con los materiales constitutivos de disciplinas artísticas, pudiendo tratarse de notas, sonidos, trazos o figuras geométricas mediante la intervención del factor humano dado por el compositor, artista o programador. Se puede señalar en este etapa el final de la "autonomía"del proceso de desarrollo del algoritmo, en su naturaleza procesual, en el cual existe el error en forma objetiva de acuerdo a instrucciones que determinan un procedimiento y el comienzo de otra fase donde la subjetividad estipula qué resultados desechar o preservar o en qué medida modificar las condiciones iniciales y las instrucciones del proceso autónomo para variar su resultado final.

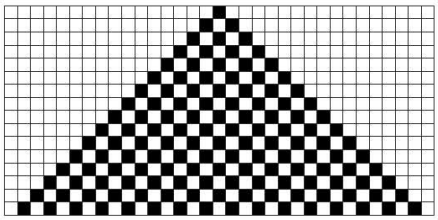
Las implicancias de la denominada *mapping technique* son muy amplias, ya que ésta es constitutiva en la utilzación de entornos como Processing, Pure Data, OpenFrameworks o similares. Los usuarios de dichos programas mapean valores numéricos que pueden ser deterministas, estocásticos o representar valores de una base de datos a procesos que los convierten en sonidos o imágenes. Esta técnica relaciona claramente estas prácticas con la visualización de datos y la sonificación de los mismos y a grandes rasgos se la puede señalar como el fundamento del arte digital, o generativo.

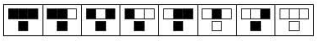
2. Autómata Celular

2.1. Introducción

Los autómatas celulares fueron concebidos por Stanislaw Ulam y John von Neumann a partir de análisis realizados sobre desarrollos morfológicos y caracterizados como sistemas complejos que pueden manifestar un comportamiento global o emergente. Coveney y Highfield definen a una propiedad emergente como "una propiedad de un sistema complejo que consiste de varias unidades que interactúan" [5]. Estos sistemas fueron objeto de estudio durante años, contando entre sus estudiosos a científicos como Konrad Zuse, Ed Fredkin y Stephen Wolfram. Este último hizo uno de los desarrollos y análisis más extensos sobre el tema.

Wolfram define a los autómatas celulares como "una línea de células, cada una coloreada de negro o blanco, representando dos estados posibles. El estado de cada célula se evalúa a cada paso, pudiendo permanecer en su estado previo o cambiar su condición, de acuerdo a una regla definida que toma por parámetros la condición de las dos células vecinas en el paso anterior" [24]. El autor destaca que los autómatas celulares son capaces de manifestar un comportamiento altamente complejo pese a que se comporten según una regla sencilla y hayan partido de un estado inicial simple.



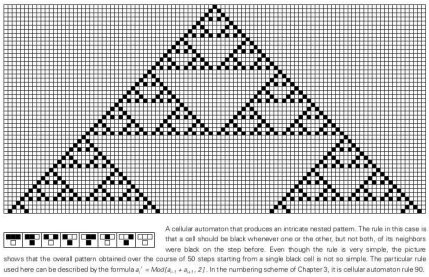


A cellular automaton with a slightly different rule. The rule makes a particular cell black if either of its neighbors was black on the step before, and makes the cell white if both its neighbors were white. Starting from a single black cell, this rule leads to a checkerboard pattern. In the numbering scheme of Chapter 3, this is cellular automaton rule 250.

Figura 1: Ejemplo de propagación de un autómata celular y su regla. [24]

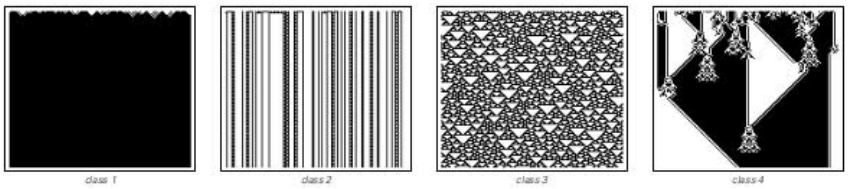
A partir de la observación realizada sobre los resultados de distintos tipos de autómatas celulares, es decir, basados en distintas reglas, y provenientes de estados iniciales aleatorios, Wolfram llega a la conclusión de que el comportamiento emergente de éstos puede ser clasificados en cuatro grupos. Éstos a su vez están ordenados según un grado creciente de complejidad.

El primer grupo es el que exhibe el comportamiento más simple y en éste se



A cellular automaton that produces an intricate nested pattern. The rule in this case is that a cell should be black whenever one or the other, but not both, of its neighbors were black on the step before. Even though the rule is very simple, the picture shows that the overall pattern obtained over the course of 50 steps starting from a single black cell is not so simple. The particular rule used here can be described by the formula $a_i' = Mod[a_{i-1} + a_{i+1}, 2]$. In the numbering scheme of Chapter 3, it is cellular automaton rule 90.

Figura 2: Ejemplo de propagación de un autómata celular y su regla [24]



Examples of the four basic classes of behavior seen in the evolution of cellular automata from random initial conditions. I first developed this classification in 1983.

Figura 3: Los cuatro grupos propuestos por Wolfram para clasificar los autómatas celulares [24]

incluye a los AC que desde cualquier punto inicial derivan en un mismo estado final uniforme. En el segundo grupo, hay varios estados finales pero todos estos permanecen inalterados o se repiten luego de un número reducido de pasos. En el tercer grupo el comportamiento es más complejo y es en su gran mayoría aleatorio, aunque puedan manifestarse pequeñas estructuras, como triángulos, de forma esporádica. En el cuarto grupo, se agrupa a los AC que muestran un comportamiento que resulta de una mezcla de aleatoriedad y orden. En esta categoría se puede observar la aparición de pequeñas estructuras que a su vez interactúan con otras de un forma compleja.

2.2. Estudio de caso

2.2.1. Introducción al estudio de caso

Una de los primeros casos de uso de autómatas celulares para la composición es la obra Horos de Xenakis, en la cual el compositor hizo uso de dichos algoritmos para determinar sucesiones de acordes [3]. Otras implementaciones

destacables de los AC para la composición algorítmica es el CAM (Celular Automata Music) desarrollado por Dale Millen en la Universidad de Arkansas y los programas CAMUS y CAMUS3D desarrollados por Eduardo Reck Miranda.

Una de las aplicaciones que vincula más claramente la representación visual y sonora del algoritmo en cuestión es el *Chaosynth* desarrollado por Eduardo Reck Miranda [14]. Dicho desarrollo utiliza los resultados o el *output* de un autómata celular para la secunciación de un sintetizador granular.

La síntesis granular es una técnica que fue conceptualizada por primera vez por Xenakis en su libro Musiques Formelles [26]. Las primeras implementaciónes se puedieron lograr con el advenimiento de la síntesis digital, siendo Curtis Roads [18] y Truax [22] los primeros en experimentar e implementar esta técnica. Posteriormente, Roads le otorgó un marco conceptual y académico que quedó plasmado en su libro Microsound [19]. Esta técnica consiste en la utilización de un gran número "granos"(sonidos de muy corta duración, por ejemplo, de hasta 50 milisegundos), que pueden tener variadas formas de onda y a los cuales se les aplica una envolvente. La masa generada como resultado de la sumatoria de granos tiene parámetros globales que la determinan, como por ejemplo, la densidad. En este aspecto Xenakis propone procedimientos estocásticos para modelar la morfología de estas nubes de granos. Eduardo Reck Miranda [14] sugiere la utilización de autómatas celulares para controlar la producción de granos en la síntesis granular a través del *Chaosynth*.

2.2.2. Chaosynth

ChaOs proviene de la expresión *Chemical Oscillator*, "una representación metafórica de un fenómeno neurofisiológico conocido como circuito de reverberación neuronal" [14]. En este caso, las células que componen al AC son interpretadas como células nerviosas que pueden manifestar tres estados: inactivo (*quiescent*), despolarizado (*depolarized*) o quemado (*burned state*). Este AC evoluciona de un estado incial de completa aleatoriedad a un comportamiento oscilatorio donde pueden discernirse patrones. Miranda plantea que esta característica emergente del sistema es similar a cómo se producen sonidos de forma natural en instrumentos acúsitcos ya que, al igual que el AC, "los sonidos tienden a converger de una amplia distribución de sus parciales (por ejemplo ruido) a patrones oscilatorios (por ejemplo, un sonido armónico)" [14]. En el Chaosynth la interacción entre células representa la circulación de corriente entre ellas. Hay dos umbrales con distintos valores, uno con un valor de voltaje mínimo($Vmin$) y

otro con un valor máximo($Vmax$). Si el valor del voltaje de la célula se encuentra por debajo del $Vmin$, la célula se encuentra en estado inactivo (o "polarizada"); si su valor está entre $Vmin$ inclusive y $Vmax$ está despolarizada. Cada célula "tiene un divisor de potencia (*potiential divider*) cuya finalidad es mantener el voltaje de la misma debajo de $Vmin$. Si el divisor falla, esto es, si Vi alcanza $Vmin$, la célula nerviosa se despolariza. También, está en juego un capacitor eléctrico que regula la velocidad de despolarización" [14]. Miranda destaca que la tendencia global del sistema es a que la células se despolaricen cada vez más con el transcurrir del tiempo, o a cada paso. Por último, se aclara que "una célula quemada en el tiempo (o paso t) es reemplazada por una nueva célula inactiva en el siguiente paso $t + 1$" [14].

A modo de resumen, se puede inferir que el comportamiento del ChaOs depende de los siguientes factores:

1. El número n de estados, siendo $n > 3$.
2. El valor de las resistencias $r1$ y $r2$ del divisor de potencia.
3. La capacitancia k del grado de depolarización.
4. La velocidad t del sistema.
5. La dimensión de la grilla.

El Chaosynth hace uso del comportamiento del sistema anteriormente descripto(ChaOs) para manejar la generación de un gran número de "partículas sonoras"para formar un evento sonoro complejo. Miranda hace uso de lo que denomina *Mapping Technique*(técnica de mapeo) para utilizar el ChaOs con el fin de secuenciar partículas sonoras, o "granos". Cada una de estas partículas "está compuesta por varios parciales. Cada parcial es una sinusoide generada por un oscilador. Un oscilador necesita tres parámetros para funcionar: frecuencia, amplitud y duración(en milisegundos). ChaOs controla los valores de frecuencia y la duración de la partícula, pero los valores de amplitud son establecidos por el usuario de forma previa. Los estados de una célula nerviosa están asociados a un valor de frecuencia y los osciladores están asociados a un cierto número de células. Los valores de frecuencia en un tiempo t son, por consiguiente, establecidos por el promedio de las frecuencias asociados a los estados de la célula nerviosa mapeada a cada oscilador" [14]. En el Chaosynth se permite al usuario establecer otros parámetros como por ejemplo, el tamaño de la grilla, la cantidad

de osciladores, la relación entre células nerviosas y osciladores y los parámetros internos del ChaOS (el número de estados, el valor de las resistencias, la capacitancia de las células). En cuanto a la síntesis de sonido se puede agregar que es conisderable también como aditiva, ya que cada partícula sonora o "grano" es generado por la unión de varias sinusoides.

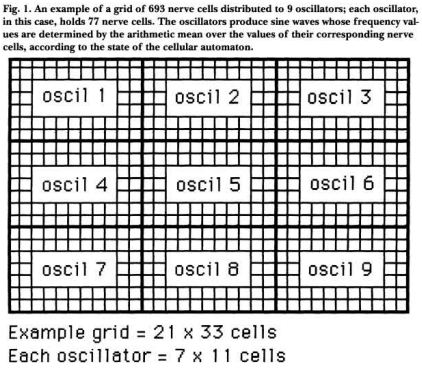


Figura 4: Técnica de mapeo. [14]

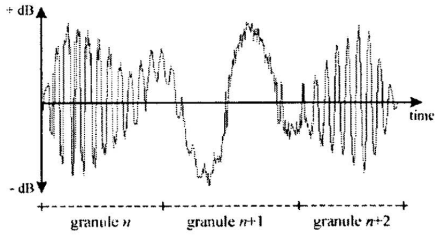


Figura 5: Resultado obtenido por el Chaosynth. [14]

2.2.3. Implementación

El Chaosyth se trata de un autómata celular de dos dimensiones. La representación varía en comparación a los expuestos en la introducción en que lo que se grafica es un estado actual en una grilla de células, en contraposición al de una serie de pasos consecutivos.

Para desarrollar el autómata celular, se decidió modificar una implementación del mismo hecha en el entorno Processing. La versión del Chaosynth realizada para el seminario está basada en la de Daniel Shiffman del *Game of Life* de John Conway. El juego desarrollado por el matemático Conway consiste

en un tablero con casilleros que pueden estar ocupados o no por fichas, que representan organismos "vivos", que se trasladan de acuerdo a "leyes genéticas". Esta clase de juegos se denominan de "simulación"(*Simulation game*) ya que exhiben un desarrollo análogo a aquellos de seres vivos. Las reglas estipuladas por el creador del juego fueron hechas para que satisfagan las siguientes tres normas: "1. No debe haber un patrón inicial[de células] mediante el cual pueda haber una propagación sin límite de células. 2. Debe haber parámetros iniciales que aparentemente crezcan sin límite. 3. Debe haber un patrón inicial simple que crezca y se transforme durante un tiempo considerable y llegue a tres finales posibles: desaparecer completamente (por superpoblación o escasez), permanecer en una configuración estable que permanece sin cambios, o entrar en una fase oscilatoria en la cual se repitan patrones de forma permanente cada dos o tres ciclos". [8] Shiffman interpreta que el *Game of Life* se trata de un autómata celular de clase 4 según la taxonomía de Wolfram. [20]. Como el Chaosynth representa metafóricamente a un oscilador químico en el cual células se despolarizan o queman, el Game of Life representa el desarrollo de organismos que nacen, mueren o sobreviven según el estado de las células vecinas: "1. Supervivencia. Cada ficha [célula] que contenga dos o tres células adyacentes sobrevive a la siguiente generación. 2. Muertes. Cada célula que posea cuatro o más vecinos muere por superpoblación. Cada, célula con un solo vecino o ninguno muere por su aislamiento. 3. Nacimientos. Cada célula muerta o casillero vacío con exactamente tres células adyacentes vivas nace, esto es, ese casillero posee una célula viva en la siguiente generación". [8] El Chaosynth es una extensión del Game of Life con otras reglas y con más estados posibles para las células.

La generación de sonido se logra mediante la utilización del programa Pure Data, desarrollado por Miller Puckette [13] y se vinculan los dos procesos mediante la utilización del protocolo OSC(*Open Sound Control*) [25]. Processing genera *frames* o imágenes de forma consecutiva y a una frecuencia estable para lograr la animación. Cada una de éstas es graficada luego de una instancia de procesamiento donde se calculan los valores venideros de cada célula del AC, esto es, los colores y las posiciones de los cuadrados en la pantalla que los representan. Es en esta instancia que también se calculan promedios de valores de secciones del tablero de células para utilizarlos como parámetros en la secuenciación del sintetizador. Para lograr este fin, son enviados utilizando el protocolo OSC.

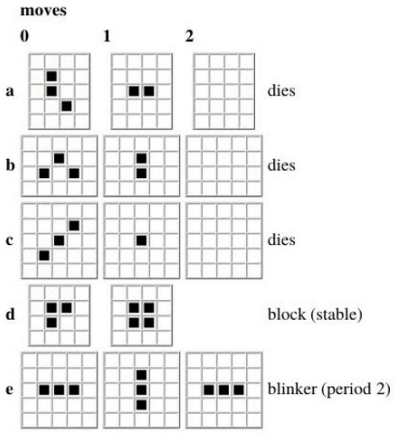


Figura 6: Ejemplo del funcionamiento del *Game of Life*. [8]

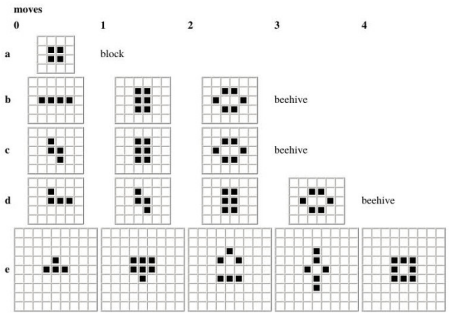


Figura 7: Patrones que emergen del *Game of Life*, como el "panal de abejas". [8]

3. Sistemas de Lindenmayer

3.1. Introducción

Los sistemas de Lindenmayer o *L-systems* son un teoría matemática concebida por el botánico Aristid Lindenmayer en 1968 [11]. El objetivo del desarrollo original era describir de forma topológica las relaciones entre el crecimiento de células y otras partes más grandes de plantas. Los aspectos geométricos y de repesentación de estos sistemas no fueron abordados por Lindenmayer. De todas formas, posteriormente fueron implementadas técnicas para lograr una representación gráfica de los mismos por Prusinkiweicsz [17], entre otros. El concepto más característico de los *L-systems* es el de la reescritura, esto es, generar un objeto complejo a partir de la sustitución de los elementos más simples que originalmente lo componen por otros más extensos, de forma iterada. Este proceso se lleva a cabo mediante *reglas de reescritura* que a su vez, en cada iteración, generan una *producción*. Un ejemplo de una representación gráfica lograda mediante la reescritura de sus partes es la "snowflake curve"de von Koch.

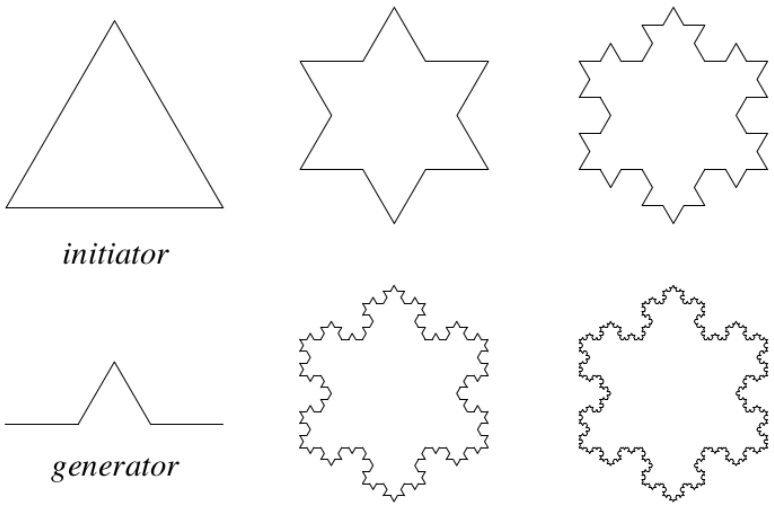


Figure 1.1: Construction of the snowflake curve

Figura 8: Construcción de "snowflake curve"de von Koch. [17]

Se puede considerar un antecedente al sistema de reescritura en cuestión al propuesto por Chomsky [4] para la representación sintáctica de lenguajes

naturales. Sin embargo, el propuesto por Lindenmayer es más orientado a la biología ya que buscaba modelar la forma en que las células se dividen en un organismo viviente.

3.2. Sistemas DOL

La clase más simple de los *L-systems* son conocidos como *DOL-systems*, y se caracterizan por ser deterministas (o no aleatorios) y libres de contexto. Se los puede caracterizar como a una cadena de caracteres constituídos por un grupo determinado de letras. Cada una de éstas está asociada a una relga de reescritura que implica de qué forma se la sustituye. El siguiente ejemplo es una muestra de cadenas formadas por las letras a y b. Las reglas de sustitución son las siguientes: $a \rightarrow ab$ y $b \rightarrow a$. Esto significa que a es reemplazado por ab en cada iteración, y a su vez b es sustituído por a.

Figure 1.3: Example of a derivation in a DOL-system

Figura 9: Derivación de un sistema DOL. [17]

3.3. Representación gráfica

La representación gráfica más extendida para transformar las cadenas de caracteres en dibujos se logra mediante la utilización de la "tortuga" LOGO [1]. Dicha "tortuga"se translada por el espacio de la pantalla dibujando su recorrido, dando pasos o cambiando su dirección. Generalmente ésta es utilizada con fines didácticos para la visualización de algoritmos. El estado de la tortuga se puede expresar mediante tres valores: (x, y, α). La posición se determina utilizando las coordenadas Cartesianas x e y; el ángulo α es la dirección hacia donde avanza. La distancia que recorre la tortuga a cada paso se llama d; la variación del

ángulo, δ. El proceso de dibujo de la tortuga se puede expresar mediante las siguientes instrucciones:

F Avanzar hacia adelante un paso de tamaño d. El estado de la tortuga cambia a (x', y', α) donde $x' = x + d\cos(\alpha)$ e $y' = y + d\sin(\alpha)$. Se dibuja una línea entre los puntos (x, y) y (x', y').

f Avanzar hacia adelante un paso de tamaño d sin dibujar una línea.

\+ Girar hacia la izquierda un ángulo δ. El estado siguiente de la tortuga es $(x, y, \alpha + \delta)$. Un ángulo positivo hace girar a la tortuga en un sentido antihorario.

\- Girar hacia la derecha un ángulo δ. El estado siguiente de la tortuga es $(x, y, \alpha - \delta)$.

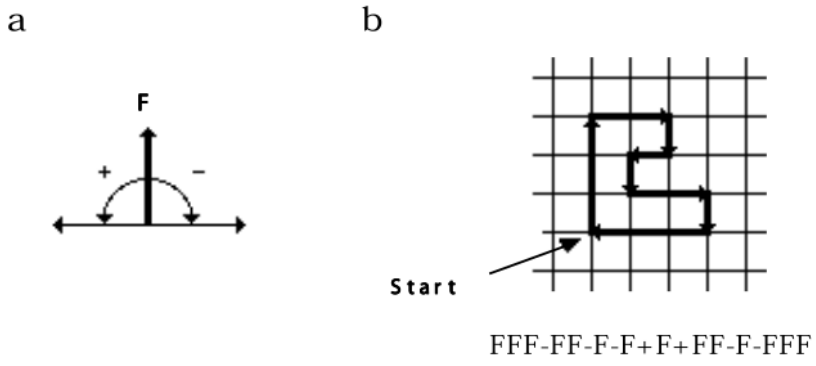


Figure 1.5: (a) Turtle interpretation of string symbols F, $+$, $-$. (b) Interpretation of a string. The angle increment δ is equal to 90°. Initially the turtle faces up.

Figura 10: Implementación gráfica de la "tortuga". [17]

3.4. Sonificación

Prusinkiweicsz sugiere aplicar los *L-systems* para la composición algorítmica utilizando los valores de la posición de la tortuga como parámetros asignables a la altura y la duración de notas musicales [16]. Ejemplos de empleo de dicho algoritmo para la composición son las piezas *Transitioning L-systems: Tramontana* de Michael Edwards y *Cells* de Hanspeter Kyburz [6] . En dichas obras

se *mapea* los resultados obtenidos a motivos musicales, que posteriormente son seleccionados o rechazados por el compositor durante el proceso creativo. La utilización de una técnica de mapeo para vincular el resultado o *output* de un algoritmo para el control de parámetros que resulten en la creación de una pieza musical es constitutiva en la práctica de la composición algorítmica. Se puede señalar en esta etapa el límite de la automaticidad de un proceso y el comienzo de la intervención de un agente humano que decide de forma personal cómo emplear el material generado. La forma de empleo de dicho material también está sujeta a las posibilidades que ofrece la tecnología en un momento dado. Se puede suponer que es por esta razón que históricamente la composición algorítmica o el arte generativo no fueran implementados en tiempo real. El advenimiento de nuevas posibilades permiten usos novedosos de los algoritmos previamente utilizados.

Stelios Manousakis propone la implementación de técnicas de síntesis de sonido no convencionales o *non-standard sound synthesis* para generar sonido a partir de los *L-systems* [12]. Dicho autor define a esta clase de síntesis como " un tipo abstracto de síntesis de sonido que describe al sonido mediante valores de tiempo y amplitud". Destaca el carácter experimental de esta técnica y su vínculación con el sonido digital y agrega "las técnicas no convencionales se relacionan más con modelos matemáticos y la abstracción compositiva que con el oído humano(como la síntesis espectral), la características físicas de los objetos(modelado físico) o reproducción de fuentes de sonido(técnicas basadas en sampleo)". Por último, concluye que este acercamiento a la generación de sonido puede considerarse un tipo de "microcomposición, uniendo al mundo de la síntesis de sonido con el de la notación en un solo sistema como una estrategia para logarar coherencia entre forma, estructura y material". El autor propone mapear el resultado de los algoritmos a tablas de forma de onda o *wavetables* y realizar procesos sobre éstos, por ejemplo modulaciones de la forma de onda basado en la *reescritura* de fragmentos. Una técnica de mapeo que guarde una relación estricta con la representación gráfica de un *L-system* se puede lograr vinculando la trayectoria del dibujo resultante en dos dimensiones x e y a dos *wavetables*.

3.5. Implementación

Se escogió *OpenFrameworks*, un *framework*(o conjunto de herramientas) orientado a la producción audiovisual desarrollado en C++, para implemen-

tar los *L-systems*. La elección se basó en que dicho entorno permite la fácil y eficiente manipulación de los valores resultantes o *output* del algoritmo para *mapearlos* a procesos que determinen su representación visual y sonora. El *framework* utilizado es similar a Processing, previamente empleado para la visualización de autómatas celulares, pero la capacidad de *OpenFrameworks* de poder escribir valores arbitrarios al *buffer* de salida de audio fue considerada esencial para lograr la sonificación.

En la primer instancia de desarrollo, se programó un proceso que creara de forma aleatoria *L-systems*. Se decidió lograr dicho cometido asignando los valores de las reglas de reescritura, el ángulo α, el número de iteraciones y la posición inicial de forma azarosa.

Generar los *L-systems* con aleatoriedad posee la ventaja de lograr un alto grado de variedad en los resultados obtenidos. Por otro lado, la impredictibilidad del *output* puede crear problemas para la manipulación del mismo. Debido a esto, se aplicaron límites que hicieran más fácil la visualización. Un ejemplo de éstos es la manipulación de la representación visual de los *L-systems* con el fin de que ocupen un lugar determinado dentro de la ventana de la aplicación y, en caso de que se excediera el mismo, forzar la vuelta al centro del espacio asignado.

Con el fin de sonificar los *L-systems*, se buscó *mapear* los valores de posición de la representación gráfica a valores que pudieran ser utilizados en una tabla de forma de onda o *wavetable* como muestras o *samples*. En un principio, se procuró normalizar las formas de onda en una etapa previa a ser escritas en el *buffer* de sonido. Para *mapear* el *output* a *wavetables* que puedieran ser transferidas sin problemas al *buffer* de salida de audio, se procuró que el número de pasos resultantes del proceso de *reescritura* del algoritmo fueran potencias de 2, con el fin de obtener producciones de 16, 64, 512, 1024 o 4096 pasos o muestras.

3.6. Conclusión

Utilizar *L-systems* para generar formas de onda arroja resultados cuyo uso se justifica esencialmente al encontrar una relación directa entre el resultado visual y sonoro. Generar *wavetables* a partir de la implementación de dichos algoritmos y cierto grado de aleatoriedad puede ser un punto de partida para una práctica experimental de síntesis sonora.

Se puede decir que es un método generativo ya que el proceso está sujeto a las instrucciones que sigue el algoritmo, en este caso la *reescritura*, y se desarrolla

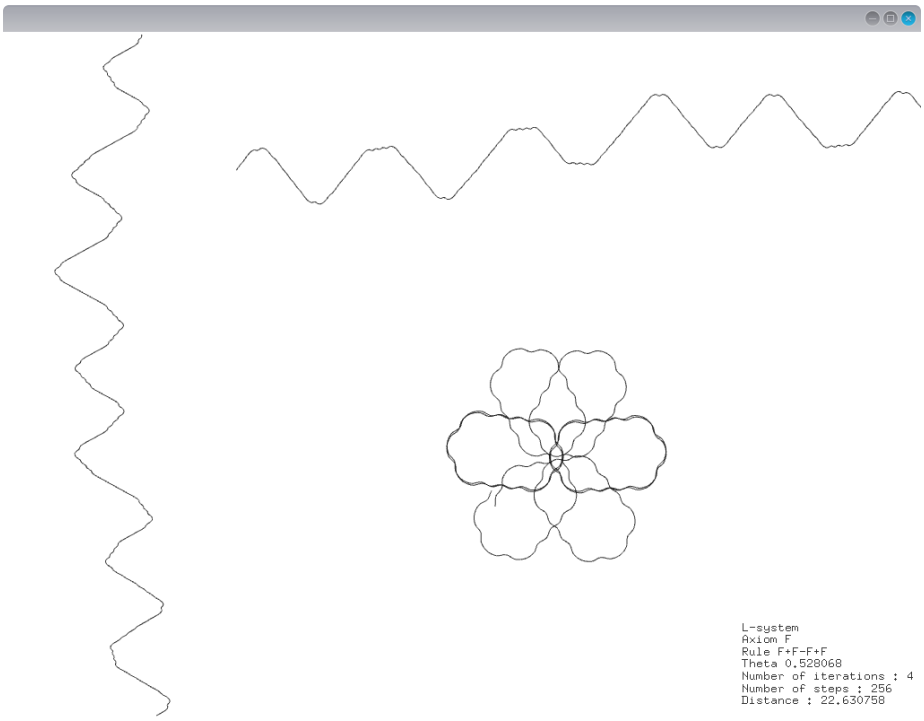


Figura 11: Captura de la aplicación, donde se puede observar en la parte central el *L-system* y sobre el margen izquierdo y superior las formas de onda obtenidas.

de forma autónoma.

El timbre resultante no es armónico. De todas formas, se puede percibir la presencia de componentes con distinta altura.

4. Algoritmos genéticos

4.1. Introducción

Los algoritmos genéticos son definidos por Whitley como "una familia de modelos computacionales inspirados en la evolución" [23]. Fueron caracterizados e investigados en un principio por John Holland [10]. Son utilizados generalmente para lograr optimizaciones de todo tipo emulando cómo se desarrollan los procesos evolutivos según la teoría de Darwin. Para lograr dicho cometido, *encodean* información en forma de una estructura de datos similar a los genes contenidos en los cromosomas de seres vivos. Se puede decir en forma general

que este algoritmo está basado en la forma en la que se desarrolla una población que sufre cambios aleatorios en sus genes y es posteriormente sometida a un proceso de selección(*fitness function*). Boden denomina a la utilización de este principio en un contexto artístico como *Evo-art* y plantea la ambigüedad que existe entre la decisión del artista de aplicar una selección arbitraria o que ésta esté incluida dentro del programa mismo [2].Trabajos destacados utilizando este algoritmo son Genetic Images y Galapagos del artista visual Karl Sims [21] y Vox Populi de Moroni[15], aplicado a la composición algorítmica.

4.2. Aplicación

Para implementar el algoritmo genético se decidió aplicarlo sobre la transformación de los sistemas de Lindenmayer. Con tal fin, en una primera instancia se *encodearon* las características de éstos: el ángulo α, la distancia d, la regla y el número de iteraciones en una estructura de datos, que representa de manera análoga la información contenida en "genes". Posteriormente, se aplica un proceso sobre estos genes en el cual se modifica aleatoriamente uno de sus componentes. De esta manera, se obtiene una nueva "generación"de *L-system* que conserva información del sistema que lo precediera.

En la instancia posterior a la alteración del gen, en la que se debiera aplicar la selección, se decidió introducir el *input* del usuario. De tal forma la *fitness function* es llevada a cabo por el criterio de un usario que decidiera si preservar o la evolución efectuada sobre el gen del *L-system*.

Referencias

[1] Harold Abelson and Andrea A DiSessa. *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press, 1986.

[2] Margaret A Boden and Ernest A Edmonds. What is generative art? *Digital Creativity*, 20(1-2):21–46, 2009.

[3] Dave Burraston and Ernest Edmonds. Cellular automata in generative electronic music and sonic art: a historical and technical review. *Digital Creativity*, 16(3):165–185, 2005.

[4] Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

[5] Peter Coveney, Roger Highfield, and N David Mermin. Frontiers of complexity: The search for order in a chaotic world. *Physics Today*, 49:58, 1996.

[6] Michael Edwards. Algorithmic composition: computational thinking in music. *Communications of the ACM*, 54(7):58–67, 2011.

[7] Philip Galanter. What is generative art? complexity theory as a context for art theory. In *In GA2003–6th Generative Art Conference*. Citeseer, 2003.

[8] Martin Gardner. Mathematical games: on cellular automata, self-reproduction, the garden of eden and the game"life". *Sci. Am.*, 224:112–117, 1971.

[9] Ira Greenberg. Processing. *Creative Coding and Computational Art. Friends of Ed*, 2007.

[10] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[11] Aristid Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299, 1968.

[12] Stelios Manousakis. Non-standard sound synthesis with l-systems. *Leonardo Music Journal*, pages 85–94, 2009.

[13] S Puckette MILLER. Theory and technique of eletronic music, 2007.

[14] Eduardo Reck Miranda. Granular synthesis of sounds by means of a cellular automaton. *Leonardo*, 28(4):297–300, 1995.

[15] Artemis Moroni, Jônatas Manzolli, Fernando Von Zuben, and Ricardo Gudwin. Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal*, pages 49–54, 2000.

[16] Przemyslaw Prusinkiewicz. Score generation with l-systems. In *ICMC*, 1986.

[17] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[18] Curtis Roads. Asynchronous granular synthesis, representations of musical signals, 1991.

[19] Curtis Roads. *Microsound*. MIT press, 2004.

[20] Daniel Shiffman. *The Nature of Code: Simulating Natural Systems with Processing*. Daniel Shiffman, 2012.

[21] Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

[22] Barry Truax. Real-time granular synthesis with a digital signal processor. *Computer Music Journal*, 12(2):14–26, 1988.

[23] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[24] Stephen Wolfram. *A new kind of science*, volume 5. Wolfram media Champaign, IL, 2002.

[25] Matthew Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3):193–200, 2005.

[26] Iannis Xenakis. Musiques formelles nouveaux principes formels de composition musicale. 1981.